R в действии

Третье издание

Роберт И. Кабаков

MANNING

ДМК

Краткое оглавление

Оглавление

Предисловие

Что толку в книжке, если в ней нет ни картинок, ни разговоров?

Алиса. *«Алиса в Стране чудес»*[1]

Оно чудесно и наделено сокровищами, способными удовлетворить всех от мала до велика, но не предназначено для робких духом.

Кью. *Сериал «Звездный путь: следующее поколение»*

Когда я начал писать эту книгу, я потратил довольно много времени на выбор хорошего эпиграфа. В итоге я остановился на этих двух. R – это потрясающе гибкая платформа и язык для исследования, визуализации и интерпретации данных. Я выбрал цитату из «Алисы в Стране чудес», чтобы передать суть современного статистического анализа – интерактивного процесса, состоящего из исследования, визуализации и интерпретации.

Вторая цитата отражает широко распространенное мнение о том, что R сложен в изучении. Я надеюсь показать вам, что это не так. R обладает настолько широкими возможностями и предлагает такое огромное число аналитических и графических функций (по последним подсчетам их более 50 000), что в одинаковой степени может вызывать бессознательный страх и у новичков, и у опытных пользователей. Однако в этом кажущемся безумии есть своя логика и поэзия. Вооружившись руководствами и инструкциями, вы сможете сориентироваться в огромном разнообразии возможностей и выбрать те инструменты, которые нужны для эффективного и элегантного решения вашей задачи.

Первое мое знакомство с R состоялось несколько лет назад, когда я подал заявление о приеме на должность консультанта по статистике. На встрече перед собеседованием будущий работодатель

[1] Перевод Н. Демуровой.

спросил меня, владею ли я языком R. Следуя стандартным советам специалистов по подбору персонала, я немедленно сказал «да» и приступил к его изучению. Я был опытным статистиком и исследователем с 25-летним опытом программирования в SAS и SPSS, свободно владел несколькими языками программирования. Что тут может быть сложного? Знаменитые последние слова.

Стремясь выучить этот язык программирования (как можно быстрее, ведь день собеседования приближался с угрожающей быстротой), я находил или тома, посвященные внутренней структуре языка, или многочисленные трактаты об отдельных продвинутых статистических методах, написанных специалистами в данной области для своих коллег. Встроенная справка была слишком лаконичной и служила скорее справочником, чем учебным пособием. Каждый раз, когда мне казалось, что я освоил общую логику и возможности R, обнаруживалось что-то новое, заставлявшее почувствовать себя невежественным и ничтожным.

Взявшись осваивать R, я подошел к процессу с позиции исследователя данных. Я пытался понять, что нужно сделать, чтобы успешно обработать, проанализировать и интерпретировать данные, и выделил следующие важные аспекты:

- доступ к данным (получение данных из разных источников);
- очистка данных (замена или удаление пропущенных значений, преобразование признаков в более удобный для обработки формат);
- аннотирование данных (чтобы можно было вспомнить, что представляет каждый их фрагмент);
- обобщение данных (вычисление описательных статистик, помогающих характеризовать данные);
- визуализация данных (потому что картинка на самом деле стоит тысячи слов);
- моделирование данных (выявление зависимостей и проверка гипотез);
- оформление результатов (подготовка таблиц и диаграмм достаточного для публикации качества).

Затем я постарался понять, как можно использовать R, чтобы выполнить каждую из этих задач. Поскольку я лучше всего учусь, обучая других, со временем я создал сайт (www.statmethods.net), на котором рассказываю все, что узнал сам.

Затем, спустя год, Марьян Бейс (Marjan Bace) из издательства Manning позвонила и спросила, не хочу ли я написать книгу про R. К этому времени у меня уже было 50 статей в научных журналах, четыре технических руководства, многочисленные главы в книгах и целая книга по методологии исследований, и что тут может быть сложного? Рискую повториться – знаменитые последние слова.

Первое издание вышло в 2011 году, а второе – в 2015-м. Над третьим изданием я начал работать два с половиной года назад. Описание R всегда было непростой задачей, но за последние несколько лет произошла почти что революция, обусловленная ростом популярности больших данных, широким внедрением программного обеспечения tidyverse (tidyverse.org), быстрой разработкой новых подходов к прогнозной аналитике и машинному обучению, а также появлением новых и более мощных технологий визуализации данных. Я хотел отразить все эти важные изменения в третьем издании.

Книгу, которую вы держите в руках, я мечтал иметь много лет назад. Я постарался написать для вас путеводитель по R, который позволит быстро овладеть всеми возможностями этого уникального продукта с открытым исходным кодом, не испытав разочарований и раздражения, которые пришлось испытать мне. Надеюсь, вам понравится.

P.S. Мне предложили ту должность, но я отказался. Однако знакомство с R развернуло мою карьеру в совершенно неожиданном направлении. Жизнь может быть забавной штукой.

Благодарности

Многие люди приложили значительные усилия, чтобы сделать эту книгу лучше:

- в первую очередь это Марьян Бейс (Marjan Bace), глава издательства Manning, которая предложила мне написать эту книгу;
- Себастьян Стирлинг (Sebastian Stirling), Дженифер Стоут (Jennifer Stout) и Карен Миллер (Karen Miller), редакторы-консультанты по аудитории (development editor) первого, второго и третьего изданий этой книги. Они провели многие часы в телефонных беседах со мной, помогая организовать материал, прояснить основные идеи и в целом сделать текст более интересным;
- Майк Шепард (Mike Shepard), научный редактор, который помог выявить непонятные места и представил свое экспертное мнение о тестировании кода. Я мог смело положиться на его подробные отзывы и суждения;
- Алекс Драгосавлевич (Aleks Dragosavljević), редактор-рецензент (review editor), помог найти рецензентов и координировал процесс рецензирования;
- Дейдре Хайам, помогавшая следить за процессом подготовки книги к печати, и ее команда: Сюзанна Дж. Фокс (Suzanne G. Fox), мой редактор, и Кэти Теннант (Katie Tennant), корректор;
- рецензенты, которые потратили много времени на внимательное чтение текста, находили опечатки и делали ценные замечания: Ален Ломпо (Alain Lompo), Алессандро Пуциелли (Alessandro Puzielli), Арав Агарвал (Arav Agarwal), Эшли Пол Итли (Ashley Paul Eatly), Клеменс Баадер (Clemens Baader), Дэниел Си Догерти (Daniel C Daugherty), Дэниел Кенни-Юнг (Daniel Kenney-Jung), Эрико Лендзиан (Erico Lendzian),

Джеймс Фронхофер (James Frohnhofer), Жан-Франсуа Морин (Jean-François Morin), Дженис Том (Jenice Tom), Джим Фронхофер (Jim Frohnhofer), Кей Энгельхардт (Kay Engelhardt), Келвин Микс (Kelvin Meeks), Кришна Шреста (Krishna Shrestha), Луис Фелипе Медейро Алвес (Luis Felipe Medeiro Alves), Марио Гизель (Mario Giesel), Мартин Перри (Martin Perry), Ник Дрозд (Nick Drozd), Николь Кенигштейн (Nicole Koenigstein), Роберт Самохил (Robert Samohyl), Тиклу Гангули (Tiklu Ganguly), Том Джеффрис (Tom Jeffries), Ульрих Гогер (Ulrich Gauger), Вишал Сингх (Vishal Singh);

- многие участники программы раннего доступа издательства Manning (Manning Early Access Program, MEAP), купившие книгу до того, как она была закончена, задавали великолепные вопросы, указывали на ошибки и давали ценные подсказки.

Все перечисленные помогли сделать эту книгу лучше и полнее.

Я также хотел бы поблагодарить многочисленных разработчиков, сделавших R такой мощной платформой для анализа данных. Этот список включает не только основную команду разработчиков, но и многочисленных сторонников, создавших и поддерживающих дополнительные пакеты, значительно расширяющие возможности R. В приложении E перечислены авторы всех пакетов, упомянутых в этой книге. Отдельно я хотел бы упомянуть Джона Фокса (John Fox), Хадли Викхама (Hadley Wickham), Франка Е. Харрела младшего (Frank E. Harrell Jr.), Дипаяна Саркара (Deepayan Sarkar) и Вильяма Ревилла (William Revelle), работами которых я восхищаюсь. Я старался как следует отразить их вклад, а ответственность за все ошибки и искажения, непреднамеренно допущенные в этой книге, лежит исключительно на мне.

На самом деле мне следовало бы начать эту книгу с благодарности моей жене и другу Кэрол Линн (Carol Lynn). Она не особенно интересуется статистикой или программированием, но неоднократно прочитала каждую главу и внесла множество исправлений и предложений. Никаким другим способом нельзя выразить свою любовь к другому человеку лучше, чем прочесть ради него текст по многомерной статистике. Она проявила необычайное терпение в вечера и выходные, которые я проводил в работе над этой книгой, выражая свою поддержку и проявляя такт и сочувствие. И за что это мне так повезло?

Есть еще два человека, которых я хочу поблагодарить. Один из них – мой отец, любовь которого к науке вдохновляла меня и помогла понять ценность данных. Другой человек – Гари К. Бургер (Gary K. Burger), мой руководитель в магистратуре. Гари заинтересовал меня статистикой и преподаванием, в то время как я собирался стать врачом. Это все он.

Об этой книге

Если вы выбрали эту книгу, скорее всего, у вас есть какие-то данные, которые нужно собрать, обобщить, преобразовать, исследовать, смоделировать, визуализировать или представить коллегам. Если это так, то R создан для вас! R стал всемирно известным языком программирования для статистического анализа и визуализации данных. В нем реализовано множество методов анализа данных, от самых простых до самых сложных и современных.

Как проект с открытым кодом он доступен для многих платформ, включая Windows, Mac OS X и Linux. Он постоянно развивается, и ежедневно появляются новые процедуры. Кроме того, R поддерживается большим и многоликим сообществом ученых и программистов, которые охотно помогут новичку советами.

Платформа R больше, пожалуй, известна за способность создавать красивые и сложные диаграммы, она может справиться с любой статистической задачей. Базовая версия содержит сотни функций для статистического анализа, управления данными и построения диаграмм. Однако некоторые особенно мощные методы реализованы в дополнительных пакетах, созданных независимыми авторами.

Эта широта возможностей имеет свою цену. Новичкам порой сложно понять, что такое R и как работать с этим языком. Даже самые опытные пользователи R с удивлением обнаруживают какие-то возможности, о которых не подозревали.

Третье издание «R в действии» – это руководство-путеводитель по R, знакомящее с самой платформой и ее возможностями. В книге описаны наиболее полезные функции базовой версии и более 90 наиболее часто используемых дополнительных пакетов. Основной упор в книге делается на практическое применение – на то, чтобы вы, руководствуясь прочитанным, могли проанализировать

ваши данные и изложить результаты коллегам. По окончании чтения этой книги вы будете иметь хорошее представление о том, как работает R и где можно получить дополнительную информацию. Вы научитесь применять разнообразные методы визуализации данных и обретете достаточно умений, чтобы справиться как с простыми, так и со сложными задачами анализа данных.

Что нового в третьем издании

В третье издание внесены многочисленные изменения, в том числе широко освещаются приемы применения tidyverse для управления данными и их анализа. Вот некоторые из наиболее заметных изменений.

Глава 2 (создание набора данных) теперь включает описание пакетов `readr`, `readxl` и `haven`, реализующих импорт данных. Также появился новый раздел о `tibbles`, современном решении поддержки наборов данных.

Главы 3 (основы управления данными) и 5 (дополнительные приемы управления данными) включают описание пакетов `dplyr` и `tidyr`, предназначенных для управления данными, их преобразования и обобщения.

Главы 4 (начало работы с диаграммами), 6 (базовые диаграммы), 11 (диаграммы средней сложности) и 19 (продвинутые методы работы с диаграммами) переписаны заново и подробно рассказывают о пакете `ggplot2` и его возможностях.

Глава 16 (кластерный анализ) описывает улучшенные приемы создания диаграмм и содержит новый раздел, посвященный оценке возможности кластеризации данных.

Глава 17 (классификация) содержит новый раздел, посвященный использованию иерархических диаграмм и диаграмм значений Шепли, помогающих в создании моделей черного ящика.

Глава 18 (продвинутые методы работы с пропущенными данными) была дополнена новыми разделами о методах *k*-ближайших соседей и случайного леса для подстановки отсутствующих значений.

Глава 20 (продвинутые приемы программирования) содержит новые разделы, посвященные нестандартным способам вычислений и визуальной отладке.

Глава 21 (создание динамических отчетов) приводит расширенное описание R Markdown и содержит новые разделы, посвященные параметризованным отчетам и распространенным ошибкам программирования.

Глава 22 (создание пакета) была полностью переписана и теперь включает описание способов использования новых инструментов для упрощенного создания пакетов, а также включает новые разделы о приемах распространения своих пакетов через CRAN, GitHub и веб-сайты.

Приложение A (графические пользовательские интерфейсы) было обновлено, чтобы отразить быстрые изменения в этой области.

Приложение B (настройка среды выполнения) было пересмотрено и теперь описывает новые методы настройки и рассказывает о влиянии потенциальных побочных эффектов на воспроизводимость исследований.

Приложение F (работа с большими наборами данных) содержит информацию о новых пакетах для работы с большими наборами данных, объем которых превышает объем ОЗУ, аналитических методах анализа наборов данных терабайтного размера и применении R в облачных службах.

На протяжении всей книги вам будут встречаться новые разделы, описывающие приемы использования RStudio для программирования, отладки и создания отчетов и пакетов. Наконец, в текст были внесены многочисленные обновления и исправления.

Кому адресована эта книга

Третье издание книги «R в действии» предназначено для всех, кто имеет дело с данными. Опыт в программировании статистических методов не требуется. Хотя эта книга доступна и новичкам, в ней содержится достаточно нового и полезного материала даже для опытных специалистов по R.

Пользователи, не владеющие познаниями в области статистики, но желающие использовать R для управления данными, их обобщения и представления в графическом виде, без особого труда освоят главы 1–6, 11 и 19. Главы 7 и 10 предполагают наличие у читателя базовых знаний математической статистики, а главы 8, 9 и 12–18 потребуют более глубоких познаний в этой области. Однако я старался писать каждую главу так, чтобы в ней было что-то интересное и полезное и для новичков, и для опытных статистиков.

Структура книги

Эта книга создана как путеводитель по R, с упором на методы, которые можно сразу применить для управления данными, их визуализации и анализа. Книга состоит из 22 глав, сгруппированных в четыре части: «Начало работы», «Базовые методы», «Методы средней сложности» и «Методы повышенной сложности». Дополнительные темы рассмотрены в семи приложениях.

Глава 1 начинается с общего обзора особенностей R, делающих его столь полезным для обработки данных. В главе рассказано, как установить платформу R и расширить ее возможности установкой дополнительных пакетов. Оставшаяся часть главы посвящена описанию интерфейса и способов запуска ее в интерактивном и пакетном режимах.

В главе 2 описаны многие методы импорта данных. Первая половина главы посвящена представлению структур, предназначенных для хранения данных в R. Во второй половине главы рассказывается о способах ввода данных с клавиатуры, импорта из текстовых файлов, веб-страниц, электронных таблиц, из других статистических программ и баз данных.

Глава 3 посвящена основам управления данными, включая сортировку, объединение и разбиение, а также преобразование, перекодировку и удаление переменных.

Глава 4 знакомит с синтаксисом создания диаграмм для визуализации данных. Здесь мы обсудим методы создания диаграмм, их изменения и сохранения в разных форматах.

Глава 5 основана на главах 3 и 4 и содержит описание функций (математических, статистических, текстовых) и управляющих конструкций (циклы, условное выполнение) для управления данными. Затем мы поговорим о том, как написать свою функцию на R и как сгруппировать данные различными способами.

Глава 6 рассказывает о создании наиболее распространенных одномерных диаграмм, таких как столбиковая и круговая диаграммы, диаграмма распределения плотности, диаграмма размахов (коробчатая диаграмма) и точечная диаграмма. Все эти диаграммы помогают изучать характер распределения значений одной переменной.

Глава 7 посвящена обобщению данных, включая использование описательных статистик и сводных таблиц. Затем в ней рассматриваются основные способы анализа взаимосвязей между двумя переменными, включая корреляцию, *t*-критерий Стьюдента, критерий хи-квадрат и непараметрические методы.

Глава 8 знакомит с применением методов регрессионного анализа для моделирования взаимосвязей между числовой переменной-откликом (outcome variable) и набором из одной или нескольких независимых переменных (predictor variables). Подробно рассмотрены методы подгонки этих моделей, оценки их адекватности и интерпретации.

В главе 9 представлены основы дисперсионного анализа и его разновидности. Обычно дисперсионный анализ выполняется с целью выяснить, как комбинации разных типов воздействий или разных условий влияют на числовую переменную-отклик. Также описаны методы оценки адекватности анализа и визуализации результатов.

Глава 10 подробно описывает анализ мощности статистических критериев. Она начинается с обсуждения проверки гипотез; затем мы поговорим о том, как определить объем выборки, необходимый для выявления влияния размера на заданный уровень достоверности. Это поможет вам повысить вероятность достижения желаемого результата при планировании экспериментов.

Глава 11 продолжает обсуждение, начатое в главе 5. В ней рассказано, как создать диаграммы для визуализации связей между двумя и более переменными. Рассматриваются разные типы двух- и трехмерных диаграмм рассеяния, матриц диаграмм рассеяния, графиков, коррелограмм и мозаичных диаграмм.

В главе 12 представлены аналитические методы для обработки данных, полученных из источников с неизвестными или смешанными распределениями, когда размеры выборок малы, когда часто встречаются выбросы или когда разработка статистического критерия на основании наблюдаемого распределения слишком сложна, в том числе метод повторной выборки (resampling) и бутстреп-анализ (bootstrapping), требующие большого объема вычислений и легко реализуемые в R.

Глава 13 вновь возвращается к обсуждению регрессионного анализа, начатому в главе 8, и охватывает подходы к анализу данных с распределением, отличным от нормального. Глава начинается с описания обобщенных линейных моделей. Затем более подробно рассматриваются случаи, когда нужно предсказать переменную-отклик, представленную либо категориальными (логистическая регрессия), либо счетными данными (пуассоновская регрессия).

Одна из сложностей, связанных с многомерными данными, – задача снижения их размерности. В главе 14 описаны методы, с помощью которых большое число коррелирующих друг с другом переменных преобразуются в меньший набор независимых переменных (метод главных компонент), а также методы выявления скрытой структуры в имеющемся наборе переменных (факторный анализ). Детально разобраны многочисленные этапы этих типов анализа.

Глава 15 описывает методы создания, обработки и моделирования временных рядов, визуализацию и разложение временных рядов, а также экспоненциальный подход и подход на основе интегрированной модели авторегрессии скользящего среднего (AutoRegressive Integrated Moving Average, ARIMA) к прогнозированию будущих значений.

Глава 16 иллюстрирует методы кластеризации наблюдений в естественные группы. Она начинается с обсуждения общих этапов комплексного кластерного анализа, а затем переходит к представлению методов иерархической кластеризации и сегментирования. Здесь мы рассмотрим несколько методов определения оптимального количества кластеров.

В главе 17 представлены популярные методы машинного обучения с учителем для классификации наблюдений по группам. По очереди будут рассмотрены деревья решений, случайные леса и метод опорных векторов. Здесь вы также узнаете о методах оценки точности каждого подхода и новых приемах анализа результатов.

Следуя стремлению познакомиться с наиболее актуальными методами анализа данных, в главе 18 мы поговорим о современных подходах к решению распространенной проблемы пропущенных значений в данных. В R реализованы разнообразные изящные подходы к анализу неполных в силу разных причин данных. Здесь описаны лучшие из этих методов и разъясняется, когда и какие стоит применять, а каких лучше избегать.

Глава 19 завершает обсуждение диаграмм знакомством с некоторыми наиболее сложными и полезными методами настройки осей координат, применения цветовых схем, шрифтов, легенд и аннотаций. Вы узнаете, как объединить несколько диаграмм в одну и, наконец, как превратить статический график в интерактивную веб-визуализацию.

Глава 20 рассматривает передовые методы программирования. Здесь вы познакомитесь с методами объектно-ориентированного программирования и приемами отладки. В этой главе также даются советы по эффективному программированию. Она будет особенно полезна тем, кто желает лучше понять, как работает R, и ее обязательно нужно прочитать, прежде чем переходить к главе 22.

В главе 21 описывается несколько методов создания красивых отчетов на R. Вы узнаете, как создавать веб-страницы, отчеты, статьи и даже книги из программ на R. Полученные документы могут включать код, таблицы результатов, графики и комментарии.

Наконец, в главе 22 представлено пошаговое руководство по созданию пакетов R. Это позволит вам писать сложные программы, эффективно документировать их и делиться ими с другими. Здесь подробно обсуждаются методы распространения и продвижения ваших пакетов.

В послесловии перечислены многие из лучших сайтов, которые следует посетить, чтобы научиться работать с R, влиться в сообщество пользователей R, получить ответы на возникшие вопросы и отслеживать изменения в этом стремительно развивающемся программном продукте.

И последнее, но не менее важное: семь приложений (от A до G) содержат дополнительные сведения по таким полезным темам, как графический пользовательский интерфейс R, настройка и обновление платформы, экспорт данных в другие приложения, использование R для матричных вычислений алгебры (по образцу MATLAB) и работа с большими наборами данных.

Мы также предлагаем дополнительную главу, доступную в интернете на сайте издательства по адресу https://www.manning.com/books/r-in-action-third-edition. Онлайн-глава 23 посвящена пакету `lattice`, предлагающему альтернативный подход к визуализации данных в R.

Совет специалистам по интеллектуальному анализу данных

Сфера интеллектуального анализа данных неразрывно связана с выявлением закономерностей в больших наборах данных. Многие специалисты по интеллектуальному анализу используют R, так как он обладает мощными аналитическими возможностями. Если вы занимаетесь анализом данных и желаете как можно быстрее освоить R, я рекомендую следующую последовательность чтения: глава 1 (введение), глава 2 (структуры данных и разделы, описывающие приемы импорта данных, имеющие к вам прямое отношение), глава 4 (основы управления данными), глава 7 (описательная статистика), глава 8 (разделы 1, 2 и 6, регрессия), глава 13 (раздел 2, логистическая регрессия), глава 16 (кластеризация), глава 17 (классификация) и приложение F (работа с большими наборами данных). Затем читайте другие главы по мере необходимости.

Примеры

Чтобы сделать книгу максимально полезной, я выбрал примеры из разных областей знаний, включая психологию, социологию, медицину, биологию, бизнес и технические науки. Ни один из примеров не требует специальных знаний в соответствующей области.

Наборы данных, используемые в этих примерах, были выбраны потому, что они позволяют формулировать интересные вопросы и имеют небольшой размер. Это позволяет сосредоточиться на рассматриваемом методе и быстро понять происходящее. Когда учишься новым методам, меньше – значит лучше. Наборы данных либо входят в состав дистрибутива R, либо доступны в виде дополнительных пакетов, которые можно скачать из интернета.

Принятые обозначения

В книге использованы следующие типографские обозначения:

- `моноширинный шрифт` – для программного кода, который нужно вводить именно так, как указано в книге;
- `моноширинный шрифт` также используется в основном тексте для обозначения фрагментов кода или ранее упомянутых объектов;
- *курсив* внутри программного кода указывает места заполнения. Его следует заменять подходящим текстом или значениями, соответствующими задаче. Например, *путь_к_моему_файлу* должен быть заменен путем к реальному файлу на вашем компьютере;
- R – это интерактивный язык, который информирует пользователя о готовности принять команду приглашением (> по умолчанию). Многие фрагменты программного кода в кни-

ге скопированы из интерактивных сеансов. Если вы видите строки кода, начинающиеся с >, не набирайте этот символ приглашения к вводу команды;

- пояснения к программному коду приведены в виде комментариев в тексте. В дополнение к этому некоторые пояснения обозначены нумерованными кружками, такими как ❶, которые отсылают к объяснению ниже в тексте;
- для экономии места или чтобы сделать текст более понятным, кое-где в вывод результатов интерактивных сеансов добавлены дополнительные пробелы или удален текст, который напрямую не относится к обсуждаемой теме.

Фрагменты выполняемого кода доступны в liveBook – онлайн-версии этой книги по адресу https://livebook.manning.com/book/r-in-action-Third-edition. Полный код всех примеров доступен для загрузки на сайте издательства Manning по адресу https://www.manning.com/books/r-in-action-third-edition и на GitHub, по адресу www.github.com/rkabacoff/RiA3. Чтобы получить максимум выгоды от этой книги, я рекомендую опробовать примеры по мере встречи с ними.

Наконец, существует распространенное правило, которое гласит, что если спросить двух статистиков, как проанализировать набор данных, то они дадут три ответа. Оборотная сторона этого утверждения: каждый ответ будет приближать вас к пониманию данных. Я не утверждаю, что тот или иной вид анализа является лучшим или единственным подходом к конкретной задаче. Используя навыки, полученные в этой книге, вы сможете поэкспериментировать с данными и посмотреть, что можно выяснить. R – интерактивный инструмент, предлагающий лучший способ обучения – экспериментирование.

Живое обсуждение книги

Приобретая книгу «R в действии», вы получаете бесплатный доступ к форуму, поддерживаемому издательством Manning Publications, где вы можете оставлять свои комментарии к книге, задавать технические вопросы и отвечать на них, а также получать помощь от автора и других пользователей. Чтобы получить доступ к форуму и зарегистрироваться на нем, откройте в веб-браузере страницу https://livebook.manning.com/book/r-in-action-third-edition/discussion. Узнать больше о форумах Manning и познакомиться с правилами поведения можно по адресу https://livebook.manning.com/#!/discussion.

Издательство Manning обязуется предоставить своим читателям место встречи, где может состояться содержательный диалог между отдельными читателями и между читателями и автором.

Но со стороны авторов отсутствуют какие-либо обязательства уделять форуму какое-то определенное внимание – их присутствие на форуме остается добровольным (и неоплачиваемым). Мы предлагаем задавать авторам стимулирующие вопросы, чтобы их интерес не угасал! Форум и архив с предыдущими обсуждениями остаются доступными на сайте издательства, пока книга продолжает издаваться.

Об авторе

Доктор наук Роберт Кабаков (Robert Kabacoff) – профессор вычислительной аналитики в Уэслианском университете (Wesleyan University) и опытный специалист по анализу данных с более чем 30-летним опытом программирования статистических вычислений и анализа данных в бизнесе, здравоохранении и государственных учреждениях. Вел курсы по анализу данных и статистическому программированию как для студентов, так и для аспирантов, а также поддерживает сайт Quick-R (statmethods.net) и сайт по визуализации данных с помощью R (rkabacoff.github.io/datavis).

Об иллюстрации на обложке

На обложке книги «R в действии» изображена иллюстрация под названием «Мужчина из Задара», взятая из альбома, посвященного хорватским национальным костюмам середины XIX века, который был составлен Николой Арсеновичем (Nikola Arsenovic).

В те дни по одежде было легко определить, где живет человек, чем занимается, а также его положение в обществе. Мы в издательстве Manning прославляем инициативу и изобретательность компьютерного бизнеса, украшая обложки книг изображениями, основанными на богатом разнообразии жизненного уклада народов многовековой давности, придавая новую жизнь иллюстрациям из таких коллекций, как эта.

Часть I

Начало работы

Добро пожаловать в «R в действии»! R – одна из наиболее популярных платформ для анализа данных и их визуализации из имеющихся в настоящее время. Это бесплатное программное обеспечение, распространяемое с открытым исходным кодом и способное работать в Windows, Mac OS X и Linux. Благодаря этой книге вы приобретете навыки, необходимые для овладения данной многофункциональной платформой и научитесь эффективно применять ее для обработки данных.

Книга разделена на четыре части. Первая часть посвящена установке базовой версии, знакомству с интерфейсом, импорту данных и преобразованию их в удобный для дальнейшего анализа вид. Глава 1 познакомит вас с окружением R. Сначала будет дан краткий обзор платформы R и ее особенностей, которые делают ее столь мощным инструментом для современного анализа данных. После краткого объяснения, как получить и установить R, последует описание пользовательского интерфейса на ряде простых примеров. Затем вы узнаете, как расширить функциональность базовой версии с помощью *дополнительных пакетов*, которые можно найти в репозиториях в интернете. И в конце главы приводится пример, на котором вы сможете проверить ваши новые умения.

После знакомства с интерфейсом R встает следующая задача – загрузить данные в программу. В современном богатом информацией мире данные могут поступать из разных источников и в разных форматах. В главе 2 описано множество методов импорта данных в R. В первой половине главы мы поговорим о форматах, в которых R может хранить данные, и как можно вводить данные вручную. Во второй части обсуждаются методы импорта данных из текстовых файлов, веб-страниц, электронных таблиц, других статистических программ и баз данных.

Данные редко поступают в формате, пригодном для использования. Зачастую приходится потратить немало времени, чтобы объединить данные из разных источников, очистить от ошибочных данных (неверно закодированные или несоответствующие данные, а также пропуски) и создать новые переменные (комбинированные, преобразованные или перекодированные), прежде чем можно будет приступать к поиску ответов на поставленные вопросы. В главе 3 описаны все основные способы управления данными в R, включая сортировку, объединение и сегментирование наборов данных, а также преобразование, перекодировку и удаление переменных.

Многие пользователи, впервые познакомившиеся с R, больше интересуются ее мощными графическими возможностями. Поэтому в главе 4 будет дан обзор *грамматики диаграмм* – пакета `ggplot2`. Для начала мы построим простую диаграмму, а затем будем последовательно расширять ее возможности, пока не получим комплексную визуализацию данных. В этой главе вы также узнаете, как задать основные настройки диаграммы и сохранить ее в различных графических форматах.

Глава 5 основана на сведениях, изложенных в главе 3. Она рассказывает, как использовать числовые (арифметические, тригонометрические, статистические) и текстовые функции (разбиение строк, конкатенация, замена) для управления данными. Для иллюстрации описываемых функций в этом разделе приводится множество примеров. Затем мы обсудим управляющие конструкции (циклы, условные операторы). Прочитав этот раздел, вы научитесь определять свои функции на R. Это позволит расширить возможности R, объединив многие команды в одну легко настраиваемую функцию. В заключение обсуждаются мощные методы реорганизации и группировки данных, которые часто бывают полезными при подготовке данных к дальнейшему анализу.

К концу части I вы будете готовы начать программировать в среде R. Приобретете навыки, необходимые для ввода данных и получения их из внешних источников, а также для их очистки. Кроме того, вы получите опыт создания, настройки и сохранения различных типов диаграмм.

1 Знакомство с R

В этой главе:

- установка R и RStudio;
- знакомство с языком программирования R;
- запуск программ.

В последние годы подходы к анализу данных принципиально изменились. С появлением персональных компьютеров и интернета объемы данных значительно возросли. Коммерческие компании обладают терабайтами данных о потребителях, правительственные, академические и частные исследовательские институты оперируют обширными архивными данными и материалами обследований по многим направлениям. Извлечение информации (не говоря уже о знаниях) из этих огромных объемов данных превратилось в самостоятельную отрасль. В то же время задача представления информации в легкодоступном и усвояемом виде значительно усложнилась.

Развитие наук, посвященных анализу данных (статистика, психометрика, эконометрика, машинное обучение), не отстает от взрывообразного роста объема данных. До эпохи персональных компьютеров и интернета новые статистические методы разрабатывались учеными-теоретиками, которые публиковали свои результаты в виде статей в специализированных журналах. Могли

пройти годы, прежде чем эти методы доходили до программистов и встраивались в программы статистической обработки данных. В наше время новые методы появляются *ежедневно*. Исследователи-статистики публикуют новые и усовершенствованные методы вместе с программным кодом, который их реализует, на общедоступных веб-сайтах.

Появление персональных компьютеров повлияло на подходы к анализу данных еще с одной стороны. Когда для анализа данных использовались большие ЭВМ, время их работы стоило дорого и его не хватало на всех. Поэтому аналитикам приходилось заранее тщательно определять все параметры анализа. Когда анализ завершался, результаты распечатывались на десятках или сотнях страниц. Аналитик должен был просмотреть их все, оставляя нужное и отсеивая лишнее. В ту пору появились многие популярные статистические пакеты (такие как SAS и SPSS), которые продолжают следовать этому алгоритму до сих пор.

С появлением недорогих и доступных персональных компьютеров произошла смена парадигмы. Теперь процесс анализа не требует предварительной установки всех параметров анализа и стал в значительной степени интерактивным. При этом результаты одного этапа анализа используются как исходные данные для следующего. На рис. 1.1 показана типичная схема анализа данных. На любом этапе анализа могут выполняться преобразование данных, вставка пропущенных значений, добавление или удаление переменных, после чего процесс продолжается. Завершается этот процесс, когда аналитик посчитает, что он или она полностью исследовал(а) данные и ответил(а) на все относящиеся к делу вопросы, на которые можно было ответить.

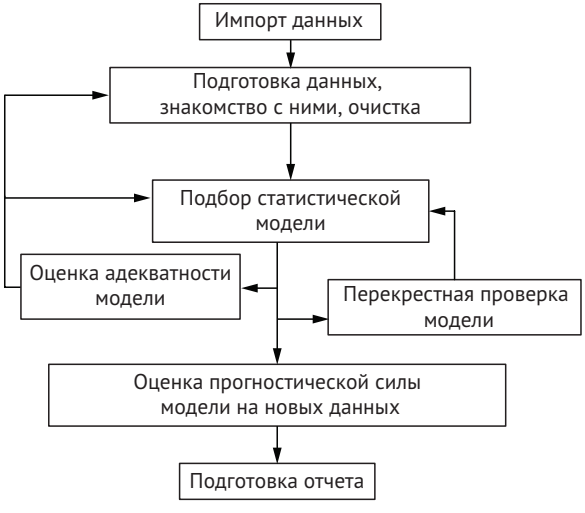


Рис. 1.1. Этапы типичного анализа данных

Появление персональных компьютеров (и особенно мониторов с высоким разрешением) также повлияло на способы представления результатов и их изучение. Изображение действительно может стоить тысячи слов, а люди весьма успешно извлекают информацию из визуальных образов. Современные методы обработки данных все больше опираются на графические способы представления результатов.

Современному исследователю необходимо получать данные из разных источников (систем управления базами данных, текстовых файлов, статистических программ и электронных таблиц), объединять фрагменты данных, маркировать их и очищать от ошибок, анализировать с помощью новейших методов, представлять результаты в наглядном и простом для интерпретации виде, а также включать результаты в привлекательные отчеты, которые не стыдно будет распространять среди заинтересованных лиц и общественности.

1.1. Зачем использовать R?

R – это язык программирования и среда для статистических вычислений и графического анализа, сходный с языком S, первоначально разработанным в Bell Labs. Это инструмент с открытым исходным кодом для анализа данных, который поддерживается большим и активным сообществом исследователей по всему миру. Однако существует много распространенных программ для статистической и графической обработки данных (таких как Microsoft Excel, SAS, IBM SPSS, Stata и Minitab). Так в чем преимущества R?

R обладает множеством уникальных особенностей, которые позволяют рекомендовать именно эту платформу:

- большинство коммерческих статистических программ стоят тысячи, если не десятки тысяч долларов. R – бесплатный программный продукт! Если вы преподаватель или студент, то выгода очевидна;
- R – мощная среда для статистических вычислений, в которой реализованы все основные виды анализа данных;
- в R реализованы сложные статистические процедуры, недоступные в других программах. На самом деле новые функции появляются еженедельно. Если используете SAS, то знаете, насколько маловероятным выглядит появление новых SAS PROC каждые несколько дней;
- R имеет современные графические возможности. Если вам нужно визуализировать сложные данные, то учтите, что в R

реализованы самые разнообразные и мощные методы анализа данных из доступных;

- R – мощная платформа для интерактивного анализа и исследования данных. С самого начала она разрабатывалась для поддержки подхода, показанного на рис. 1.1. Например, результаты любого этапа анализа можно сохранить, обработать и использовать в роли исходных данных для дополнительного анализа;
- зачастую невозможно получить данные из разных источников в пригодном для анализа виде. R может импортировать данные из самых разных источников, включая текстовые файлы, системы управления базами данных, другие статистические программы и специализированные хранилища данных, а также экспортировать данные в форматах всех этих систем;
- R предоставляет непревзойденную платформу, упрощающую программирование новых статистических методов. Она легко расширяется и имеет удобный и естественный язык, позволяющий, например, быстро запрограммировать недавно опубликованный метод;
- возможности R можно интегрировать в приложения, написанные на других языках, включая C++, Java, Python, PHP, Pentaho, SAS и SPSS. Это позволяет продолжать работать на знакомом языке, добавляя возможности R в приложения;
- R поддерживает разные операционные системы, включая Windows, Unix и Mac OS X. Эту платформу можно запустить практически на любом компьютере (я даже видел руководства по установке R на iPhone, что впечатляет, но вряд ли является хорошей идеей);
- для тех, кто не желает учить новый язык, существует множество графических пользовательских интерфейсов, в которых мощь R реализована в форме меню и диалогов.

Демонстрацию графических возможностей R можно видеть на рис. 1.2. На этой диаграмме показана взаимосвязь между стажем работы и заработной платой мужчин и женщин в шести отраслях, на основе данных, полученных в ходе обследования населения США в 1985 году. Технически это матрица диаграмм рассеяния, где пол отображается цветом и символом. Тенденции описываются с помощью линейной регрессии. Если эти термины, *диаграмма рассеяния* и *линейная регрессия*, вам незнакомы, не волнуйтесь. Мы рассмотрим их в последующих главах.

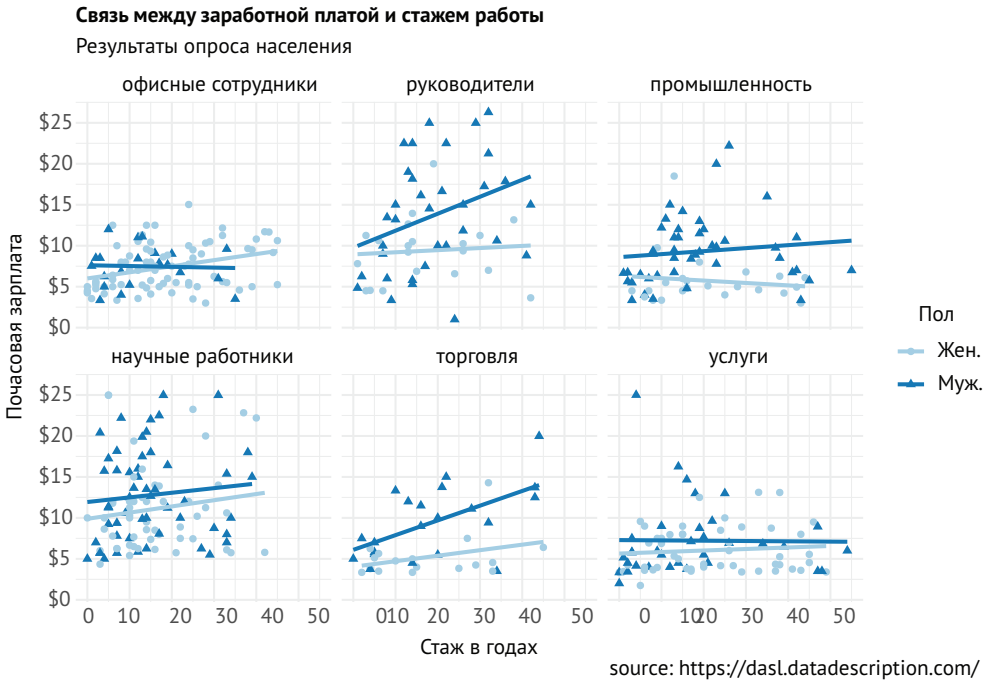


Рис. 1.2. Взаимосвязь между стажем работы и заработной платой мужчин и женщин в шести отраслях. Диаграммы, подобные этой, в R создаются всего несколькими строками кода (эта диаграмма создана с помощью пакета mosaicData)

Вот главное, что показывает этот график:

- взаимосвязи между стажем работы и заработной платой различаются в зависимости от пола и отрасли;
- в сфере услуг заработная плата мало зависит от стажа работы для обоих полов;
- на руководящих должностях заработная плата растет пропорционально стажу работы для мужчин, но не для женщин.

Являются ли эти различия реальными или их можно объяснить случайной изменчивостью выборки? Мы обсудим это далее в главе 8. Но важно отметить, что R позволяет создавать красивые и информативные графики простым и понятным способом. Создание подобных графиков в других статистических программах – сложная, а подчас невыполнимая задача.

К сожалению, для языка R свойственна крутая кривая обучения. Вследствие богатейшего набора возможностей объем документации и файлов справки очень велик. Кроме того, многие из этих возможностей реализованы в дополнительных модулях, созданных независимыми исследователями, поэтому справочная до-

кументация бывает разобщенной и труднодоступной. Научиться выполнять все виды анализа, реализованные в R, действительно очень непросто.

Задача этой книги – сделать овладение R быстрым и простым. Мы рассмотрим многие возможности R, чтобы вы смогли начать обработку своих данных, и подскажем, где можно получить дополнительную информацию. Начнем с установки программы.

1.2. Получение и установка R

R можно бесплатно скачать из «всеобъемлющего сетевого архива R» (Comprehensive R Archive Network, CRAN) по адресу http://cran.r-project.org. Предварительно скомпилированные файлы доступны для Linux, Mac OS X и Windows. Просто следуйте инструкциям по установке для вашей операционной системы. Позже мы обсудим, как расширить возможности R при помощи дополнительных модулей, называемых пакетами, – они также доступны в CRAN.

1.3. Работа в R

R – это чувствительный к регистру символов интерпретирующий язык программирования. Команды можно вводить по одной в строке приглашения к вводу (>) или запускать наборы команд, перечисленные в файлах. Типы данных очень разнообразны: векторы, матрицы, таблицы данных (похожи на наборы данных) и списки (коллекции объектов). Мы обсудим все эти типы данных в главе 2.

Основные функциональные возможности R реализуются при помощи встроенных и пользовательских функций, которые создают и управляют объектами. *Объектами* называются программные структуры, способные хранить значения. В R объектами является вообще все (данные, функции, диаграммы, результаты анализа и т. д.). У каждого объекта есть *атрибут класса* (один или несколько текстовых описателей), который определяет, как выводить, отображать, обобщать или как-то иначе управлять объектом.

В течение интерактивного сеанса все объекты хранятся в памяти. Основные функции доступны по умолчанию. Другие функции содержатся в пакетах, которые при необходимости следует подключать к текущему сеансу.

Инструкции состоят из имен функций и операторов присваивания. Для обозначения присваивания в R используется символ <- вместо привычного =.

Например, инструкция

```
x <- rnorm(5)
```

создает объект вектора с именем x, содержащий пять случайных значений из нормального распределения.

ПРИМЕЧАНИЕ. R позволяет использовать знак равенства = для присваивания. Однако очень немногие функции написаны с его использованием. Это нестандартный синтаксис, поэтому в некоторых ситуациях он не будет работать, и программисты на R поднимут вас на смех. Операцию присваивания допускается записывать в обратном порядке. Например, выражение `rnorm (5) -> x` эквивалентно предыдущей инструкции. И снова, так писать не принято, и я не рекомендую использовать такой стиль.

Комментарии начинаются с символа #. Любой текст после # игнорируется программой. Пример программы с комментариями можно увидеть в разделе 1.3.1.

1.3.1. Начало работы

Первый шаг на пути к использованию R – это, конечно же, установка. Инструкции по установке вы найдете на сайте CRAN. После установки запустите R. Если вы используете Windows, запустите R из меню Start (Пуск). В операционной системе Mac дважды щелкните на значке R в папке Applications (Приложения). В Linux введите команду `R` в командной строке. Любое из этих действий запустит R (см. рис. 1.3 в качестве примера).

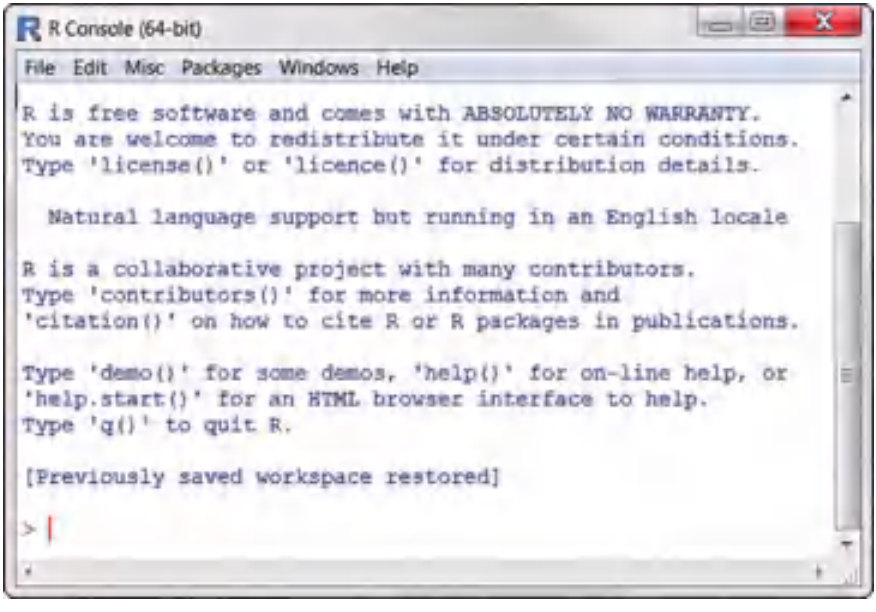


Рис. 1.3. Пример интерфейса R в операционной системе Windows

Чтобы освоиться с интерфейсом, давайте потренируемся на простом выдуманном примере. Представьте, что вы изучаете физическое развитие и собрали данные о возрасте и весе 10 младен-

цев первого года жизни (табл. 1.1). Вам интересно узнать закономерность распределения веса в зависимости от возраста.

Таблица 1.1. Возраст и вес 10 младенцев

Возраст (месяцы)	Вес (кг)	Возраст (месяцы)	Вес (кг)
01	4,4	09	7,3
03	5,3	03	6,0
05	7,2	09	10,4
02	5,2	12	10,2
11	8,5	03	6,1

В листинге 1.1 показан сеанс анализа. Вес и возраст вводятся в виде векторов с помощью функции `c()`, которая преобразует свои аргументы в вектор или список. Затем вычисляется среднее арифметическое и стандартное отклонение для значений веса, а также коэффициент корреляции между возрастом и весом, которые вычисляются с помощью функций `mean()`, `sd()` и `cor()` соответственно. После этого с помощью функции `plot()` отображается диаграмма, отражающая зависимость веса от возраста и позволяющая визуально наблюдать тенденцию. Функция `q()` завершает сеанс.

Листинг 1.1. Пример сеанса работы с R

```
> age <- c(1,3,5,2,11,9,3,9,12,3)
> weight <- c(4.4,5.3,7.2,5.2,8.5,7.3,6.0,10.4,10.2,6.1)
> mean(weight)
[1] 7.06
> sd(weight)
[1] 2.077498
> cor(age,weight)
[1] 0.9075655
> plot(age,weight)
```

Как можно видеть в листинге 1.1, средний вес этих 10 младенцев составляет 7,06 кг, а стандартное отклонение равно 2,08 кг и существует сильная линейная взаимосвязь между возрастом и весом (коэффициент корреляции равен 0,91). Эта взаимосвязь также видна на диаграмме рассеяния на рис. 1.4. Неудивительно, что по мере взросления младенцы в среднем становятся тяжелее.

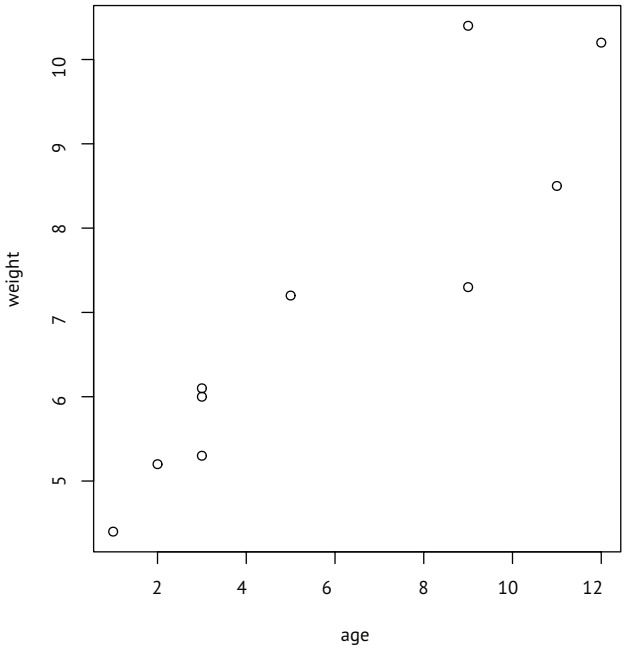


Рис. 1.4. Диаграмма рассеяния веса младенцев (weight, в килограммах) в зависимости от их возраста (age, в месяцах)

Диаграмма рассеяния на рис. 1.4 достаточно информативна, но выглядит не очень привлекательно. В последующих главах вы узнаете, как создавать более привлекательные и сложные диаграммы.

СОВЕТ. Чтобы получить представление о графических возможностях R, взгляните на примеры, представленные в разделе «Data Visualization with R» в документации (http://rkabacoff.github.io/datavis) и в статье «The Top 50 ggplot2 Visualizations – The Master List» (http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html).

1.3.2. Использование RStudio

Стандартный интерфейс R очень прост и предлагает ничем не примечательную командную строку для ввода инструкций. Для создания реальных проектов лучше использовать более полноценный инструмент, помогающий писать код и просматривать результаты. Для R было создано несколько таких инструментов, которые называют интегрированными средами разработки (Integrated Development Environment, IDE), включая Eclipse с расширением StatET, Visual Studio for R и RStudio Desktop.

RStudio Desktop (https://www.rstudio.com) – самый, пожалуй, популярный выбор. Эта среда разработки предлагает многооконный интерфейс с вкладками и инструментами для импорта данных, написания кода, отладки ошибок, визуализации вывода и создания отчетов.

RStudio распространяется бесплатно, как продукт с открытым исходным кодом, и поддерживает Windows, Mac и Linux. Поскольку RStudio является интерфейсом к R, перед установкой RStudio Desktop следует обязательно установить R.

> **СОВЕТ.** Для настройки интерфейса RStudio выберите меню Tools > Global Options... (Инструменты > Общие настройки...). На вкладке General (Общие) я рекомендую снять флажок Restore .RData into Workspace at Startup (Восстанавливать .RData в рабочей области при запуске) и выбрать значение Never (Никогда) для параметра Save Workspace to .RData on Exit (Сохранять рабочую область в .RData при выходе). Это обеспечит получение чистого окружения при каждом запуске RStudio.

Давайте повторно запустим код из листинга 1.1, но теперь в RStudio. В Windows запустите RStudio из меню Start (Пуск). В Mac дважды щелкните на значке RStudio в папке Applications (Приложения). В Linux введите команду `rstudio` в командной строке. На всех трех платформах появится один и тот же интерфейс (рис. 1.5).

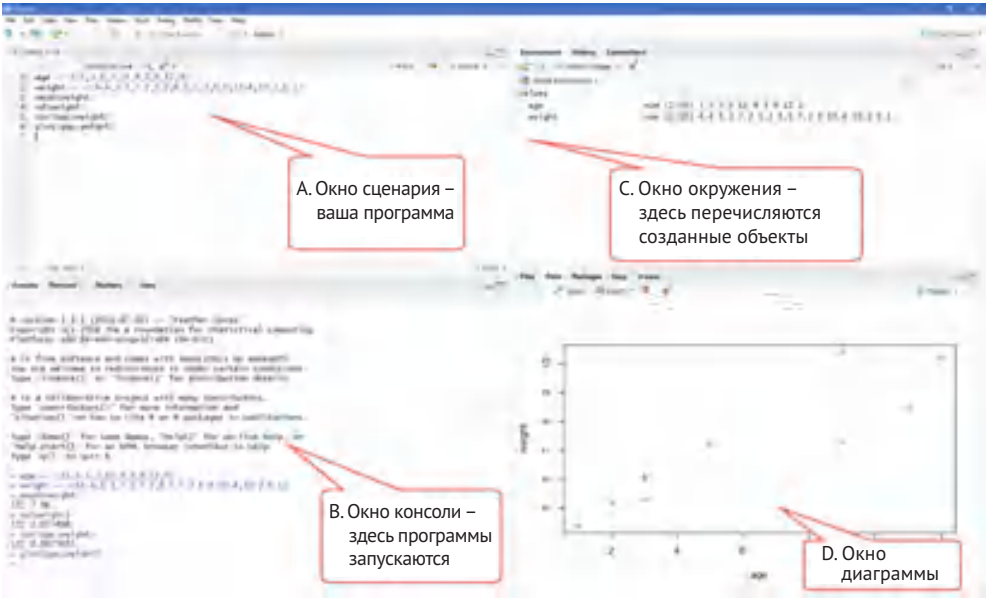


Рис. 1.5. RStudio Desktop

Окно сценария

В меню File (Файл) выберите пункт New File > R Script (Новый файл > Сценарий R). В левом верхнем углу откроется новое окно сценария (рис. 1.5 А). Введите в него код из листинга 1.1.

По мере ввода редактор будет подсвечивать синтаксис и предлагать варианты завершения кода (рис. 1.6). Например, по мере ввода `plot` появится всплывающее окно с именами доступных функций, начинающимися с букв, набранных к текущему моменту. Вы можете использовать клавиши со стрелками вверх и вниз, чтобы выбрать функцию из списка, и нажать Tab, для ее выбора. В круглых скобках, следующих за именем функции, можно нажать Tab, чтобы просмотреть список параметров функции. Нажатие Tab в кавычках приводит к завершению путей к файлам.

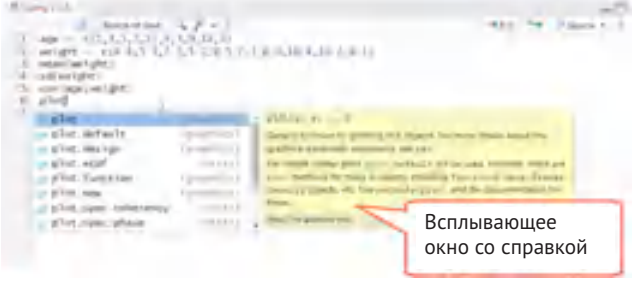


Рис. 1.6. Окно сценария

Чтобы выполнить код, выделите его и щелкните на кнопке Run (Выполнить) или нажмите Ctrl+Enter. Комбинация Ctrl+Shift+Enter запустит весь сценарий, а не только выделенный код.

Чтобы сохранить сценарий, щелкните на значке Save (Сохранить) или выберите пункт меню File > Save (Файл > Сохранить). В открывшемся диалоге выберите имя для сценария и папку, куда его следует сохранить. По соглашению файлы сценариев получают расширение `.R`. Имя файла сценария выводится на вкладке окна с красной звездочкой, если текущая версия не была сохранена.

Окно консоли

Код запускается в окне консоли (рис. 1.5 В). По сути, это та же консоль, которую можно видеть в базовом интерфейсе R. Код из окна сценария в окно консоли можно отправлять с помощью команды Run (Выполнить) или вводить команды непосредственно в этом окне, в строке приглашения к вводу (`>`).

Если приглашение к вводу меняется на знак плюс (`+`), то это означает, что интерпретатор ожидает завершения инструкции. Такое часто происходит, когда инструкция слишком длинная и не умещается на одной строке или если в коде есть непарные круглые

скобки. Вы можете отменить ввод инструкции и вернуться в командную строку, нажав клавишу Esc.

Кроме того, клавиши со стрелками вверх и вниз будут циклически переключать прошлые выполненные инструкции. Вы можете отредактировать инструкцию и повторно запустить ее клавишей Enter. Щелчок на значке с изображением метлы удаляет текст из окна.

Окна окружения и истории

Любые создаваемые объекты (в данном примере `age` и `weight`) будут появляться в окне окружения (рис. 1.5 C), а история выполненных команд будет сохраняться в окне истории – вкладка History (История) правее вкладки Environment (Окружение).

Окно диаграммы

Любые диаграммы, созданные сценарием, появятся вокне диаграммы (рис. 1.5 D). Панель инструментов этого окна позволяет циклически перемещаться по созданным диаграммам. Кроме того, окно диаграммы можно масштабировать, чтобы рассматривать диаграммы в разных масштабах, экспортировать диаграммы в несколько форматов и удалить одну или все диаграммы, созданные к настоящему моменту.

1.3.3. Как получить помощь

R имеет обширную справку. Научитесь ориентироваться в ней, и это поможет вам в работе. Встроенная система помощи содержит подробные разъяснения, ссылки на документацию и примеры для каждой функции из установленных пакетов. Справку можно вызвать при помощи функций, перечисленных в табл. 1.2.

Таблица 1.2. Функции вызова справки в R

Функция	Описание
`help.start()`	Общая справка
`help("foo") или ?foo`	Справка по функции `foo` (кавычки необязательны)
`help(package="foo")`	Справка по пакету `foo`
`help.search("foo") или ??foo`	Поиск в справке записей, содержащих `foo`
`example("foo")`	Примеры использования функции `foo` (кавычки необязательны)
`data()`	Список всех демонстрационных примеров данных, содержащихся в загруженных пакетах
`vignette()`	Список всех доступных руководств по загруженным пакетам
`vignette("foo")`	Список руководств по теме `foo`

Функция `help.start()` открывает окно браузера с перечнем доступных руководств разного уровня сложности, часто задаваемых

вопросов и ссылок на справочные материалы. То же самое можно получить выбором пункта меню Help > R Help (Справка > Справка по R). Функция `vignette()` вызывает список вводных статей в формате PDF или HTML. Такие статьи имеются не для всех пакетов.

Все файлы справки имеют схожий формат (рис. 1.7). Они включают заголовок и краткое описание, за которыми следует описание синтаксиса и параметров функции. Детали вычислений приводятся в разделе Details (Подробности). В разделе See Also (См. также) описаны связанные функции и ссылки на них. Страницы справки почти всегда заканчиваются примерами, иллюстрирующими типичное использование функции.

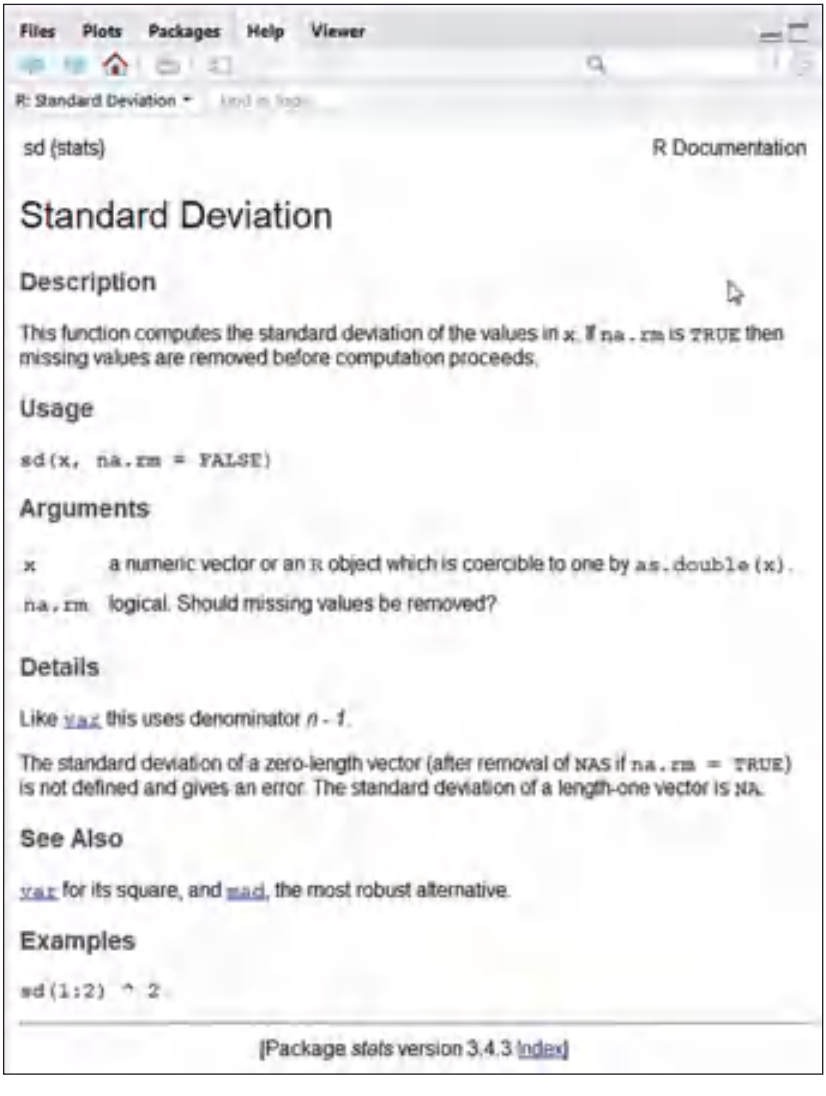


Рис. 1.7. Окно справки

Как видите, R предоставляет обширную справку, и умение ориентироваться в ней пойдет вам на пользу. Крайне редко, когда в сеансах работы с R я не использую справку, чтобы получить подробную информацию (параметры или возвращаемые значения) о какой-нибудь функции.

1.3.4. Рабочее пространство

Рабочее пространство – это текущее рабочее окружение R, включающее все созданные объекты (векторы, матрицы, функции, таблицы данных или списки). Текущий рабочий каталог – это каталог, где хранятся файлы с данными и куда по умолчанию сохраняются результаты. Узнать путь к рабочему каталогу можно с помощью функции `getwd()`. Назначить рабочий каталог можно с помощью функции `setwd()`. Чтобы импортировать файл, который находится не в рабочем каталоге, нужно указать полный путь к нему. Всегда заключайте имена файлов и каталогов в кавычки. В табл. 1.3 перечислены некоторые стандартные команды для управления рабочим пространством.

Таблица 1.3. Функции для управления рабочим пространством в R

Функция	Описание
`getwd()`	Выводит имя текущего рабочего каталога
`setwd("mydirectory")`	Назначает *mydirectory* текущим рабочим каталогом
`ls()`	Выводит список объектов в текущем рабочем пространстве
`rm(objectlist)`	Удаляет объекты, перечисленные в списке *objectlist*
`help(options)`	Выводит справку о доступных параметрах
`options()`	Позволяет просмотреть или установить текущие параметры
`save.image("myfile")`	Сохраняет рабочее пространство в файл *myfile* (по умолчанию `.Rdata`)
`save(objectlist, file="myfile")`	Сохраняет указанные объекты в файл *myfile*
`load("myfile")`	Загружает рабочее пространство в текущий сеанс

Чтобы увидеть эти функции в действии, рассмотрим следующий пример (листинг 1.2).

Листинг 1.2. Пример использования функций для управления рабочим пространством R

```
setwd("C:/myprojects/project1")
options()
options(digits=3)
```

Первая функция назначает `C:/myprojects/project1` текущим рабочим каталогом. Вторая выводит текущие значения параметров, а третья задает формат вывода чисел с тремя цифрами после запятой.

Обратите внимание, что в пути к каталогу в функции `setwd()` используются прямые слеши (/). R воспринимает обратный слеш (\) как экранирующий символ. Даже работая с R в Windows, используйте прямые слеши в путях к файлам и каталогам. Отметьте также, что функция `setwd()` не создает указанный каталог, если он не существует. Если необходимо создать каталог, используйте функцию `dir.create()`, а уже затем вызывайте `setwd()`, чтобы сделать этот каталог рабочим.

1.3.5. Проекты

Старайтесь хранить свои проекты в отдельных каталогах. RStudio предлагает для этого простой механизм. Выберите пункт меню File > New Project... (Файл > Новый проект...) и укажите либо New Directory (Новый каталог), чтобы создать проект в новом рабочем каталоге, либо Existing Directory (Существующий каталог), чтобы связать проект с существующим рабочим каталогом. Все программные файлы, история команд, отчеты, диаграммы и данные будут сохранены в каталоге проекта. Переключаться между проектами можно с помощью раскрывающегося меню Project (Проект) в верхней правой части окна RStudio.

В файлах проекта легко запутаться. Поэтому я советую создать несколько подкаталогов в основном каталоге проекта. Обычно я создаю каталог *data* для хранения файлов с исходными данными, каталог *img* для файлов изображений и создаваемых диаграмм, каталог *docs* для проектной документации и каталог *reports* для отчетов. Сценарии на R и файл *README* я храню в основном каталоге. При наличии нескольких сценариев на R, выполняемых последовательно, я нумерую их (например, *01_import_data.R*, *02_clean_data.R* и т. д.). Файл *README* – это текстовый файл, содержащий информацию об авторе, дате создания, заинтересованных сторонах и их контактах, а также цели проекта. Через полгода эта информация помогает мне вспомнить, что я сделал и зачем.

1.4. Пакеты

В базовой установке R обладает обширными возможностями. Однако некоторые наиболее впечатляющие возможности реализованы в дополнительных модулях, которые можно загрузить и установить. Существует более 10 000 созданных пользователями модулей, называемых *пакетами* (packages), которые вы можете загрузить с http://cran.r-project.org/web/packages. В них заключены почти безграничные возможности – от анализа геопространственных данных до масс-спектроскопии белков и анализа психологических тестов! В этой книге мы познакомимся со многими из них.

Особого внимания заслуживает один из наборов пакетов – `tidyverse`. Это относительно новая коллекция, предлагающая целостный и интуитивно понятный подход к обработке и анализу данных. Преимущества, предлагаемые пакетами из набора `tidyverse` (такими как `tidyr`, `dplyr`, `lubridate`, `stringr` и `ggplot2`), меняют подходы к разработке кода на R, и мы будем часто использовать эти пакеты. На самом деле необходимость описания особенностей использования этих пакетов для анализа и визуализации данных и послужила основным мотивом для выпуска этого третьего издания книги.

1.4.1. Что такое пакеты?

Пакеты – это коллекции функций на R, данных и скомпилированного программного кода в определенном формате. Каталог, где пакеты хранятся на вашем компьютере, называется *библиотекой*. Функция `.libPath()` показывает, где расположена ваша библиотека, а функция `library()` выводит названия всех имеющихся в библиотеке пакетов.

В дистрибутив R уже входит стандартный набор пакетов (включая `base`, `datasets`, `utils`, `gr-Devices`, `graphics`, `stats` и `methods`). В них уже содержатся разнообразные функции и наборы данных, доступные по умолчанию. Также есть возможность скачивать и устанавливать дополнительные пакеты. После установки они загружаются в ходе сеанса по мере необходимости. Функция `search()` выводит названия загруженных и готовых к использованию пакетов.

1.4.2. Установка пакета

В R существует множество функций для управления пакетами. Установить пакет можно с помощью функции `install.packages()`. Например, есть такой пакет, как `gclus`, который содержит функции для создания улучшенных диаграмм рассеяния. Этот пакет можно скачать и установить вызовом функции `install.packages("gclus")`.

Пакет нужно установить только один раз. Однако, как и любые другие программы, пакеты часто обновляются их разработчиками. Обновить все установленные пакеты можно вызовом функции `update.package()`. Для получения информации об установленных пакетах используйте функцию `installed.packages()`. Она выведет список всех установленных пакетов с номерами их версий, названиями пакетов, от которых они зависят, и другой информацией.

Устанавливать и обновлять пакеты можно также из интерфейса RStudio. Выберите вкладку Packages (Пакеты) в окне справа внизу. Введите имя (или часть имени) в поле поиска в правом верхнем углу этого окна со вкладками. Поставьте галочки напротив пакетов, которые вы хотите установить, и щелкните на кнопке Install (Установить) или Update (Обновить), чтобы обновить уже установленные пакеты.

1.4.3. Загрузка пакета

В процессе установки пакет сначала скачивается с сайта CRAN в вашу библиотеку. Для использования этого пакета в текущем сеансе нужно загрузить его вызовом функции `library()`. Например, чтобы использовать пакет `gclus`, введите команду `library(gclus)`.

Разумеется, прежде чем загрузить пакет, его необходимо установить. В течение сеанса достаточно загрузить пакет только один раз. При необходимости можно настроить рабочее пространство так, чтобы часто используемые пакеты загружались автоматически в начале каждого сеанса. Настройка рабочего окружения подробно описана в приложении B.

1.4.4. Получение информации о пакете

После загрузки пакета становятся доступны новые функции и наборы данных. Небольшие наборы данных поставляются вместе с демонстрационным программным кодом, что позволяет протестировать новые возможности. Справочная система содержит описание каждой функции (с примерами) и информацию о каждом встроенном наборе данных. Функция `help(package="имя_пакета")` выведет краткое описание указанного пакета и список всех входящих в него функций и наборов данных. Вызов `help()` с именами этих функций или наборов данных позволит выяснить новые детали. Эту информацию можно также найти на сайте CRAN в виде руководства в формате PDF.

Чтобы получить справку по пакету в интерфейсе RStudio, щелкните на вкладке Packages (Пакеты) в окне внизу справа, введите имя пакета в окне поиска и щелкните на имени пакета.

Распространенные ошибки в программировании на R

Существует ряд распространенных ошибок, которые часто допускают и новички, и опытные программисты. Если программа выдает сообщение об ошибке, проверьте, не сделали ли вы что-то из нижеперечисленного:

- *использовали неправильный регистр:* `help()`, `Help()` *и* `HELP()` *– это три разные функции (только первое имя правильное);*
- *забыли поставить кавычки там, где они необходимы:* `install.packages(«gclus»)` *работает, а* `install.packages(gclus)` *выдаст сообщение об ошибке;*
- *забыли добавить круглые скобки в вызов функции: например,* `help()` *– правильный вызов, а* `help` *– нет. Скобки должны добавляться, даже когда функция вызывается без аргументов;*
- *использовали* `\` *в пути к файлу в операционной системе Windows: R воспринимает обратный слеш как экранирующий символ. Вызов* `setwd(«c:\mydata»)` *сгенерирует сообщение об ошибке. Используйте* `setwd(«c:/mydata»)` *или* `setwd(«c:\\mydata»)`*;*

- *использовали функцию из пакета, который еще не загрузили. В пакете* `gclus` *имеется функция* `order.clusters()`. *Если попытаться вызвать ее до загрузки пакета* `gclus`, *то появится сообщение об ошибке.*

Сообщения об ошибках в R могут быть непонятными, однако появление многих из них можно предотвратить, если внимательно следовать вышеперечисленным правилам.

1.5. Передача вывода на ввод: повторное использование результатов

Одна из наиболее полезных особенностей R – возможность сохранить результаты анализа и использовать в качестве входных данных в дополнительном анализе. Рассмотрим пример, воспользовавшись одним из наборов данных, распространяемых вместе с R. Если какие-то детали будут вам непонятны, не волнуйтесь. Здесь важно понять общий принцип, а не частности.

Вместе с R распространяется множество встроенных наборов данных, на которых можно попрактиковаться в анализе данных. Один из таких наборов, с именем `mtcars`, содержит информацию о 32 автомобилях, собранную в ходе дорожных испытаний журналом «Motor Trend». Предположим, нам нужно описать взаимосвязь между топливной эффективностью автомобиля и его весом.

Для начала можно попробовать выполнить простую линейную регрессию, предсказывающую, сколько миль проедет на одном галлоне[1] топлива (`mpg`) автомобиль с заданным весом (`wt`):

```
lm(mpg~wt, data=mtcars)
```

Результаты появятся на экране, но не будут сохранены.

Ту же регрессию можно вычислить с сохранением результатов в объекте:

```
lmfit <- lm(mpg~wt, data=mtcars)
```

Операция присваивания создаст объект списка с именем `lmfit` и сохранит в нем обширную информацию с результатами анализа (включая прогнозируемые значения, остатки, коэффициенты регрессии и т. д.). В этом случае на экран ничего не выводится, но результаты сохраняются, и их можно вывести на экран или подвергнуть дальнейшей обработке.

Вызов `summary(lmfit)` отобразит сводку результатов, а вызов `plot(lmfit)` выведет прогностические графики. Инструкция `cook<-cooks.distance(lmfit)` сгенерирует и сохранит затронутые статистики, а `plot(cook)` выведет ее график. Чтобы спрогнозировать расстоя-

[1] 1 галлон ≈ 4,5 л в Англии и ≈ 3,7 л в США. – *Прим. перев.*

ние в милях, которое проедет на одном галлоне автомобиль с некоторым весом, нужно выполнить вызов `predict(lmfit, mynewdata)`.

Чтобы узнать, что возвращает заинтересовавшая вас функция, загляните в раздел Value (Значение) на странице справки для этой функции. Например, вызвав `help(lm)` или `?lm`, вы узнаете, что сохранит операция присваивания результатов этой функции объекту.

1.6. Работа с большими массивами данных

Программисты часто спрашивают меня, может ли R обрабатывать большие массивы данных. Как правило, они работают со значительными объемами данных, собранных в процессе исследований климата или генетики. R хранит объекты в памяти, поэтому объем ОЗУ является ограничивающим фактором. Например, на моем девятилетнем компьютере с операционной системой Windows и 2 Гбайт оперативной памяти я могу обрабатывать наборы данных с 10 млн элементов (100 признаков 100 000 объектов). На iMac с 4 Гбайт оперативной памяти без особых затруднений можно обрабатывать наборы данных, включающие 100 млн элементов.

Но есть два аспекта, которые следует учитывать: размер набора данных и применяемые статистические методы. R способен обрабатывать наборы данных в диапазоне от гигабайта до терабайта, но для этого необходимо применять специальные процедуры. Подробнее об анализе больших массивов данных рассказывается в приложении F.

1.7. Учимся на примере

Закончим эту главу примером, объединяющим многие рассмотренные идеи. Вот задание:

1. Откройте общий файл справки и загляните в раздел Introduction to R (Введение в R).
2. Установите пакет `vcd` (пакет для визуализации категориальных данных, который мы подробно рассмотрим в главе 11).
3. Выведите список всех функций и наборов данных в этом пакете.
4. Загрузите пакет в текущий сеанс и прочтите описание набора данных `Arthritis`.
5. Выведите этот набор данных на экран (набрав его имя в командной строке).
6. Запустите пример, прилагаемый к набору данных `Arthritis`. Не волнуйтесь, если результаты покажутся вам непонятными. Если в двух словах, то они показывают, что страдающие артритом пациенты, получавшие лекарство, выздоравливали гораздо быстрее, чем получавшие плацебо.

В листинге 1.3 приводится программный код, который вам понадобится, а некоторые результаты приведены на рис. 1.8. Как показывает это упражнение, с помощью короткого фрагмента программного кода можно сделать многое.

Листинг 1.3. Работа с новым пакетом

```
help.start()
install.packages("vcd")
help(package="vcd")
library(vcd)
help(Arthritis)
Arthritis
example(Arthritis)
```

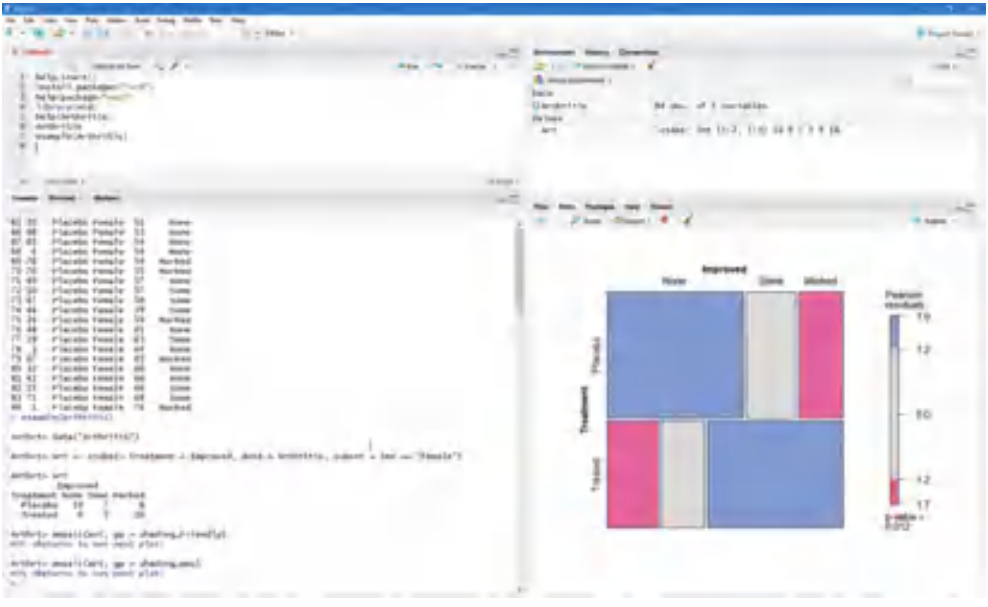


Рис. 1.8. Окно RStudio после выполнения кода в листинге 1.3

В этой главе мы рассмотрели некоторые сильные стороны R, особенно привлекательные для студентов, исследователей, статистиков и аналитиков, желающих анализировать свои данные. Мы рассмотрели порядок установки платформы R и расширения ее возможностей с использованием дополнительных пакетов. Исследовали основные характеристики интерфейса и создали несколько графиков для примера. R – сложная платформа, поэтому мы потратили некоторое время на знакомство с ее справочной системой. Надеюсь, вы уже почувствовали, насколько мощным может быть бесплатное программное обеспечение.

Теперь, установив R и RStudio, можно приступать к анализу данных. В следующей главе мы познакомимся с типами данных, поддерживаемыми в R, и узнаем, как импортировать данные из текстовых файлов, других программ и систем управления базами данных.

Итоги

- R предоставляет мощную интерактивную среду для анализа и визуализации данных.
- RStudio – это интегрированная среда разработки, делающая программирование на R проще и продуктивнее.
- Пакеты – это бесплатные дополнительные модули, расширяющие возможности платформы R.
- R имеет обширную справочную систему, и умение ее использовать значительно облегчит программирование и повысит его эффективность.

2 Создание набора данных

В этой главе:

- структуры данных в R;
- ввод данных;
- импорт данных;
- аннотирование набора данных.

Первый этап в анализе любых данных – создание набора данных, содержащего информацию для изучения в подходящем формате. В R эта задача распадается на:

- выбор типа данных;
- ввод или импорт данных в выбранном формате.

Разделы 2.1 и 2.2 содержат описание многочисленных типов данных, используемых в R. В частности, в разделе 2.2 описываются векторы, факторы, матрицы, таблицы данных и списки. Знакомство с этими типами данных (и обозначениями, используемыми для доступа к их элементам) очень поможет вам понять, как работает R. Этому разделу стоит уделить особое внимание.

Раздел 2.3 охватывает разные способы импорта данных в R. Данные можно вводить вручную или импортировать из внешнего источника. Таким источником могут быть текстовые файлы, электронные таблицы, статистические программы и системы управле-

ния базами данных. Например, я обычно работаю с данными, которые изначально хранятся в базах данных SQL, но иногда извлекаю их из наборов данных SAS и SPSS. Возможно, вам достаточно будет использовать один или два метода из описанных в этом разделе, поэтому просто выберите тот, что вам подходит.

После создания набора данных его, как правило, нужно аннотировать – добавить описательные метки для переменных. Раздел 2.4 посвящен аннотированию наборов данных, а раздел 2.5 – обзору некоторых полезных функций для работы с ними. Давайте начнем с самого начала.

2.1. Что такое набор данных?

Набор данных – это, как правило, прямоугольный массив данных, в котором строки соответствуют наблюдениям, а столбцы – признакам (или переменным). Для примера в табл. 2.1 представлен гипотетический набор данных о пациентах.

Таблица 2.1. Набор данных о пациентах

PatientID	AdmDate	Age	Diabetes	Status
1	10/15/2018	25	Type1	Poor
2	11/01/2018	34	Type2	Improved
3	10/21/2018	28	Type1	Excellent
4	10/28/2018	52	Type1	Poor

PatientID – порядковый номер пациента; AdmDate (admission date) – дата поступления: месяц/день/год; Age – возраст; Diabetes – тип диабета (Type1 – первый тип, Type2 – второй тип); Status – состояние (Poor – плохое; Improved – улучшившееся; Excellent – превосходное).

Представители разных профессий по-разному называют строки и столбцы в наборе данных. Статистики называют их *наблюдениями* (observation) и *переменными* (variable), аналитики, работающие с базами данных, говорят о *записях* (record) и *полях* (field), а работающие в области поиска скрытой информации в данных (data mining) и машинного обучения (machine learning) называют их *образцами* (example) и *свойствами* (attribute). На протяжении этой книги мы будем использовать термины *наблюдения* и *переменные*.

Нужно различать структуру набора данных (в данном случае – прямоугольный массив) и типы данных, которые его составляют. В наборе данных, приведенном в табл. 2.1, `PatientID` – это номер или идентификатор строки либо наблюдения, `AdmDate` – переменная, представляющая дату, `Age` – переменная, значения которой принадлежат непрерывному диапазону (или количественная переменная), `Diabetes` – номинальная (nominal; т. е. с ограничен-

ным кругом значений) переменная и `Status` – это порядковая (или шкальная – ordered) переменная. Оба вида переменных, номинальные и порядковые, относятся к категориальному типу, с той лишь разницей, что категории для порядковых переменных имеют естественное упорядочение.

Для хранения данных в R имеется множество самых разных структур данных, включая скаляры, векторы, массивы, таблицы и списки. Данные в табл. 2.1 соответствуют таблице в R. Такое большое разнообразие поддерживаемых структур дает языку R большую гибкость в работе с данными.

Язык R может манипулировать следующими типами данных: числовыми, текстовыми, логическими (TRUE/FALSE – истина/ложь), комплексными (мнимые числа) и неструктурированными (байты). Переменные `PatientID`, `AdmDate` и `Age` – это числовые переменные, а `Diabetes` и `Status` – текстовые. Дополнительно нужно сообщить программе, что `PatientID` – это идентификаторы наблюдений, `AdmDate` хранит даты, `Diabetes` и `Status` – это номинальная и порядковая переменные соответственно. Идентификаторы строк в R называются именами строк, а категориальные (номинальные и порядковые) переменные – факторами. В следующем разделе мы рассмотрим все это по порядку, а с датами познакомимся в главе 3.

2.2. Структуры данных

R работает с самыми разными структурами данных, включая скаляры, векторы, матрицы, массивы, таблицы и списки. Они отличаются типами данных, которые могут хранить, способом создания, сложностью устройства, а также способом обозначения и извлечения отдельных элементов. Эти структуры данных схематически изображены на рис. 2.1.

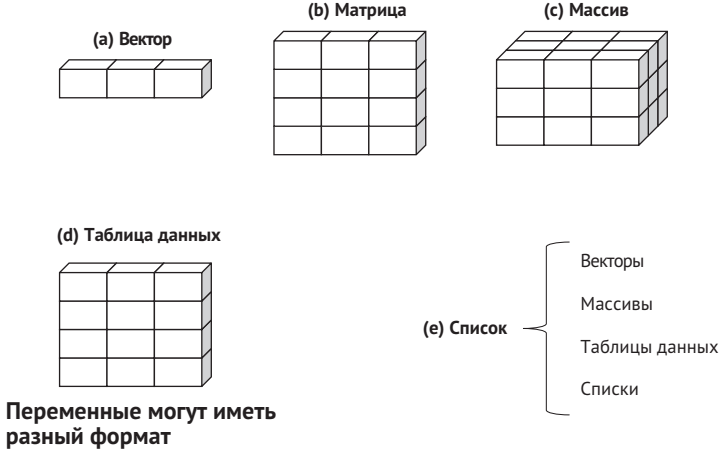


Рис. 2.1. Структуры данных в R

Некоторые определения

Существует несколько терминов, присущих только R, которые приводят в замешательство новых пользователей.

Объектом в R называется все, что может быть представлено в виде переменной, включая константы, разные типы данных, функции и даже диаграммы. У объектов есть вид (определяет, в каком виде объект хранится в памяти) и *класс* (с помощью которого общие функции, такие как `print`, определяют, как обращаться с тем или иным объектом).

Таблица данных (data frame) – это тип структуры данных в R, подобный наборам данных в обычных статистических программах (например, в SAS, SPSS и STATA). Столбцы – это переменные, а строки – это наблюдения. В одной таблице данных могут храниться переменные разных типов (например, числовые и текстовые). Таблицы данных – это основной тип структуры данных.

Факторы – это номинальные или порядковые переменные. В R они хранятся и обрабатываются особым образом. Вы узнаете больше о факторах в разделе 2.2.5.

Большинство остальных терминов должны быть вам знакомы, они широко используются в статистике и в вычислениях.

Теперь рассмотрим все виды структур данных по порядку, начав с векторов.

2.2.1. Векторы

Векторы – это одномерные массивы, которые могут хранить числовые, текстовые или логические значения. Создаются векторы с помощью функции объединения `c()`. Вот примеры векторов с данными некоторых типов:

```
a <- c(1, 2, 5, 3, 6, -2, 4)
b <- c("one", "two", "three")
c <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
```

Здесь `a` – числовой вектор, `b` – текстовый вектор, `c` – логический вектор. Обратите внимание, что все элементы вектора должны быть одного типа (числовые, текстовые или логические). В одном векторе нельзя смешивать данные разных типов.

ПРИМЕЧАНИЕ. *Скаляры* – это векторы, состоящие из одного элемента, например `f <- 3`, `g <- "US"` и `h <- TRUE`. Они используются для хранения констант.

К отдельным элементам вектора можно обращаться с помощью числового вектора с порядковыми номерами (позициями) элементов в квадратных скобках. Например, `a[c(2, 4)]` обозначает второй и четвертый элементы вектора `a`. Вот еще примеры:

```
> a <- c("k", "j", "h", "a", "c", "m")
> a[3]
[1] "h"
> a[c(1, 3, 5)]
[1] "k" "h" "c"
> a[2:6]
[1]  "j" "h" "a" "c" "m"
```

Оператор двоеточия (`:`) в последнем примере создает последовательности чисел. Например, `a <- c(2:6)` – это то же самое, что `a <- c(2, 3, 4, 5, 6)`.

2.2.2. Матрицы

Матрица – это двумерный массив данных, в котором все элементы имеют один и тот же тип (числовой, текстовый или логический). Матрицы создаются при помощи функции `matrix`. Вот общий синтаксис этой функции:

```
mymatrix <- matrix(вектор, nrow=число_строк, ncol=число_столбцов,
                   byrow=логическое_значение, dimnames=list(
                       текстовый_вектор_с_названиями_строк,
                       текстовый_вектор_с_названиями_столбцов))
```

где `вектор` – это вектор, содержащий элементы матрицы, `nrow` и `ncol` определяют число строк и столбцов в матрице, а список `dimnames` определяет названия строк и столбцов (их указывать не обязательно) в виде текстовых векторов. Параметр `byrow` определяет, как должна заполняться матрица – по строкам (`byrow=TRUE`) или по столбцам (`byrow=FALSE`). По умолчанию матрица заполняется по столбцам. Код в листинге 2.1 иллюстрирует применение функции `matrix`.

Листинг 2.1. Создание матриц

```
> y <- matrix(1:20, nrow=5, ncol=4)                          (1)
> y
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
> cells    <- c(1,26,24,68)
> rnames   <- c("R1", "R2")
> cnames   <- c("C1", "C2")
> mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE,
                     dimnames=list(rnames, cnames))          (2)

> mymatrix
   C1 C2
```

```
R1  1 26
R2 24 68
> mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=FALSE,
                     dimnames=list(rnames, cnames))      ❸
> mymatrix
   C1 C2
R1  1 24
R2 26 68
```

❶ Создаст матрицу 5×4
❷ Матрица 2×2 заполняется по строкам
❸ Матрица 2×2 заполняется по столбцам

Здесь сначала создается матрица 5×4 ❶. Затем создается матрица 2×2 с названиями строк и столбцов и заполняется по строкам ❷. Наконец, создается матрица 2×2 и заполняется по столбцам ❸.

Обращаться к строкам, столбцам и элементам матриц можно с помощью индексов и квадратных скобок. Например, `X[i,]` обозначает *i*-ю строку матрицы `X`, `X[,j]` – обозначает ее *j*-й столбец, а `X[i, j]` соответствует элементу, расположенному на пересечении *i*-й строки и *j*-го столбца. В качестве индексов *i* и *j* можно использовать числовые векторы для выбора сразу нескольких строк или столбцов, как показано в листинге 2.2.

Листинг 2.2. Использование индексов при работе с матрицами

```
> x <- matrix(1:10, nrow=2)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> x[2,]
  [1]  2  4  6  8 10
> x[,2]
[1] 3 4
> x[1,4]
[1] 7
```

Здесь сначала создается матрица 2×5, содержащая цифры от 1 до 10. По умолчанию матрица заполняется по столбцам. Затем выбираются все элементы во второй строке, а далее – все элементы во втором столбце. После этого выбирается элемент, находящийся в первой строке и в четвертом столбце. Наконец, выбираются элементы первой строки, находящиеся в четвертом и пятом столбцах.

Матрицы имеют два измерения и, подобно векторам, могут содержать данные только одного типа. Для хранения наборов данных с большим числом измерений следует использовать массивы (раздел 2.2.3). А для хранения вместе данных разных типов можно – таблицы (раздел 2.2.4).

2.2.3. Массивы

Массивы похожи на матрицы, но могут иметь больше двух измерений. Массивы создаются при помощи функции `array`:

```
myarray <- array(vector, dimensions, dimnames)
```

где *`vector`* – это вектор с данными, *`dimensions`* – числовой вектор, определяющий размеры измерений, а *`dimnames`* – необязательный список названий измерений. В листинге 2.3 показан пример создания трехмерного (2×3×4) числового массива.

Листинг 2.3. Создание массива данных

```
> dim1 <- c("A1", "A2")
> dim2 <- c("B1", "B2", "B3")
> dim3 <- c("C1", "C2", "C3", "C4")
> z <- array(1:24, c(2, 3, 4), dimnames=list(dim1, dim2, dim3))
> z
, , C1
   B1 B2 B3
A1  1  3  5
A2  2  4  6

, , C2
   B1 B2 B3
A1  7  9 11
A2  8 10 12

, , C3
   B1 B2 B3
A1 13 15 17
A2 14 16 18

, , C4
   B1 B2 B3
A1 19 21 23
A2 20 22 24
```

Очевидно, что массивы данных – это просто расширенные матрицы. Они могут пригодиться в программах для создания функций, выполняющих статистические вычисления. Все элементы массива, как и в матрицах, должны иметь одинаковый тип. Обращения к элементам массивов выполняются так же, как к элементам матриц. В примере выше элемент `z[1,2,3]` – это число 15.

2.2.4. Таблицы данных

Таблица данных (data frame) – это наиболее широко используемый по сравнению с матрицами объект, поскольку разные столбцы могут содержать данные разных типов (числовые, текстовые и т. д.). Таблицы данных схожи с наборами данных в SAS, SPSS и Stata и являются самыми часто используемыми структурами данных в R.

Набор данных о пациентах (табл. 2.1) состоит из числовых и текстовых данных. Поскольку эти данные имеют разные типы, для их хранения следует использовать таблицу данных, а не матрицу.

Таблица данных создается при помощи функции `data.frame()`:

```
mydata <- data.frame(col1, col2, col3, ...),
```

где – `col1`, `col2`, `col3`, `...` это векторы любого типа (текстового, числового или логического), которые станут столбцами таблицы. Названия каждому столбцу можно присвоить при помощи функции `names()`. Проиллюстрируем сказанное на примере (листинг 2.4).

Листинг 2.4. Создание таблицы данных

```
> patientID <- c(1, 2, 3, 4)
> age <- c(25, 34, 28, 52)
> diabetes <- c("Type1", "Type2", "Type1", "Type1")
> status <- c("Poor", "Improved", "Excellent", "Poor")
> patientdata <- data.frame(patientID, age, diabetes, status)
> patientdata
  patientID age diabetes    status
1         1  25    Type1      Poor
2         2  34    Type2  Improved
3         3  28    Type1 Excellent
4         4  52    Type1      Poor
```

Каждый столбец может содержать данные только одного типа, при этом столбцы в одной таблице данных могут быть разных типов. Таблицы данных очень близки к тому, что аналитики называют наборами данных, поэтому при обсуждении таблиц данных мы будем использовать термины *столбцы* и *переменные* как синонимы.

Обращаться к элементам таблиц можно несколькими способами. Можно использовать индексы (например, как при работе с матрицами) или указывать имена столбцов. Листинг 2.5 демонстрирует эти подходы на примере созданной выше таблицы данных `patientdata`.

Листинг 2.5. Обращение к элементам таблицы данных

```
> patientdata[1:2]
  patientID age
1         1  25
2         2  34
3         3  28
4         4  52
> patientdata[c("diabetes", "status")]
  diabetes    status
1    Type1      Poor
2    Type2  Improved
3    Type1 Excellent
4    Type1      Poor
```

```
 > patientdata$age          ❶
[1] 25 34 28 52
```

❶ **Ссылка на переменную** `age` **в наборе данных** `patientdata`

Знак `$` в третьем примере не встречался нам раньше. Он используется для ссылки на определенную переменную в таблице данных. Например, вот как можно создать сводную таблицу типов диабета в зависимости от состояния больного:

```
> table(patientdata$diabetes, patientdata$status)

        Excellent Improved Poor
  Type1         1        0    2
  Type2         0        1    0
```

Поскольку добавление префикса `patientdata$` перед именем каждой переменной может быстро надоесть, был предусмотрен сокращенный синтаксис для обращения к переменным. Примером может служить функция `with()`.

Функция with

Рассмотрим встроенную таблицу данных `mtcars`, содержащую данные о расходе топлива для 32 автомобилей. Следующий код:

```
> summary(mtcars$mpg)
  plot(mtcars$mpg, mtcars$disp)
  plot(mtcars$mpg, mtcars$wt)
```

выведет сводную переменную с расстоянием, которое может преодолеть автомобиль на одном галлоне топлива (`mpg`; в милях на галлон), в зависимости от объема двигателя и массы автомобиля. Этот код можно записать более кратко:

```
with(mtcars, {
  summary(mpg)
  plot(mpg, disp)
  plot(mpg, wt)
})
```

В этом случае инструкции в фигурных скобках будут автоматически ссылаться на таблицу данных `mtcars`. Если нужно выполнить только одну инструкцию (например, `summary(mpg)`), то фигурные скобки можно опустить.

Основное ограничение функции `with()` – она не действует за пределами фигурных скобок. Рассмотрим следующий пример:

```
> with(mtcars, {
   stats <- summary(mpg)
   stats
  })
   Min. 1st Qu. Median    Mean 3rd Qu.   Max.
  10.40   15.43   19.20   20.09   22.80   33.90
```

```
> stats
Error: object 'stats' not found
```

Чтобы создать объекты, которые будут существовать вне конструкции `with()`, используйте специальный оператор присваивания `<<-` вместо обычного (`<-`). Этот прием позволит сохранить созданный объект в рабочем пространстве вне конструкции `with()`. Например:

```
> with(mtcars, {
   nokeepstats <- summary(mpg)
   keepstats <<- summary(mpg)
})
> nokeepstats
Error: object 'nokeepstats' not found
> keepstats
   Min. 1st Qu. Median    Mean 3rd Qu.   Max.
   10.40   15.43   19.20   20.09   22.80   33.90
```

Названия строк

В примере с данными о пациентах столбец `patientID` использовался для идентификации отдельных пациентов в наборе данных. В R названия строк можно задать при помощи параметра `rowname` функции создания таблицы данных `data.frame()`. Например, инструкция

```
patientdata <- data.frame(patientID, age, diabetes,
                          status, row.names=patientID)
```

назначит `patientID` переменной, которая будет использоваться для идентификации строк при выводе данных и создании диаграмм.

2.2.5. Факторы

Как мы уже знаем, переменные бывают номинальными, порядковыми или непрерывными. Номинальные переменные – это категориальные данные, не имеющие определенного порядка. Переменная `Diabetes` – пример номинальных данных. Даже если обозначить значение `Type 1` единицей, а значение `Type 2` – двойкой, все равно их нельзя будет сравнивать в терминах «больше/меньше».

Порядковые данные можно упорядочить, но нельзя оценить количественно. Переменная `Status` – хороший пример порядковых данных. Понятно, что у больного с плохим (`poor`) самочувствием дела идут не так хорошо, как у больного, чье состояние улучшилось (`improved`), но не ясно, насколько. Непрерывные переменные могут принимать любое значение в пределах определенного диапазона. Их значения можно упорядочить и понять, насколько одно из них больше другого. Возраст, выраженный в годах, является непрерывной переменной и может принимать такие значения, как `14.5` или `22.8`, а также любые значения между ними. Мы знаем, что пятнадцатилетний подросток старше четырнадцатилетнего на один год.

Категориальные (номинальные и порядковые) данные называются в R *факторами*. Факторы играют важную роль в R, определяя, как данные будут анализироваться и изображаться на диаграммах. Факторы будут встречаться нам на протяжении всей книги.

Функция `factor()` сохраняет категориальные данные в виде вектора целых чисел в диапазоне [1...*k*] (где *k* – число уникальных значений категориальной переменной) и в виде внутреннего вектора из цепочки символов (исходных значений переменной), соответствующих этим целым числам.

К примеру, представьте, что у вас есть вектор

```
diabetes <- c("Type1", "Type2", "Type1", "Type1")
```

Инструкция `diabetes <- factor(diabetes)` преобразует этот вектор в `(1, 2, 1, 1)` и установит внутреннее соответствие `1=Type1` и `2=Type2` (в алфавитном порядке). Любой анализ с участием вектора `diabetes` будет воспринимать эту переменную как номинальную и выбирать статистические методы, подходящие для этого типа данных.

При работе с векторами, содержащими порядковые данные, в вызов функции `factor()` следует добавлять параметр `ordered=TRUE`. Для вектора

```
status <- c("Poor", "Improved", "Excellent", "Poor")
```

инструкция `status <- factor(status, ordered=TRUE)` преобразует этот вектор в вид `(3, 2, 1, 3)` и установит внутренние соответствия `1=Excellent`, `2=Improved`, `3=Poor`. Во время любой обработки этого вектора он будет воспринят как порядковая переменная и к нему будут применяться соответствующие статистические методы.

По умолчанию порядковые значения факторов для значений в векторе назначаются в алфавитном порядке. Мы видели это на примере фактора `status`, где порядок `"Excellent"`, `"Improved"`, `"Poor"` имеет смысл. Если бы вместо `"Poor"` использовалось значение `"Ailing"` (чахнущий), то возникло бы затруднение, потому что тогда порядок получился бы иным: `"Ailing"`, `"Excellent"`, `"Improved"`. Сходная проблема возникла бы, если бы нам понадобился обратный порядок: `"Poor"`, `"Improved"`, `"Excellent"`. Для упорядоченных факторов редко подходит алфавитный порядок назначения числовых значений, предлагающийся по умолчанию.

Порядок по умолчанию можно изменить при помощи параметра `levels`. Например:

```
status <- factor(status, order=TRUE,
                 levels=c("Poor", "Improved", "Excellent"))
```

установит следующие ассоциации: `1=Poor`, `2=Improved`, `3=Excellent`.

Проверьте сами, соответствуют ли присвоенные уровни реальным значениям данных. Все значения, которые не были указаны, будут обозначены как отсутствующие.

Приведенный ниже программный код показывает, как назначение факторов и упорядоченных факторов влияет на анализ данных.

Числовые переменные тоже можно определить как факторы, используя параметры `levels` и `labels`. Если, к примеру, в исходных данных мужской пол представлен 1, а женский – 2, то

```
sex <- factor(sex, levels=c(1, 2), labels=c("Male", "Female"))
```

преобразует переменную в неупорядоченный фактор. Обратите внимание, что порядок меток должен соответствовать порядку уровней. В этом примере пол будет рассматриваться как категориальное значение и в выводе вместо 1 и 2 появятся метки `"Male"` (мужской) и `"Female"` (женский), а любое значение пола, кроме 1 и 2, будет определено как отсутствующее и отобразится как `missing`.

Листинг 2.6 демонстрирует, как определение упорядоченных и неупорядоченных факторов влияет на анализ данных.

Листинг 2.6. Использование факторов

```
> patientID <- c(1, 2, 3, 4)                                          ❶
> age <- c(25, 34, 28, 52)
> diabetes <- c("Type1", "Type2", "Type1", "Type1")
> status <- c("Poor", "Improved", "Excellent", "Poor")
> diabetes <- factor(diabetes)
> status <- factor(status, order=TRUE)
> patientdata <- data.frame(patientID, age, diabetes, status)
> str(patientdata)                                                    ❷
'data.frame':   4 obs. of  4 variables:
$ patientID: num  1 2 3 4
 $ age      : num  25 34 28 52
 $ diabetes : Factor w/ 2 levels "Type1","Type2": 1 2 1 1
 $ status   : Ord.factor w/ 3 levels "Excellent"<"Improved"<..: 3 2 1 3
> summary(patientdata)                                                ❸
   patientID          age         diabetes       status
 Min.   :1.00   Min.   :25.00   Type1:3      Excellent:1
 1st Qu.:1.75   1st Qu.:27.25   Type2:1      Improved :1
 Median :2.50   Median :31.00                Poor     :2
 Mean   :2.50   Mean   :34.75
 3rd Qu.:3.25   3rd Qu.:38.50
 Max.   :4.00   Max.   :52.00
```

❶ Ввод данных в виде векторов.

❷ Вывод структуры объектов.

❸ Вывод сводной статистики объекта.

Здесь сначала вводятся исходные данные в виде векторов ❶. Затем `diabetes` объявляется фактором, а `status` – упорядоченным фактором. Наконец, данные объединяются в таблицу. Функция

`str(object)` выводит информацию об объекте (в нашем случае это таблица данных) (2). Ясно видно, что `diabetes` – это фактор, а `status` – упорядоченный фактор; также сообщается его внутреннее представление. Обратите внимание, что функция `summary()` обрабатывает переменные по-разному (3). Для непрерывной переменной `age` она вычислила минимум, максимум, среднее и квартили, а для категориальных переменных `diabetes` и `status` – частоту встречаемости каждого значения.

2.2.6. Списки

Списки – самый сложный тип данных в R. Фактически список – это упорядоченная коллекция объектов (компонентов). Список может объединять разные (возможно, не связанные между собой) объекты. К примеру, список может одновременно содержать векторы, матрицы, таблицы данных и даже другие списки. Создаются списки при помощи функции `list()`:

```
mylist <- list(объект1, объект2, ...)
```

где объекты – это любые структуры данных, упоминавшиеся до этого. Объектам в списке можно присваивать имена:

```
mylist <- list(имя1=объект1, имя2=объект2, ...)
```

В листинге 2.7 показан пример работы со списками.

Листинг 2.7. Создание списка

```
> g <- "My First List"
> h <- c(25, 26, 18, 39)
> j <- matrix(1:10, nrow=5)
> k <- c("one", "two", "three")
> mylist <- list(title=g, ages=h, j, k)        (1)
> mylist                                       (2)
$title
[1] "My First List"

$ages
[1] 25 26 18 39

[[3]]
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10

[[4]]
[1] "one"   "two"   "three"
```

```
> mylist[[2]]                                    3
[1] 25 26 18 39
> mylist[["ages"]]
[[1] 25 26 18 39
```

1. **Создание списка.**
2. **Вывод содержимого списка.**
3. **Вывод второго компонента из списка.**

Этот пример создает список из четырех компонентов: текстовой строки, числового вектора, матрицы и текстового вектора. В списке можно сохранить любое число объектов.

Обратиться к элементу списка можно по его порядковому номеру или по имени внутри двойных квадратных скобок. В этом примере `mylist[[2]]` и `mylist[["ages"]]` соответствуют одному и тому же числовому вектору с четырьмя элементами. Списки – это важный тип структур данных в R. Во-первых, они позволяют без труда упорядочить и вывести разрозненную информацию, а во-вторых, многие функции возвращают результаты в виде списков (в таких случаях аналитик должен сам извлечь из списков нужную ему информацию). У вас еще будет возможность увидеть многочисленные примеры функций, возвращающих списки.

2.2.7. Усовершенствованные таблицы данных

Прежде чем двигаться дальше, стоит упомянуть *усовершенствованные таблицы данных* – tibble, – которые, по сути, являются теми же таблицами данных, но обладают дополнительными возможностями, делающими их более полезными. Они создаются с помощью функции `tibble()` или `as_tibble()` из пакета `tibble`. Установить пакет `tibble` можно командой `install.packages("tibble")`. Ниже перечислены некоторые из привлекательных особенностей усовершенствованных таблиц.

Усовершенствованные таблицы выводятся в более компактном формате, а метки переменных включают описание их типов:

```
library(tibble)
mtcars <- as_tibble(mtcars)
mtcars

# Усовершенствованная таблица: 32×11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21       6  160    110  3.9   2.62  16.5     0     1     4     4
 2  21       6  160    110  3.9   2.88  17.0     0     1     4     4
 3  22.8     4  108     93  3.85  2.32  18.6     1     1     4     1
 4  21.4     6  258    110  3.08  3.22  19.4     1     0     3     1
 5  18.7     8  360    175  3.15  3.44  17.0     0     0     3     2
 6  18.1     6  225    105  2.76  3.46  20.2     1     0     3     1
 7  14.3     8  360    245  3.21  3.57  15.8     0     0     3     4
```

```
 8  24.4     4  147.   62  3.69  3.19  20       1     0     4     2
 9  22.8     4  141.   95  3.92  3.15  22.9     1     0     4     2
10  19.2     6  168.  123  3.92  3.44  18.3     1     0     4     4
# ... и еще 22 строки
```

Усовершенствованные таблицы никогда не преобразовывают символьные переменные в факторы. В более старых версиях R (до R 4.0) такие функции, как `read.table()`, `data.frame()` и `as.data.frame()`, по умолчанию преобразуют символьные данные в факторы, и чтобы подавить такое их поведение, необходимо передавать таким функциям параметр `stringsAsFactors = FALSE`.

Усовершенствованные таблицы никогда не меняют имена переменных. Если в импортируемом наборе данных есть переменная с именем `Last Address`, базовые функции R преобразуют это имя в `Last.Address`, потому что в именах переменных R не допускаются пробелы. Усовершенствованные таблицы сохранят имя как есть и будут использовать обратные кавычки (например, `` `Last Address` ``), чтобы сделать имя переменной синтаксически правильным.

При выборке подмножества из усовершенствованной таблицы всегда возвращается усовершенствованная таблица. Например, выборка подмножества из (обычной) таблицы данных `mtcars` обращением `mtcars[,"mpg"]` вернет вектор, а не таблицу данных с одним столбцом.

R автоматически упрощает результаты. Чтобы получить таблицу данных с одним столбцом, нужно добавить параметр `drop = FALSE` (`mtcars[, "mpg", drop = FALSE]`). Если `mtcars` является усовершенствованной таблицей, то `mtcars[, "mpg"]` вернет усовершенствованную таблицу с одним столбцом. Результаты не упрощаются, что позволяет предсказать, какой результат вернет операция выборки подмножества.

Наконец, усовершенствованные таблицы не поддерживают имена строк. К обычным таблицам данных можно применить функцию `rownames_to_column()`, чтобы преобразовать имена строк в переменную в таблице.

Усовершенствованные таблицы играют важную роль, потому что многие популярные пакеты, такие как `readr`, `tidyr`, `dplyr` и `purr`, сохраняют наборы данных в виде усовершенствованных таблиц. Усовершенствованные таблицы были разработаны как «современный взгляд на наборы данных», но обратите внимание, что их можно использовать взаимозаменяемо с обычными таблицами данных. Любая функция, принимающая обычную таблицу данных, будет принимать и усовершенствованную таблицу, и наоборот. За дополнительной информацией обращайтесь по адресу: https://r4ds.had.co.nz/tibbles.html.

Информация для программистов

Для многих профессиональных программистов некоторые аспекты языка R кажутся необычными. Вот некоторые из особенностей, о которых следует помнить:

- *точка (`.`) в названиях объектов не имеет никакого специального значения. Однако знак доллара (`$`) имеет примерно такое же значение, как точка в других объектно-ориентированных языках программирования, и может использоваться для обозначения частей таблицы данных или списка. Например,* `A$x` *обозначает переменную* `x` *в таблице данных* `A`*;*
- *в R нет возможности писать многострочные или блочные комментарии. Каждую строку комментария нужно начинать со знака #. При отладке программ для предотвращения выполнения больших блоков кода их можно заключать в конструкцию* `if(FALSE){...}`*. Замена* `FALSE` *на* `TRUE` *сделает возможным выполнение кода;*
- *присваивание значения несуществующему элементу вектора, матрицы, массива данных или списка добавит соответствующий элемент в объект. Например:*

```
> x <- c(8, 6, 4)
> x[7] <- 10
> x
[1]  8  6  4 NA NA NA 10
```

В результате присваивания размер вектора `x` увеличился с трех до семи элементов;

- *в R нет обычных скаляров. Скаляры представляются в виде векторов с одним элементом;*
- *нумерация в R начинается с* `1`*, а не с* `0`*. В приведенном выше векторе элемент* `x[1]` *– это число* `8`*.*

Дополнительную информацию можно найти в превосходном блоге Джона Кука (John Cook) «R Language for Programmers» (http://mng.bz/6NwQ). Программисты, ищущие руководство по стилю программирования, могут заглянуть в руководство «The Tidyverse Style Guide» Хэдли Уикхема (Hadley Wickham) (https://style.tidyverse.org/).

2.3. Ввод данных

Имея структуры данных, их нужно наполнить данными! Обычно аналитикам приходится извлекать данные из разных источников, где они хранятся в разных форматах. Задача состоит в том, чтобы импортировать данные в программу, проанализировать их и представить отчет с результатами. R предоставляет широкий выбор инструментов для импорта данных. Исчерпывающее руководство по импорту данных в R «R Data Import/Export» можно найти по адресу: http://mng.bz/urwn.

Как показано на рис. 2.2, R позволяет вводить данные с клавиатуры, импортировать их из текстовых файлов, из таблиц Microsoft Excel и Access, из распространенных статистических программ, из разнообразных баз данных, из веб-сайтов и онлайн-служб. Поскольку никогда нельзя угадать, откуда вы будете получать данные, я расскажу обо всех поддерживаемых возможностях по порядку, а вам останется только выбрать и прочитать то, что вам нужно.

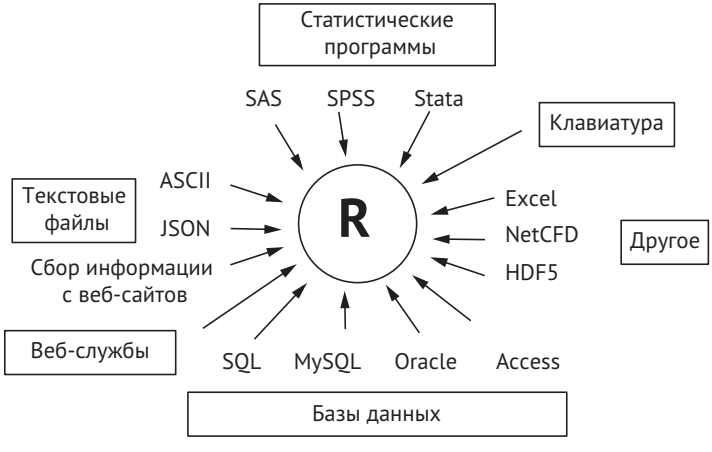


Рис. 2.2. Источники, откуда можно импортировать данные в R

2.3.1. Ввод данных с клавиатуры

Наверное, самый простой способ ввода данных – это ввод с клавиатуры: посредством текстового редактора, встроенного в R, или внедрением данных непосредственно в код. Сначала рассмотрим встроенный текстовый редактор.

Функция `edit()` откроет текстовый редактор, куда можно внести данные. Вот пошаговая инструкция:

1. Создайте пустую таблицу данных (или матрицу), указав названия и типы переменных.
2. Откройте текстовый редактор с этим объектом, введите данные и сохраните результат в виде объекта с данными.

В следующем примере показано, как создать таблицу данных с названием `mydata` и с тремя переменными: `age` (возраст, числовая), `gender` (пол, текстовая) и `weight` (вес, числовая). Затем открывается текстовый редактор, в него вводятся данные, и результат сохраняется.

```
mydata <- data.frame(age=numeric(0), gender=character(0),
                     weight=numeric(0))
mydata <- edit(mydata)
```

Операции присваивания, такие как `age=numeric(0)`, создают пустую (без данных) переменную заданного типа. Обратите внимание, что результат редактирования вновь присваивается исход-

ному объекту. Функция `edit()` работает с копией объекта. Если не присвоить результат ее работы какому-либо объекту, все изменения пропадут!

На рис. 2.3 показан результат вызова функции `edit()` в Windows. На рисунке видно, что я добавил некоторые данные. Если щелкнуть на заголовке столбца, то редактор даст возможность изменить название и тип соответствующей переменной. Щелкая на заголовках неиспользуемых столбцов, можно добавлять дополнительные переменные. После закрытия текстового редактора результаты сохраняются в выбранном объекте (в данном случае `mydata`). Повторно вызвав инструкцию `mydata <- edit(mydata)`, можно отредактировать уже введенные данные и добавить новые. Инструкцию `mydata <- edit(mydata)` можно заменить более краткой версией: `fix(mydata)`.

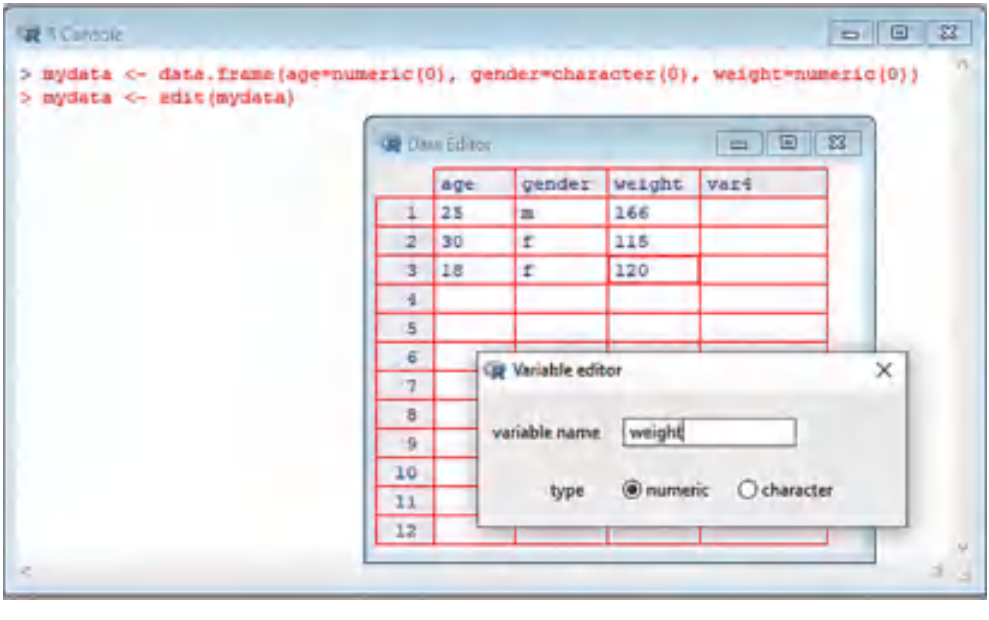


Рис. 2.3. Редактирование данных при помощи встроенного текстового редактора в Windows

Как вариант данные можно включить непосредственно в программный код. Например, следующий код создаст таблицу `samedata`, подобную той, что была создана с помощью функции `edit()`:

```
mydatatxt <- "
age gender weight
25 m 166
30 f 115
18 f 120
"
mydata <- read.table(header=TRUE, text=mydatatxt)
```

Символьная строка содержит исходные данные, а функция `read.table()` обрабатывает эту строку и возвращает таблицу данных. Более подробно функция `read.table()` будет описана в следующем разделе.

Метод ввода данных с клавиатуры хорошо подходит для случаев, когда объемы данных невелики. Для ввода больших наборов данных предпочтительнее использовать описанные ниже методы: импорт данных из существующих текстовых файлов, электронных таблиц Excel, статистических программ или систем управления базами данных.

2.3.2. Импорт данных из текстового файла с разделителями

Импорт данных из текстовых файлов с разделителями можно осуществить с помощью функции `read.table()`, которая сохраняет данные в виде таблицы. Например:

```
mydataframe <- read.table(file, options)
```

где *file* – это текстовый файл ASCII (American Standard Code for Information Interchange – американский стандартный код обмена информацией) с разделителями, а *options* – список параметров, управляющих обработкой данных. Наиболее часто используемые параметры перечислены в табл. 2.2.

Таблица 2.2. Параметры функции read.table()

Параметр	Описание
`header`	Логическое значение, указывающее, содержит ли файл имена переменных в первой строке
`sep`	Определяет разделитель, отделяющий значения друг от друга. По умолчанию `sep=" "`, т. е. значения данных разделены одним или несколькими пробелами, табуляциями или разрывами строк. Используйте параметр `sep=","` для импорта данных из файлов CSV и `sep="\t"` – из файлов TSV
`row.names`	Необязательный параметр, определяющий одну или несколько переменных, представляющих идентификаторы строк
`col.names`	Если первая строка в файле не содержит имен переменных (`header=FALSE`), то можно передать параметр `col.names`, определяющий текстовый вектор с именами переменных. Если `header=FALSE` и параметр `col.names` не задан, то переменные получат имена `V1`, `V2` и т. д.
`na.strings`	Необязательный текстовый вектор, определяющий значения, которые должны интерпретироваться как отсутствующие значения. Например, `na.strings=c("-9", "?")` преобразует все значения `-9` и `?` в `NA`
`colClasses`	Необязательный вектор классов, назначаемых столбцам. Например, если передать параметр `colClasses=c("numeric", "numeric", "character", "NULL", "numeric")`, то функция `read.table()` прочитает первые два столбца как числовые, третий столбец как текстовый, пропустит четвертый столбец и прочитает пятый столбец как числовой. Если в данных больше пяти столбцов, то параметр `colClasses` будет использован повторно. При импорте больших текстовых файлов включение параметра `colClasses` может значительно ускорить обработку

Окончание табл. 2.2

Параметр	Описание
`quote`	Символ(ы), используемый(е) для ограничения строк, содержащих специальные символы. Обычно в этом параметре передается двойная (") или одинарная (') кавычка
`skip`	Количество строк в файле, которые нужно пропустить перед началом чтения. Этот параметр можно использовать для пропуска начальных комментариев в файле
`stringAsFactors`	Логическое значение, указывающее на необходимость преобразования текстовых переменных в факторы. До R 4.0 этот параметр принимал значение по умолчанию `TRUE`. В более свежих версиях используется значение по умолчанию `FALSE`, если оно не переопределено в параметре `colClasses`. При импорте больших текстовых файлов включение параметра `stringsAsFactors=FALSE` может значительно ускорить обработку
`text`	Текстовая строка для импорта. Если передан параметр `text`, то параметр `file` можно опустить

Рассмотрим пример импорта текстового файла с именем *studentgrades.csv*, содержащего оценки учащихся по математике (`math`), естествознанию (`Science`) и обществоведению (`Social Studies`). Каждая строка в файле представляет определенного учащегося. Первая строка содержит имена переменных, разделенные запятыми. Каждая последующая строка – информацию о студенте, в которой значения также разделены запятыми. Вот как выглядят первые несколько строк:

```
StudentID,First,Last,Math,Science,Social Studies
011,Bob,Smith,90,80,67
012,Jane,Weary,75,,80
010,Dan,"Thornton, III",65,75,70
040,Mary,"O'Leary",90,95,92
```

Импортировать этот файл можно следующей инструкцией:

```
grades <- read.table("studentgrades.csv", header=TRUE,
    row.names="StudentID", sep=",")
```

и в результате получить:

```
> grades

   First             Last Math Science Social.Studies
11   Bob            Smith   90      80             67
12  Jane            Weary   75      NA             80
10   Dan    Thornton, III   65      75             70
40  Mary          O'Leary   90      95             92

> str(grades)

'data.frame':   4 obs. of  5 variables:
 $ First         : chr  "Bob" "Jane" "Dan" "Mary"
 $ Last          : chr  "Smith" "Weary" "Thornton, III" "O'Leary"
 $ Math          : int  90 75 65 90
 $ Science       : int  80 NA 75 95
 $ Social.Studies: int  67 80 70 92
```

Отметим несколько интересных моментов, касающихся импорта этих данных. Имя переменной `Social Studies` автоматически преобразуется в соответствии с соглашениями R. Столбец `StudentID` теперь считается содержащим идентификаторы строк – он лишился своей метки и ведущего нуля. Отсутствующая оценка по естествознанию для Jane правильно обозначена как отсутствующая (`NA`). Мне пришлось заключить фамилию «Thornton, III» в кавычки, чтобы избежать неверной интерпретации запятой между Thornton и III. В противном случае R увидел бы в этой строке семь значений, а не шесть. Мне также пришлось заключить в кавычки имя O'Leary. В противном случае R воспринял бы одинарную кавычку как разделитель строк, а это не то, что мне нужно.

Параметр stringsAsFactors

В функциях `read.table()`, `data.frame()` и `as.data.frame()` параметр `stringsAsFactors` определяет, будут ли символьные переменные автоматически преобразовываться в факторы. До версии R 4.0.0 этот параметр принимал значение по умолчанию `TRUE`. Начиная с R 4.0.0 по умолчанию используется значение `FALSE`. Если вы пользуетесь более старой версией R, то переменные `First` и `Last` в предыдущем примере будут преобразованы в факторы.

Преобразование символьных переменных в факторы не всегда желательно. Например, нет причин преобразовывать в фактор символьную переменную, содержащую комментарий респондента. Кроме того, иногда бывает желательно извлекать текст из переменной, а это трудно сделать после ее преобразования в фактор.

Подавить это поведение несколькими способами. Добавить параметр `stringsAsFactors=FALSE`, чтобы отключить поведение по умолчанию для всех символьных переменных, или использовать параметр `colClasses`, чтобы явно указать класс (например, логический, числовой, символьный или факторный) для каждого столбца.

Давайте импортируем те же данные, указав класс для каждой переменной:

```
grades <- read.table("studentgrades.csv", header=TRUE,
            row.names="StudentID", sep=",",
            colClasses=c("character", "character", "character",
                         "numeric", "numeric", "numeric"))
> grades

    First             Last Math Science Social.Studies
011   Bob            Smith   90      80             67
012  Jane            Weary   75      NA             80
010   Dan    Thornton, III   65      75             70
040  Mary          O'Leary   90      95             92
```

```
> str(grades)

'data.frame':   4 obs. of  5 variables:
 $ First         : chr  "Bob" "Jane" "Dan" "Mary"
 $ Last          : chr  "Smith" "Weary" "Thornton, III" "O'Leary"
 $ Math          : num  90 75 65 90
 $ Science       : num  80 NA 75 95
 $ Social.Studies: num  67 80 70 92
```

Теперь имена строк сохранили ведущие нули, а переменные `First` и `Last` не были преобразованы в факторы (даже в более ранних версиях R). Кроме того, оценки хранятся в виде действительных значений, а не целых чисел.

Функция `read.table()` поддерживает множество параметров для точной настройки импорта данных. Подробности ищите в `help(read.table)`.

Импорт данных через соединения

Во многих примерах в этой главе демонстрируется импорт данных из файлов, хранящихся на локальном компьютере. Однако R предоставляет также несколько механизмов доступа к данным через соединения. Например, вместо имени файла можно использовать функции `file()`, `gzfile()`, `bzfile()`, `xzfile()`, `unz()` и `url()`. Функция `file()` позволяет получить доступ к файлам, буферу обмена и стандартному вводу. Функции `gzfile()`, `bzfile()`, `xzfile()` и `unz()` позволяют читать сжатые файлы.

Функция `url()` позволяет получить доступ к файлам в интернете через полный URL, включающий схему `http://`, `ftp://` или `file://`. Для HTTP и FTP можно указать прокси-сервер. Для удобства вместо имен файлов можно также использовать полные URL (заключенные в двойные кавычки). Подробности ищите в `help(file)`.

В R также имеются функции `read.csv()` и `read.delim()` для импорта прямоугольных текстовых файлов. Это просто функции-обертки, вызывающие `read.table()` с определенными значениями в параметрах. Например, `read.csv()` вызывает `read.table()` с параметрами `header =TRUE` и `sep=","`, а `read.delim()` вызывает `read.table()` с параметрами `header=TRUE` и `sep="\t"`. Подробности ищите в справке для функции `read.table()`.

Пакет `readr` предоставляет мощную альтернативу базовым функциям R для чтения прямоугольных текстовых файлов: основную функцию `read_delim()` и вспомогательные функции `read_csv()` и `read_tsv()`, предназначенные для чтения текстовых файлов с разделителями-запятыми и разделителями-табуляцией соответственно. После установки пакета предыдущие данные можно импортировать так:

```
library(readr)
grades <- read_csv("studentgrades.csv")
```

Пакет также позволяет импортировать файлы фиксированной ширины (где данные находятся в определенных позициях), табличные файлы (где столбцы разделены пробелами) и файлы веб-журналов.

Функции в пакете `readr` имеют ряд преимуществ перед базовыми функциями. Прежде всего они значительно быстрее. Это может стать огромным преимуществом при чтении больших файлов данных. Кроме того, они очень хорошо определяют тип данных каждого столбца (числовой, символьный, дата и дата/время). Наконец, в отличие от базовых функций R (до версии R 4.0.0), они по умолчанию не преобразуют символьные данные в факторы. Функции в пакете `readr` возвращают данные в виде усовершенствованных таблиц `tibble`. Узнать больше о пакете можно по адресу: https://readr.tidyverse.org.

2.3.3. Импорт данных из Excel

Лучший способ импортировать файл в формате Excel – предварительно сохранить его в формате текстового файла с разделителями и только потом импортировать, как описано выше. Также файлы Excel можно импортировать с помощью пакета `readxl`. Не забудьте скачать и установить его, прежде чем пытаться использовать.

Пакет `readxl` можно использовать для чтения файлов Excel в форматах `.xls` и `.xlsx`. Функция `read_excel()` импортирует рабочий лист в таблицу данных в виде усовершенствованной таблицы tibble. Самый простой вариант вызова – `read_excel(file, n)`, где `file` определяет путь к книге Excel, а `n` – номер импортируемой таблицы, в которой первая строка содержит имена переменных. Например, в Windows следующий код:

```
library(readxl)
workbook <- "c:/myworkbook.xlsx"
mydataframe <- read_xlsx(workbook, 1)
```

импортирует первый лист из книги *myworkbook.xlsx*, хранящейся на диске *C:*, и сохранит его в виде таблицы данных `mydataframe`.

Функция `read_excel()` принимает параметры, позволяющие задать определенный диапазон ячеек (например, `range = "Mysheet!B2:G14"`) вместе с классом каждого столбца (`col_types`). Подробности ищите в `help(read_excel)`.

В числе других пакетов, которые могут помочь в работе с файлами Excel, можно назвать: `xlsx`, `XLConnect` и `openxlsx`. Пакеты `xlsx` и `XLConnect` требуют наличия установленной среды выполнения Java на компьютере, а `openxlsx` – нет. В отличие от `readxl`, эти пакеты могут не только импортировать листы, но также создавать файлы Excel и управлять ими. Программистам, разрабатывающим интерфейс между R и Excel, следует внимательно изучить один или несколько из этих пакетов.

2.3.4. Импорт данных из JSON-файлов

В последнее время все чаще данные распространяются в виде файлов в формате JSON (JavaScript Object Notation – форма записи объектов JavaScript). В R есть несколько пакетов для работы с такими файлами. Например, пакет `jsonlite` позволяет читать, писать и преобразовывать объекты JSON. Данные в формате JSON можно импортировать непосредственно в таблицы данных R. Обсуждение этого формата выходит за рамки нашей книги. Заинтересованным в работе с JSON-файлами я рекомендую обратиться к прекрасной справочной документации с описанием пакета `jsonlite` на сайте https://cran.r-project.org/web/packages/jsonlite/.

2.3.5. Извлечение данных из веб-страниц

Данные из интернета можно получить методом веб-скраппинга (web-scrapping) или с использованием прикладных программных интерфейсов (Application Programming Interfaces, API). Веб-скраппинг обычно используется для извлечения информации, находящейся внутри веб-страниц, тогда как API позволяют взаимодействовать с веб-службами и удаленными хранилищами данных.

При извлечении данных из интернета методом веб-скраппинга пользователи обычно сохраняют их в виде объектов R для дальнейшего анализа. Например, при помощи функции `readLines()` можно извлечь текст из веб-страницы и дальше работать с ним, используя функции `grep()` и `gsub()`.

Пакет `rvest` предлагает функции, упрощающие извлечение данных из веб-страниц. Он был создан по образу и подобию библиотеки Beautiful Soup для Python. Для извлечения информации из сложно устроенных веб-страниц можно также использовать пакеты `RCurl` и `XML`. Более полную информацию, включая примеры, можно найти в руководстве «Examples of Web Scraping with R» на сайте *ProgrammingR* (http://www.programmingr.com/).

Прикладные интерфейсы API определяют порядок взаимодействий программных компонентов друг с другом. В R имеется несколько пакетов, использующих такой подход для извлечения данных из веб-ресурсов, служащих источниками данных по биологии, медицине, науках о Земле, физических науках, экономике, финансах, литературе, маркетингу и т. д.

Например, интересующиеся социальными сетями могут получить доступ к данным в Twitter через `twitteR`, в Facebook через `Rfacebook` и в Flickr через `Rflickr`. Также есть пакеты для доступа к популярным веб-службам Google, Amazon, Dropbox, Salesforce и др. Полный список пакетов для доступа к веб-ресурсам вы найдете в статье «CRAN Task View: Web Technologies and Services» (http://mng.bz/370r).

2.3.6. Импорт данных из SPSS

Наборы данных в формате IBM SPSS можно импортировать в R при помощи функции `read_spss()` из пакета `haven`. Сначала нужно скачать и установить пакет:

```
install.packages("haven")
```

и затем выполнить следующий код:

```
library(haven)
mydataframe <- read_spss("mydata.sav")
```

Набор данных будет импортирован в усовершенствованную таблицу данных (tibble), а переменным, содержащим метки импортированных значений SPSS, будет назначен класс `labelled`. Эти переменные класса `labelled` можно преобразовать в факторы, используя следующий код:

```
labelled_vars <- names(mydataframe)[sapply(mydataframe, is.labelled)]
for (vars in labelled_vars){
  mydataframe[[vars]] = as_factor(mydataframe[[vars]])
}
```

Пакет `haven` имеет ряд дополнительных функций для чтения файлов SPSS в сжатом (`.zsav`) или транспортном (`.por`) формате.

2.3.7. Импорт данных из SAS

Наборы данных SAS можно импортировать с помощью функции `read_sas()` из пакета `haven`. После установки пакета данные можно импортировать, как показано ниже:

```
library(haven)
mydataframe <- read_sas("mydata.sas7bdat")
```

Если у вас есть и каталог форматов переменных, их также можно импортировать и применять к данным, например:

```
mydataframe <- read_sas("mydata.sas7bdat",
                        catalog_file = "mydata.sas7bcat")
```

В любом случае результатом будет усовершенствованная таблица данных tibble.

Имеется также коммерческий продукт Stat/Transfer (описанный в разделе 2.3.10), отлично справляющийся с импортом наборов данных из SAS (включая любые существующие форматы переменных) в виде таблиц данных R.

2.3.8. Импорт данных из Stata

Импортировать данные из Stata в R очень просто:

```
library(haven)
mydataframe <- read_dta("mydata.dta")
```

Здесь `mydata.dta` – это набор данных Stata, а `mydataframe` – это итоговая таблица данных R в формате усовершенствованной таблицы tibble.

2.3.9. Импорт данных из баз данных

R может взаимодействовать с самыми разными базами данных, включая Microsoft SQL Server, Microsoft Access, MySQL, Oracle, PostgreSQL, DB2, Sybase, Teradata и SQLite. Некоторые пакеты предоставляют доступ через оригинальные драйверы баз данных, тогда как другие обеспечивают доступ к данным через ODBC (Open Database Connectivity interface – открытый интерфейс взаимодействия с базами данных) или JDBC (Java Database Connectivity interface – Java-интерфейс взаимодействия с базами данных). Поддержка в R доступа к данным, хранящимся во внешних базах данных, позволяет эффективно обрабатывать большие наборы данных (см. приложение F), используя мощные возможности обоих языков, SQL и R.

Интерфейс ODBC

Самый, пожалуй, популярный способ доступа к базам данных в R – с использованием пакета `RODBC`, позволяющий подключиться к любой базе данных, для которой имеется драйвер ODBC. К таким базам данных относятся все перечисленные выше.

Первый шаг – установка и настройка драйвера ODBC для конкретной операционной системы и базы данных, поскольку они не являются частью R. Если нужные драйверы у вас еще не установлены, их можно найти в интернете (хорошей отправной точкой может стать статья «Setting up ODBC Drivers», доступная по адресу: https://db.rstudio.com/best-practices/drivers/).

После установки и настройки драйверов установите пакет `RODBC`, выполнив инструкцию `install.packages("RODBC")`.

Основные функции, предлагаемые этим пакетом, перечислены в табл. 2.3.

Таблица 2.2. Функции пакета RODBC

Функция	Описание
`odbcConnect(`*dsn*`, uid="", pwd="")`	Открывает соединение с базой данных ODBC
`sqlFetch(`*channel*`, `*sqltable*`)`	Читает таблицу из базы данных ODBC в таблицу данных в R
`sqlQuery(`*channel*`, `*query*`)`	Посылает запрос в базу данных ODBC и возвращает результаты
`sqlSave(`*channel*`, `*mydf*`, tablename = `*sqltable*`, append=FALSE)`	Сохраняет или обновляет (`append=TRUE`) таблицу данных в виде таблицы в базе данных ODBC
`sqlDrop(`*channel*`, `*sqltable*`)`	Удаляет таблицу из базы данных ODBC
`close(`*channel*`)`	Закрывает соединение

Пакет RODBC устанавливает двустороннее соединение между R и базой данных SQL, что позволяет не только загружать данные в R, но и изменять содержимое базы данных из программы на R. Представьте, что вам нужно импортировать две таблицы (Crime – преступление и Punishment – наказание) из БД в таблицы данных R с именами crimedat и pundat соответственно. Это можно сделать так:

```
library(RODBC)
myconn <-odbcConnect("mydsn", uid="Rob", pwd="aardvark")
crimedat <- sqlFetch(myconn, Crime)
pundat <- sqlQuery(myconn, "select * from Punishment")
close(myconn)
```

Этот код загружает пакет RODBC и открывает соединение с базой данных ODBC через зарегистрированное имя источника данных (mydsn) с идентификатором пользователя UID (rob) и паролем (aardvark). Объект соединения передается в функцию sqlFetch, которая импортирует содержимое таблицы Crime в таблицу данных crimedat. Затем выполняется SQL-запрос select, извлекающий данные из таблицы Punishment, и эти данные сохраняются в таблице данных pundat. В заключение соединение закрывается.

Функция sqlQuery() открывает массу возможностей, позволяя выполнить любой действительный SQL-запрос. Эта гибкость дает возможность извлекать отдельные переменные и подмножества данных, создавать новые переменные, а также перекодировать и переименовывать существующие переменные.

Пакеты, связанные с DBI

Пакет DBI – универсальный и последовательный пользовательский интерфейс для доступа к базам данных. Созданный на его основе фреймворк RJDBC позволяет получить доступ к БД через драйвер JDBC. Не забудьте перед его использованием установить драйверы JDBC для соответствующей операционной системы и базы данных. В числе других полезных пакетов, основанных на DBI, можно назвать: RMySQL, ROracle, RPostgreSQL и RSQLite. Эти пакеты содержат также оригинальные драйверы для конкретных баз данных, однако они могут работать не во всех операционных системах. Ознакомьтесь с документацией на CRAN (http://cran.r-project.org), чтобы узнать подробности.

2.3.10. Импорт данных при помощи Stat/Transfer

Прежде чем закончить обсуждение импорта данных, стоит упомянуть коммерческий продукт, способный значительно упростить импорт данных. Stat/Transfer (https://www.stattransfer.com/) – это самостоятельная программа, которая может преобразовывать данные в любой из 34 форматов, включая R (рис. 2.4).

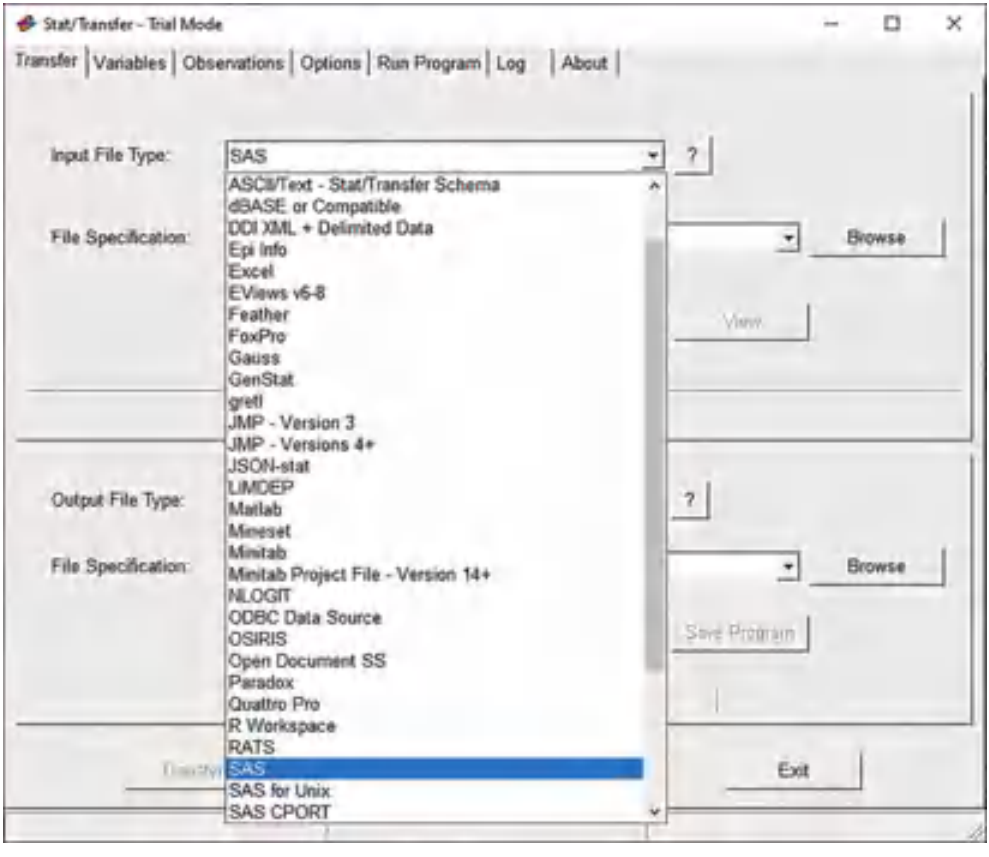


Рис. 2.4. Основное окно программы Stat/Transfer в Windows

Существуют версии для Windows, Mac и Unix, которые поддерживают форматы данных самых последних версий всех статистических программ, упоминавшихся выше, а также такие БД, как Oracle, Sybase, Informix и DB/2.

2.4. Аннотирование наборов данных

Исследователи обычно аннотируют свои наборы данных, чтобы упростить интерпретацию результатов. Аннотирование, как правило, заключается в добавлении поясняющих подписей (меток) к именам переменных и кодам данных, используемым в категориальных переменных. Например, переменную `age` можно снабдить пояснительной подписью `"Age at hospitalization (in years)"` («Возраст в момент госпитализации (в годах)»). А коды значений `1` и `2` для переменной `gender` – подписями `"male"` (мужской) и `"female"` (женский).

2.4.1. Подписи для переменных

К сожалению, R имеет довольно ограниченные возможности для работы с подписями для переменных. Один из подходов: оформить подпись в виде имени переменной, а в операциях ссылаться на нее по ее индексу. Вернемся к предыдущему примеру с данными о пациентах. Во втором столбце, `age`, хранится возраст пациентов, в котором они попали в больницу. Инструкция

```
names(patientdata)[2] <- "Age at hospitalization (in years)"
```

переименует `age` в `"Age at hospitalization (in years)"`. Конечно, это новое название слишком длинное, чтобы писать его в программах по многу раз. Вместо этого на переменную можно ссылаться как `patientdata[2]`, при этом в выводе будет отображаться ее полное имя `"Age at hospitalization (in years)"`. Конечно, это не идеальное решение, и, возможно, разумнее остановиться на более простом имени (например, `admissionAge`).

2.4.2. Подписи для значений переменных

Функция `factor()` позволяет добавлять подписи к значениям категориальных переменных. Продолжим наш пример и предположим, что у нас есть переменная `gender`, в которой значение «мужской пол» закодировано как `1`, а «женский пол» – как `2`. Подписи к значениям этой переменной можно создать так:

```
patientdata$gender <- factor(patientdata$gender,
                             levels = c(1,2),
                             labels = c("male", "female"))
```

Здесь параметр `levels` определяет фактические значения переменной, а `labels` – текстовый вектор с соответствующими подписями.

2.5. Полезные функции для работы с объектами

Закончим эту главу кратким обзором полезных функций для работы с объектами (табл. 2.4).

Таблица 2.4. Полезные функции для работы с объектами

Функция	Описание
`length(`*`object`*`)`	Возвращает число элементов/компонентов объекта *`object`*
`dim(object)`	Возвращает число измерений объекта *`object`*
`str(object)`	Возвращает структуру объекта *`object`*
`class(`*`object`*`)`	Возвращает класс (тип) объекта *`object`*
`mode(`*`object`*`)`	Возвращает способ хранения (вид) объекта *`object`*
`names(`*`object`*`)`	Возвращает имена компонентов объекта *`object`*

Окончание табл. 2.4

Функция	Описание
`c(object, object, ...)`	Объединяет перечисленные объекты в вектор
`cbind(object, object, ...)`	Объединяет перечисленные объекты в столбцы
`rbind(object, object, ...)`	Объединяет перечисленные объекты в строки
`object`	Выводит объект *object*
`head(object)`	Выводит первую часть объекта *object*
`tail(object)`	Выводит последнюю часть объекта *object*
`ls()`	Выводит список существующих объектов
`rm(object, object, ...)`	Удаляет перечисленные объекты. Инструкция `rm(list = ls())` удалит почти все объекты из рабочего пространства
newobject `<-` `edit(object)`	Поможет исправить объект *object* и сохранить результат правки в новом объекте *newobject*
`fix(object)`	Исправляет объект *object* на месте

Мы уже обсудили большинство из этих функций. `head()` и `tail()` используются для быстрого просмотра больших наборов данных. Например, `head(patientdata)` выведет первые шесть строк из таблицы данных, а `tail(patientdata)` – последние шесть. Функции `length()`, `cbind()` и `rbind()` мы рассмотрим в следующей главе. Здесь они упомянуты только для справки.

Как вы успели узнать, в R имеется множество функций для доступа к внешним источникам данных. Экспорт данных из R в другие форматы обсуждается в приложении C, а методы работы с большими наборами данных (объем которых измеряется в гигабайтах или терабайтах) рассматриваются в приложении F.

После импортирования данных в R вам, скорее всего, нужно будет преобразовать их в более удобный для работы вид. В главе 3 мы рассмотрим способы создания новых переменных, преобразования и перекодирования имеющихся переменных, объединения наборов данных и выборки отдельных наблюдений.

Итоги

- R предоставляет несколько видов объектов для хранения данных, включая векторы, матрицы, таблицы данных и списки.
- Данные можно импортировать в таблицы R из самых разных внешних источников, включая текстовые файлы, листы Excel, веб-службы, статистические программы и базы данных.
- Существует множество функций для описания, изменения и объединения структур данных.

3 Основы управления данными

В этой главе:

- работа с датами и пропущенными значениями;
- преобразование данных из одного типа в другой;
- создание и перекодирование переменных;
- сортировка, объединение и разделение наборов данных;
- выборка и исключение переменных из анализа.

Во второй главе мы обсудили разнообразные методы импорта данных в R. К сожалению, преобразование данных в прямоугольную форму матрицы или таблицы данных – это только первый шаг на пути их подготовки к анализу. Перефразируя капитана Кирка из серии «Вкус Армагеддона» сериала «Звездный путь» (и раз и навсегда оправдывая мое помешательство): «Данные – это запутанная штука, очень и очень запутанная штука». В моей собственной работе не менее 60 % времени, отведенного на анализ данных, я трачу на подготовку данных к нему. Я выйду за пределы частного опыта и скажу, что это, вероятно, в той или иной степени справедливо для большинства исследователей. Рассмотрим пример.

3.1. Рабочий пример

Одна из задач, которую я решаю по долгу службы, – выявляю различия в стиле руководства организациями между мужчинами и женщинами. Вот типичные вопросы, которые я задаю себе:

- различаются ли мужчины и женщины на руководящих должностях по степени лояльности к вышестоящему начальству?
- зависит ли это от страны или выявленные гендерные различия носят универсальный характер?

Один из способов ответить на эти вопросы – взять руководителей из разных стран и ранжировать подчиненных им менеджеров по степени лояльности, например так:

Этот менеджер спрашивает мое мнение перед принятием кадровых решений.				
1 Абсолютно не согласен	2 Не согласен	3 Бывает по-разному	4 Согласен	5 Абсолютно согласен

В результате можно получить данные вроде тех, что представлены в табл. 3.1. Каждая строка – это оценка, которую дал менеджеру его или ее начальник.

Таблица 3.1. Гендерные различия в стиле руководства

Maneger (менеджер)	Date (дата)	Country (страна)	Gender (пол)	Age (возраст)	q1	q2	q3	q4	q5
1	10/24/14	US	M	32	5	4	5	5	5
2	10/28/14	US	F	45	3	5	2	5	5
3	10/01/14	UK	F	25	3	5	5	5	2
4	10/12/14	UK	M	39	3	3	4		
5	05/01/09	UK	F	99	2	2	1	2	1

Здесь каждый менеджер оценен своим начальником по пяти параметрам (`q1`–`q5`), связанным с лояльностью к вышестоящим руководителям. Например, менеджер 1 – это 32-летний мужчина, работающий в США, который склонен подчиняться начальству, тогда как менеджер 5 – это женщина неизвестного возраста (99, вероятно, означает отсутствие информации), работающая в Великобритании и недостаточно лояльная к начальству. В табл. 3.1 также указана дата проведения опроса.

Подобные наборы данных могут состоять из десятков переменных и тысяч наблюдений, но мы оставили только десять столбцов и пять строк для простоты. Кроме того, мы ограничили число во-

просов, характеризующих лояльность менеджеров к начальству. В реальном исследовании обычно используют 10–20 вопросов, чтобы получить более надежные и обоснованные результаты. На основе сведений в табл. 3.1 можно создать таблицу данных, как показано в листинге 3.1.

Листинг 3.1. Создание таблицы данных leadership

```
leadership <- data.frame(
   manager = c(1, 2, 3, 4, 5),
   date    = c("10/24/08", "10/28/08", "10/1/08", "10/12/08", "5/1/09"),
   country = c("US", "US", "UK", "UK", "UK"),
   gender  = c("M", "F", "F", "M", "F"),
   age     = c(32, 45, 25, 39, 99),
   q1      = c(5, 3, 3, 3, 2),
   q2      = c(4, 5, 5, 3, 2),
   q3      = c(5, 2, 5, 4, 1),
   q4      = c(5, 5, 5, NA, 2),
   q5      = c(5, 5, 2, NA, 1)
)
```

Для ответа на интересующие нас вопросы нужно сначала решить несколько проблем, связанных с управлением данными. Вот их неполный список:

- нужно объединить пять рейтингов (от `q1` до `q5`), чтобы для каждого менеджера получить единый усредненный показатель лояльности к руководству;
- при анкетировании респонденты часто пропускают вопросы. Например, начальник, который оценивал менеджера 4, не ответил на вопросы 4 и 5. Нам потребуется как-то обработать неполные данные. Также нужно будет обозначить значения вроде `99` как *отсутствующие*;
- набор данных может содержать сотни переменных, но нас, скорее всего, заинтересуют только некоторые из них. Чтобы упростить себе работу, может появиться желание создать новый набор данных, состоящий только из этих переменных;
- предыдущие исследования показали, что лояльность к начальству может меняться с возрастом. Чтобы проверить это, можно попробовать перекодировать значения возраста в возрастные группы (например, молодые, среднего возраста и старшего возраста);
- лояльность к начальству может меняться со временем. Чтобы проверить это, можно сосредоточиться на периоде последнего глобального финансового кризиса, ограничившись данными, собранными, скажем, с 1 января по 31 декабря 2009 года.

В этой главе мы разберем все перечисленные проблемы и основные задачи управления данными, такие как объединение и сортировка наборов данных. Затем в главе 5 мы обратимся к более сложным темам.

3.2. Создание новых переменных

Обычно при анализе данных приходится создавать новые переменные и преобразовывать существующие. Это можно сделать с помощью операции присваивания:

```
переменная <- выражение
```

Под *выражением* здесь подразумевается широкий спектр функций и операторов. В табл. 3.2 перечислены арифметические операторы, которые используются при составлении формул в R.

Таблица 3.2. Арифметические операторы

Оператор	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
^ или **	Возведение в степень
x%%y	Деление по модулю (остаток от деления) x на y: 5%%2=1
x%/%y	Деление нацело x на y: 5%/%2=2

Представим, что у нас есть таблица данных с именем `leadership` и в нее требуется добавить новые переменные: `total_score`, представляющую сумму значений переменных от `q1` до `q5`, и `mean_score`, представляющую среднее арифметическое значений этих же переменных. Если попробовать выполнить такой код:

```
total_score  <-  q1 + q2 + q3 + q4 + q5
mean_score <- (q1 + q2 + q3 + q4 + q5)/5
```

он сообщит об ошибке, потому что R не знает о том, что `q1`, `q2`, `q3`, `q4` и `q5` – это часть таблицы данных `leadership`. Если попробовать выполнить такой код:

```
total_score  <-  leadership$q1 + leadership$q2 + leadership$q3 +
                   leadership$q4 + leadership$q5
mean_score <- (leadership$q1 + leadership$q2 + leadership$q3 +
                   leadership$q4 + leadership$q5)/5
```

он вычислит требуемые значения, но в результате мы получим исходную таблицу данных (`leadership`) и два отдельных вектора (`total_score` и `mean_score`). Однако это совсем не то, что нам нужно. Нам нужно было создать новые переменные и сделать их частью исходной таблицы данных. В листинге 3.2 показаны два альтернативных способа достичь этой цели. Вы можете выбрать любой, какой вам больше понравится, результат от этого не изменится.

Листинг 3.2. Создание новых переменных

```
leadership$total_score  <-  leadership$q1 + leadership$q2 + leadership$q3 +
                             leadership$q4 + leadership$q5
leadership$mean_score <- (leadership$q1 + leadership$q2 + leadership$q3 +
                          leadership$q4 + leadership$q5)/5

leadership <- transform(leadership,
                  total_score  =  q1 + q2 + q3 + q4 + q5,
                  mean_score = (q1 + q2 + q3 + q4 + q5)/5)
```

Лично я предпочитаю второй метод, с применением функции `transform()`. Она упрощает создание необходимого числа новых переменных и сохраняет результат в таблице данных.

3.3. Перекодирование переменных

При перекодировании новые значения переменной определяются на основе значений этой и/или других переменных. Например, можно:

- преобразовать непрерывные данные в категориальные;
- заменить ошибочные данные правильными значениями;
- создать переменную типа «сдал / не сдал» на основе экзаменационных баллов.

Для перекодирования данных можно использовать один или несколько логических операторов (табл. 3.3), то есть выражений, возвращающих значения `TRUE`/`FALSE`.

Таблица 3.3. Логические операторы

Оператор	Описание
<	Меньше, чем
<=	Меньше или равно
>	Больше, чем
>=	Больше или равно
==	Точно равно

Окончание табл. 3.3

Оператор	Описание
!=	Не равно
!x	Инверсия x (НЕ x)
x | y	x ИЛИ y
x & y	x И y
isTRUE(x)	Проверяет истинность значения x

Предположим, что мы решили перекодировать возраст менеджеров в нашем наборе данных из непрерывной переменной `age` в категориальную переменную `agecat` (`Young`, `Middle Aged`, `Elder` – молодой, средних лет, старшего возраста). Сначала нужно закодировать значение `99` как отсутствующее:

```
leadership$age[leadership$age == 99] <- NA
```

Инструкция вида `переменная[условие] <- выражение` выполнит присваивание, только если `условие` выполняется.

После выявления пропущенных значений можно переходить к созданию переменной `agecat`:

```
leadership$agecat[leadership$age  > 75]   <- "Elder"
leadership$agecat[leadership$age >= 55 &
                  leadership$age <= 75]   <- "Middle Aged"
leadership$agecat[leadership$age  < 55]   <- "Young"
```

Имя таблицы данных входит в `leadership$agecat`, поэтому новая переменная сохранится именно в этой таблице. (Для среднего возраста я выбрал диапазон от 55 до 75 лет, чтобы не чувствовать себя слишком старым.) Обратите внимание, что если предварительно не закодировать значение `99` как пропущенное, то менеджер 5 был бы ошибочно отнесен к категории людей старшего возраста.

Тот же код можно записать более компактно:

```
leadership <- within(leadership,{
                     agecat <- NA
                     agecat[age > 75]              <- "Elder"
                     agecat[age >= 55 & age <= 75] <- "Middle Aged"
                     agecat[age < 55]              <- "Young" })
```

Функция `within()` похожа на функцию `with()` (раздел 2.2.4), но, в отличие от последней, позволяет модифицировать таблицу данных. Она сначала создаст переменную `agecat`, состоящую из пропущенных значений. Затем инструкции в фигурных скобках последовательно заменят эти значения соответствующими возрастными категориями, согласно условиям в квадратных скобках. Напомню, что `agecat` – это

текстовая переменная, и иногда может понадобиться преобразовать ее в упорядоченный фактор, как описывается в разделе 2.2.5.

Существует ряд пакетов с полезными функциями для перекодирования; в частности, в пакете car имеется функция recode(), упрощающая перекодирование числовых и текстовых векторов и факторов. Пакет doBy предлагает recodeVar() – еще одну популярную функцию. Наконец, в R имеется функция cut(), которая делит числовые переменные на интервалы, возвращая фактор.

3.4. Переименование переменных

Если вас не устраивают названия переменных, их можно переименовать в интерактивном режиме или программно. Допустим, нам захотелось изменить имя переменной manager на managerID и date на testDate. Это можно сделать с помощью инструкции

```
fix(leadership)
```

Она откроет интерактивный редактор, в котором нужно выбрать имена переменных и изменить их в диалоговом режиме (рис. 3.1).

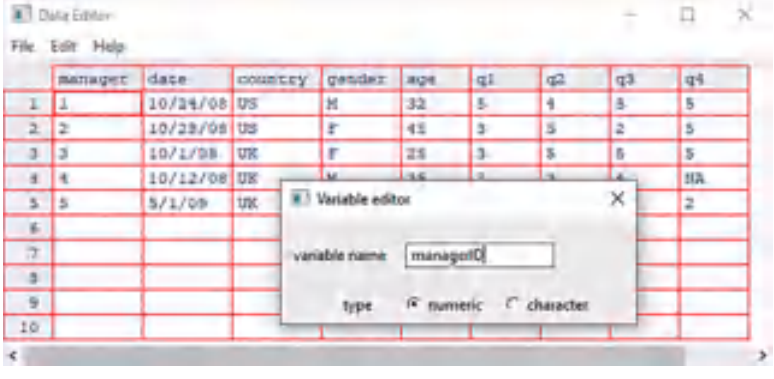


Рис. 3.1. Переименование переменных с использованием функции fix()

Изменить имена переменных программно можно с помощью функции names(). Например, инструкция

```
names(leadership)[2] <- "testDate"
```

переименует переменную date в testDate, как показано ниже:

```
> names(leadership)
 [1] "manager" "date"     "country" "gender"   "age"      "q1"       "q2"
 [8] "q3"      "q4"       "q5"
> names(leadership)[2] <- "testDate"
> leadership
  manager testDate country gender age q1 q2 q3 q4 q5
1       1 10/24/08      US      M  32  5  4  5  5  5
2       2 10/28/08      US      F  45  3  5  2  5  5
```

```
3       3  10/1/08      UK      F  25  3  5  5  5  2
4       4 10/12/08      UK      M  39  3  3  4 NA NA
5       5   5/1/09      UK      F  99  2  2  1  2  1
```

Аналогично инструкция

```
names(leadership)[6:10] <- c("item1", "item2", "item3", "item4", "item5")
```

переименует переменные `q1`–`q5` в `item1`–`item5`.

3.5. Пропущенные значения

При любых исследованиях данные с большой вероятностью будут неполными из-за пропущенных вопросов, барахлящего оборудования или ошибок, допущенных при вводе. В R пропущенные данные обозначаются символом `NA` (not available – нет в наличии). В отличие от таких программ, как SAS, в R используется одно и то же обозначение для пропущенных значений в текстовых и числовых данных.

В R есть несколько функций, предназначенных для выявления пропущенных значений. Функция `is.na()` позволяет проверить данные на наличие пропущенных значений. Предположим, что у нас имеется вектор

```
y <- c(1, 2, 3, NA)
```

Тогда инструкция

```
is.na(y)
```

вернет вектор `c(FALSE, FALSE, FALSE, TRUE)`.

Обратите внимание, как функция `is.na()` работает с объектом. Она возвращает объект того же размера, в котором элементы заменены символом `TRUE`, если значение пропущено, и `FALSE` – если не пропущено. В листинге 3.3 показано, что получится, если эту инструкцию применить к таблице данных `leadership`.

Листинг 3.3. Использование функции is.na()

```
> is.na(leadership[,6:10])
        q1    q2    q3    q4    q5
[1,] FALSE FALSE FALSE FALSE FALSE
[2,] FALSE FALSE FALSE FALSE FALSE
[3,] FALSE FALSE FALSE FALSE FALSE
[4,] FALSE FALSE FALSE  TRUE  TRUE
[5,] FALSE FALSE FALSE FALSE FALSE
```

В этом примере `leadership[,6:10]` ограничивает таблицу данных столбцами с 6 по 10, а функция `is.na()` выявляет пропущенные значения.

При работе с пропущенными значениями в R важно помнить обстоятельства. Во-первых, пропущенные значения нельзя срав-

нивать, даже с самими собой, т. е. нельзя использовать операторы сравнения для выявления пропущенных значений. Например, логическое выражение `myvar == NA` никогда не вернет `TRUE`. Вместо этого для выявления пропущенных значений следует использовать специальные функции, такие как `is.na()`.

Во-вторых, R представляет бесконечные или невозможные значения иначе, не как отсутствующие значения. И в этом он тоже отличается от других программ, таких как SAS, способных обрабатывать такие данные. Положительная и отрицательная бесконечности представлены символами `Inf` и `-Inf` соответственно. Таким образом, выражение `5/0` вернет `Inf`. Невозможные значения (например, `sin(Inf)`) обозначаются символом `NaN` (а не числом). Чтобы определить эти значения, нужно использовать `is.infinite()` или `is.nan()`.

3.5.1. Перекодирование значений в отсутствующие

Как отмечалось в разделе 3.3, для перекодирования значений в *отсутствующие* можно использовать инструкцию присваивания. В примере с таблицей `leadership` пропущенные значения возраста были закодированы как `99`. Перед тем как анализировать такой набор данных, нужно обозначить значение `99` как *отсутствующее* (иначе среднее значение возраста для этой выборки будет далеким от реальности!). Этого можно достичь с помощью инструкции перекодирования переменной:

```
leadership$age[leadership$age == 99] <- NA
```

Любое значение возраста, равное `99`, будет заменено на `NA`. Перед анализом данных всегда проверяйте, что все пропущенные значения закодированы соответственно, иначе результаты анализа будут лишены смысла.

3.5.2. Исключение пропущенных значений из анализа

После выявления пропущенных значений их нужно каким-то образом удалить, чтобы исключить из анализа данных. Это необходимо, потому что арифметические операции и функции, применяемые к данным с пропущенными значениями, будут возвращать отсутствующие значения. Например:

```
x <- c(1, 2, NA, 3)
y <- x[1] + x[2] + x[3] + x[4]
z <- sum(x)
```

Здесь обе переменные, `y` и `z`, получат значение `NA`, поскольку третий элемент вектора `x` отсутствует.

К счастью, большая часть числовых функций принимает параметр `na.rm=TRUE`, удаляющий пропущенные значения перед вычислениями, и применяет функцию к оставшимся элементам:

```
x <- c(1, 2, NA, 3)
y <- sum(x, na.rm=TRUE)
```

Теперь y получит значение 6.

Собираясь применить функцию к неполным данным, узнайте в справке (например, help(sum)), как она обрабатывает пропущенные значения. Функция sum() – это лишь одна из многих функций, которые мы рассмотрим в главе 5. Подобные функции позволяют с легкостью осуществлять разнообразные преобразования данных.

При помощи функции na.omit() можно избавиться от всех наблюдений с пропущенными данными. Она удалит все строки, имеющие хотя бы одно пропущенное значение. Давайте применим эту функцию к нашей таблице leadership (листинг 3.4).

Листинг 3.4. Использование функции na.omit() для удаления неполных наблюдений

```
> leadership
  manager     date country gender age q1 q2 q3 q4 q5      (1)
1       1 10/24/08      US      M  32  5  4  5  5  5      (1)
2       2 10/28/08      US      F  40  3  5  2  5  5      (1)
3       3 10/01/08      UK      F  25  3  5  5  5  2      (1)
4       4 10/12/08      UK      M  39  3  3  4 NA NA      (1)
5       5 05/01/09      UK      F  NA  2  2  1  2  1      (1)

> newdata <- na.omit(leadership)
> newdata
  manager     date country gender age q1 q2 q3 q4 q5      (2)
1       1 10/24/08      US      M  32  5  4  5  5  5      (2)
2       2 10/28/08      US      F  40  3  5  2  5  5      (2)
3       3 10/01/08      UK      F  25  3  5  5  5  2      (2)
```

(1) **Таблица данных с пропущенными значениями**

(2) **Таблица данных только с полными записями**

Строки с пропущенными значениями будут удалены из таблицы leadership перед сохранением результатов в таблице newdata.

Удаление всех наблюдений с пропущенными значениями (так называемое *построчное удаление*) – один из способов обработки неполных наборов данных. В том случае, когда пропущено лишь несколько значений в небольшом количестве строк, построчное удаление – хороший способ решить проблему пропущенных значений. Однако если пропущенные значения рассеяны по всей таблице данных или находятся в нескольких переменных, построчное удаление может уничтожить заметную часть данных. В главе 18 мы познакомимся с некоторыми более сложными способами работы с пропущенными данными. А теперь поговорим о календарных датах.

3.6. Календарные даты

В R даты обычно вводятся в виде текстовых строк, а затем преобразуются в формат даты и хранятся в числовом виде. Для этого преобразования используется функция `as.Date()`, имеющая следующий синтаксис:

```
as.Date(x, "input_format")
```

где `x` – это дата в текстовом формате, а `input_format` определяет формат представления даты (табл. 3.4).

Таблица 3.4. Форматы представления дат

Символ	Значение	Пример
%d	День месяца в виде числа (01–31)	01–31
%a	Сокращенное название дня недели	Mon
%A	Полное название дня недели	Monday
%m	Порядковый номер месяца (01–12)	01–12
%b	Сокращенное название месяца	Jan
%B	Полное название месяца	January
%y	Две последние цифры года	07
%Y	Все четыре цифры года	2007

По умолчанию даты вводятся в формате `yyyy-mm-dd` («год-месяц-день»). Инструкция

```
mydates <- as.Date(c("2007-06-22", "2004-02-13"))
```

преобразует текстовые данные в даты, используя этот формат по умолчанию. В отличие от нее, следующая инструкция:

```
strDates <- c("01/05/1965", "08/16/1975")
dates <- as.Date(strDates, "%m/%d/%Y")
```

читает даты в формате `mm/dd/yyyy` («месяц/день/год»).

В нашем наборе данных `leadership` даты представлены в формате `mm/dd/yy`. Поэтому в инструкциях

```
myformat <- "%m/%d/%y"
leadership$date <- as.Date(leadership$date, myformat)
```

используется нестандартный формат, позволяющий прочитать текстовую переменную и преобразовать ее в столбец, содержащий календарные даты. После преобразования переменной в формат даты ее можно анализировать и отображать графически, используя разнообразные аналитические приемы, описанные в следующих главах.

Для создания временных отметок особенно удобно использовать две функции. Функция `Sys.Date()` возвращает текущую дату, а функция `date()` – текущие дату и время. Я пишу эти строки 7 июля 2021 года в 18:43 и, выполнив следующие инструкции, получил показанные результаты:

```
> Sys.Date()
[1] "2021-07-20"
> date()
[1] "Tue Jul 20 18:43:40 2021"
```

Для вывода дат в заданном формате и извлечения отдельных компонентов можно использовать функцию `format(x, format="output_format")`:

```
> today <- Sys.Date()
> format(today, format="%B %d %Y")
[1] "July 20 2021"
> format(today, format="%A")
[1] "Tuesday"
```

Функция `format()` преобразует аргумент (в данном случае дату) в заданный формат вывода (состоящий из символов, описание которых приводится в табл. 3.4). Самое важное здесь – до выходных осталось всего два дня!

Даты хранятся в памяти R в виде числа дней, прошедших с 1 января 1970 года. Более ранние даты представлены отрицательными числами. Это значит, что с датами можно выполнять арифметические действия. Например, вот как можно узнать, сколько дней прошло с 13 февраля 2020 года по 22 января 2021 года:

```
> startdate <- as.Date("2020-02-13")
> enddate   <- as.Date("2021-01-22")
> days      <- enddate - startdate
> days
Time difference of 344 days
```

Наконец, для вычисления продолжительности временного отрезка в секундах, минутах, часах, днях или неделях можно использовать функцию `difftime()`. Предположим, что я родился 12 октября 1956 года. Сколько времени прошло с той поры?

```
> today <- Sys.Date()
> dob   <- as.Date("1956-10-12")
> difftime(today, dob, units="weeks")
Time difference of 3380 weeks
```

Оказывается, мне 3380 недель. Кто бы мог подумать? Вопрос на засыпку: в какой день недели я родился?

3.6.1. Преобразование дат в текстовые переменные

Также есть возможность преобразовать данные в формате даты и времени в текстовые значения, хотя это бывает нужно не так часто. Это можно сделать при помощи функции `as.character()`:

```
strDates <- as.character(dates)
```

Это позволяет применять к датам многие текстовые функции (разделение на подгруппы, замена, конкатенация и т. д.). Текстовые функции будут детально разобраны в разделе 5.2.4.

3.6.2. Получение дополнительной информации

Чтобы узнать больше о преобразовании текстовых переменных в даты, загляните в справку `help(as.Date)` и `help(strftime)`. Также дополнительную информацию о форматах даты и времени можно найти в `help(ISOdatetime)`. Пакет `lubridate` содержит множество функций, упрощающих работу с датами, в том числе функции, обнаруживающие данные в формате даты и времени и извлекающие отдельные составляющие таких данных (например, годы, месяцы, дни и т. д.), а также выполняющие арифметические операции с ними. Если вам нужно произвести сложные вычисления с датами, воспользуйтесь пакетом `fCalendar`. В нем содержится несметное число функций для работы с датами. Эти функции позволяют одновременно работать с несколькими часовыми поясами и совершать сложные операции с календарем, распознавая рабочие, выходные и праздничные дни.

3.7. Преобразования данных из одного типа в другой

В предыдущем разделе обсуждалось преобразование текстовых значений в формат даты и наоборот. В R существует еще множество функций, позволяющих определить тип данных и преобразовать их в другой формат. Такие преобразования выполняются в R так же, как в иных языках статистического программирования. Например, при сложении текстового вектора с числовым все числовые значения преобразуются в текстовый формат. Для проверки и преобразования типа данных можно использовать функции, перечисленные в табл. 3.5.

Функции с именами вида `is.тип_данных()` возвращают `TRUE` или `FALSE`, а функции с именами вида `as.тип_данных()` преобразуют данные в соответствующий формат. Пример их использования показан в листинге 3.5.

Таблица 3.5. Функции преобразования типов данных

Функции проверки	Функции преобразования
`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`
`is.factor()`	`as.factor()`
`is.logical()`	`as.logical()`

Листинг 3.5. Преобразование данных из одного типа в другой

```
> a <- c(1,2,3)
> a
[1] 1 2 3
> is.numeric(a)
[1] TRUE
> is.vector(a)
[1] TRUE
> a <- as.character(a)
> a
[1] "1" "2" "3"
> is.numeric(a)
[1] FALSE
> is.vector(a)
[1] TRUE
> is.character(a)
[1] TRUE
```

Функции с именами `is.тип_данных()` в сочетании с операторами управления потоком выполнения (такими как `if-then`), которые мы обсудим в главе 5, могут стать мощным инструментом обработки данных разных типов по-разному. Кроме того, некоторые функции в R работают только с данными определенного типа (текстовыми или числовыми, матрицами или таблицами данных). Функции `as.тип_данных()` позволяют преобразовать данные в нужный формат перед началом их анализа.

3.8. Сортировка данных

Иногда просмотр отсортированных данных помогает лучше разобраться в них. К примеру, какие менеджеры наиболее лояльно относятся к начальству? Для сортировки таблицы данных в R используется функция `order()`. По умолчанию данные сортируются в порядке возрастания. Добавьте перед интересующей вас пере-

менной знак минус, чтобы отсортировать ее значения в порядке убывания[1]. Следующий пример иллюстрирует сортировку таблицы данных `leadership` по возрасту менеджеров:

```
newdata <- leadership[order(leadership$age),]
```

Эта инструкция создаст новый набор данных, в котором строки отсортированы, начиная с самого молодого менеджера и заканчивая самым старым.

Инструкция

```
newdata <- leadership[order(leadership$gender, leadership$age),]
```

отсортирует сначала по полу, а затем по возрасту.

Наконец, инструкция

```
newdata <-leadership[order(leadership$gender, -leadership$age),]
```

сортирует строки сначала по полу, а потом, в пределах каждого пола, по возрасту – в порядке убывания.

3.9. Объединение наборов данных

Если данные существуют в виде разрозненных фрагментов, их нужно объединить, прежде чем двигаться дальше. В этом разделе я расскажу, как добавлять столбцы (переменные) и строки (наблюдения) в таблицы данных.

3.9.1. Добавление столбцов

Для объединения двух таблиц можно использовать функцию `merge()`. В большинстве случаев две таблицы объединяются по значениям одной или нескольких ключевых переменных. К примеру, инструкция

```
total <- merge(dataframeA, dataframeB, by="ID")
```

объединит таблицы данных `dataframeA` и `dataframeB` по значениям переменной `ID`. Аналогично инструкция

```
total <- merge(dataframeA, dataframeB, by=c("ID","Country"))
```

объединит две таблицы данных по значениям переменных `ID` и `Country`. Подобное соединение таблиц данных по горизонтали часто используется для добавления переменных в таблицы данных.

[1] Такой прием работает не всегда. Правильнее использовать аргумент `decreasing=T`. Например, вот как можно отсортировать список менеджеров в порядке убывания возраста: `leadership[order(leadership$age, decreasing=T),]`. – *Прим. перев.*

Объединение по горизонтали с помощью cbind()

Если нужно просто объединить две матрицы или таблицы данных по горизонтали без указания общего ключа, по которому должно выполняться объединение, можно использовать функцию `cbind()`:

```
total <- cbind(A, B)
```

Эта функция объединяет объекты A и B по горизонтали. Чтобы функция работала правильно, каждый объект должен иметь одинаковое число строк, расположенных в одинаковом порядке.

3.9.2. Добавление строк

Для объединения двух таблиц данных по вертикали можно использовать функцию `rbind()`:

```
total <- rbind(dataframeA, dataframeB)
```

Объединяемые таблицы должны содержать одинаковые наборы переменных, но не обязательно в одной и той же последовательности. Если в таблице `dataframeA` есть переменные, отсутствующие в `dataframeB`, то перед объединением нужно сделать одно из двух:

- удалить лишние переменные из таблицы `dataframeA`;
- создать дополнительные переменные в таблице `dataframeB` и присвоить им значения `NA` (пропущенные).

Объединение по вертикали обычно используется для добавления наблюдений из одной таблицы данных в другую.

3.10. Разделение наборов данных на составляющие

В R имеются обширные возможности для извлечения отдельных элементов из объектов. Их можно использовать для выбора определенных переменных и/или наблюдений. В следующих разделах обсуждается несколько способов выбора переменных и наблюдений.

3.10.1. Выбор переменных

Часто бывает так, что новый набор данных создается из небольшого числа переменных, выбранных из большего набора данных. В главе 2 вы узнали, как выбирать элементы таблицы данных при помощи инструкции вида `таблица_данных[номера_строк, номера_столбцов]`. Этот прием можно использовать также для выбора отдельных переменных. Например, инструкция

```
newdata <- leadership[,c(6:10)]
```

выберет переменные `q1`, `q2`, `q3`, `q4` и `q5` из таблицы данных `leadership` и сохранит их в таблице данных `newdata`. Если номера строк не указаны `(,)`, то по умолчанию выбираются все строки.

Инструкции

```
vars <- c("q1", "q2", "q3", "q4", "q5")
newdata <-leadership[, vars]
```

выберут те же самые переменные. В этом случае имена переменных (в кавычках) обозначают извлекаемые переменные.

Если для таблицы данных указан только один набор индексов, то R интерпретирует его как набор столбцов. В следующей инструкции подразумевается присутствие запятой перед `vars`:

```
newdata <- leadership[vars]
```

и она вернет тот же набор переменных.

Наконец, тот же результат можно получить с помощью инструкций:

```
myvars <- paste("q", 1:5, sep="")
newdata <- leadership[myvars]
```

Этот пример создаст тот же вектор, что и предыдущий, но на этот раз с использованием функции `paste()`. Эту функцию мы подробно рассмотрим в главе 5.

3.10.2. Исключение переменных из выборки

Существует много причин, когда желательно исключить переменные из выборки. Например, если переменная содержит несколько пропущенных значений, то может понадобиться удалить ее до начала анализа. Рассмотрим несколько способов исключения переменных.

Переменные `q3` и `q4` можно исключить с помощью следующих инструкций:

```
myvars <- names(leadership) %in% c("q3", "q4")
newdata <- leadership[!myvars]
```

Чтобы понять, как они работают, разберем их по частям:

1. `names(leadership)` создает текстовый вектор с именами переменных:

```
c("managerID","testDate","country","gender","age","q1","q2","q3","q4","q5")
```

2. `names(leadership)%in% c("q3", "q4")` возвращает логический вектор со значениями `TRUE` для каждого элемента вектора `names(leadership)`, соответствующего переменной `q3` или `q4`, и со значениями `FALSE` во всех остальных случаях:

```
c(FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, FALSE)
```

3. оператор «НЕ» (`!`) изменяет логические значения на противоположные:

```
c(TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE)
```

4 `leadership[c(TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE)]` выбирает столбцы, для которых значения логического вектора равны `TRUE`, соответственно, `q3` и `q4` исключаются из выборки.

Зная, что `q3` и `q4` – это восьмая и девятая переменные, можно исключить их так:

```
newdata <- leadership[c(-8,-9)]
```

Минус (`-`) перед номером столбца означает, что этот столбец следует исключить[1].

Наконец, эти же два столбца можно исключить инструкцией

```
leadership$q3 <- leadership$q4 <- NULL
```

Здесь переменным `q3` и `q4` присваивается неопределенное значение (`NULL`). Обратите внимание, что `NULL` – это не то же самое, что `NA` (отсутствующие значения).

Исключение переменных – действие, противоположное выборке. Выбор между этими действиями зависит от того, какое из них проще записать. Если нужно исключить много переменных, то часто проще выбрать все остальные, и наоборот.

3.10.3. Выборка наблюдений

Выбор или удаление наблюдений (строк) – это в большинстве случаев залог успешной подготовки данных и их анализа. В листинге 3.6 показано несколько примеров.

Листинг 3.6. Выборка наблюдений

```
newdata <- leadership[1:3,]                                    (1)

newdata <- leadership[leadership$gender=="M" &                 (2)
                      leadership$age > 30,]                    (3)
```

(1) Выбрать строки с 1 по 3 (первые три наблюдения)

(2) (3) Выбрать всех мужчин старше 30 лет

В этих примерах указаны только индексы строк, а индексы номеров столбцов опущены (т. е. выбираются все столбцы). Давайте рассмотрим строку кода (2) поближе, чтобы лучше понять ее.

1 Логическое сравнение `leadership$gender=="M"` создает вектор `c(TRUE, FALSE, FALSE, TRUE, FALSE)`.

2 Логическое сравнение `leadership$age > 30` создает вектор `c(TRUE, TRUE, FALSE, TRUE, TRUE)`.

[1] В данном случае для удаления столбцов проще использовать такой синтаксис: `leadership[-c(8,9)]`. – *Прим. перев.*

3 Логическое выражение `c(TRUE, FALSE, FALSE, TRUE, FALSE) & c(TRUE, TRUE, FALSE, TRUE, TRUE)` создает вектор `c(TRUE, FALSE, FALSE, TRUE, FALSE)`.

4 Инструкция `leadership[c(TRUE, FALSE, FALSE, TRUE, FALSE),]` выбирает из таблицы данных первое и четвертое наблюдения (когда индекс строки равен `TRUE`, строка включается в выборку; когда он равен `FALSE`, строка исключается из выборки). Это соответствует нашим критериям (мужчины старше 30 лет).

В начале этой главы я предположил, что при анализе данных может появиться желание ограничиться наблюдениями, сделанными в период между 1 января и 31 декабря 2009 года. Как это можно реализовать? Вот одно из возможных решений:

```
leadership$date <- as.Date(leadership$date, "%m/%d/%y")      1

startdate <- as.Date("2009-01-01")                           2
enddate   <- as.Date("2009-12-31")                           3

newdata <- leadership[which(leadership$date >= startdate &   4
           leadership$date <= enddate),]                     4
```

1 **Преобразовать даты из текстового формата в формат дат (мм/дд/гг).**

2 **Задать начальную дату.**

3 **Задать конечную дату.**

4 **Выбрать наблюдения, удовлетворяющие заданному критерию, как мы уже делали это в предыдущем примере.**

Обратите внимание, что по умолчанию функция `as.Date()` создает даты в формате гггг/мм/дд, поэтому в инструкциях 2 и 3 формат представления дат в анализируемых строках можно не указывать.

3.10.4. Функция subset()

Примеры в двух предыдущих разделах помогают понять, как R интерпретирует логические векторы и операторы сравнения. Понимание особенностей работы этих примеров поможет понять общие принципы выполнения программного кода на R. Теперь, освоив сложные способы, посмотрим, как сделать то же самое проще.

Функция `subset()` – самый простой, пожалуй, способ выбора переменных и наблюдений. Вот два примера:

```
newdata <- subset(leadership, age >= 35 | age < 23,      1
                  select=c(q1, q2, q3, q4))              1

newdata <- subset(leadership, gender=="M" & age > 25,    2
                  select=gender:q4)                      2
```

1 **Выберет все строки, где значение переменной** `age` **больше или равно 35 или меньше 24, оставляя в них только переменные с** `q1` **по** `q4`.

❷ **Выберет всех мужчин старше 25 лет, оставляя в выбранных строках переменные с** `gender` **по** `q4` (`gender`, `q4` **и все столбцы между ними**).

Вы уже видели оператор двоеточия (`:`) в выражениях типа `от:до` в главе 2. Здесь этот оператор позволяет оставить все переменные в таблице данных, начиная с переменной `от` и заканчивая переменной `до`.

3.10.5. Выборка случайных наблюдений

Получение ограниченных выборок из больших наборов данных – обычное дело при поиске структуры в данных или в машинном обучении. К примеру, вам может понадобиться получить две случайные выборки, чтобы создать прогнозную модель для одной из них и оценить эффективность этой модели на второй выборке. Функция `sample()` позволяет извлекать из набора данных случайные выборки (с замещением или без него).

Вот как можно выбрать три случайных наблюдения из таблицы данных `leadership`:

```
mysample <- leadership[sample(1:nrow(leadership), 3, replace=FALSE),]
```

Первый аргумент функции `sample()` – это вектор с номерами наблюдений, из числа которых будет получена выборка. Здесь вектор состоит из чисел от единицы до числа наблюдений в таблице данных. Второй аргумент – это число элементов в выборке, а третий аргумент указывает, что выборка должна быть создана без замещения. Функция `sample()` возвращает случайно выбранные элементы из вектора в первом аргументе, которые затем используются для выбора строк из таблицы данных[1].

R обладает обширными возможностями создания выборок, включая извлечение и проверку пробных выборок (пакет `sampling`) и анализ сложных выборочных данных (пакет `survey`). Другие методы, основанные на создании случайных выборок, включая бутстреп-анализ и метод повторных выборок, будут описаны в главе 11.

3.11. Использование dplyr для работы с таблицами данных

До этого момента для работы с таблицами данных мы использовали базовые функции R. Однако существует также пакет `dplyr`, включающий функции, которые позволяют решать те же задачи более простым способом. Он быстро превратился в один из самых популярных пакетов R для управления данными.

[1] Существует более специализированная функция `sample.int()`. – *Прим. перев.*

3.11.1. Основные функции из пакета dplyr

Пакет dplyr предлагает набор функций, которые можно использовать для выборки переменных и наблюдений, преобразования и переименования переменных и сортировки строк. Соответствующие функции перечислены в табл. 3.6.

Таблица 3.6. Функции из пакета dplyr для работы с таблицами данных

Функция	Описание
select()	Выбирает переменные/столбцы
filter()	Выбирает наблюдения/строки
mutate()	Преобразует переменные
rename()	Переименовывает переменные/столбцы
recode()	Перекодирует значения переменных
arrange()	Упорядочивает строки по значениям переменных

Вернемся к таблице данных, представленной в табл. 3.1 и повторно воспроизведенной в табл. 3.7 для удобства.

Таблица 3.7. Гендерные различия в стиле руководства

Maneger (менеджер)	Date (дата)	Country (страна)	Gender (пол)	Age (возраст)	q1	q2	q3	q4	q5
1	10/24/14	US	M	32	5	4	5	5	5
2	10/28/14	US	F	45	3	5	2	5	5
3	10/01/14	UK	F	25	3	5	5	5	2
4	10/12/14	UK	M	39	3	3	4		
5	05/01/09	UK	F	99	2	2	1	2	1

На этот раз для работы с набором данных используем функции из пакета dplyr. Код примера приводится в листинге 3.7. Поскольку пакет dplyr не является стандартной частью R, его нужно предварительно установить (install.packages("dplyr")).

Листинг 3.7. Манипулирование данными с помощью функций из пакета dplyr

```
leadership <- data.frame(
   manager = c(1, 2, 3, 4, 5),
   date    = c("10/24/08", "10/28/08", "10/1/08", "10/12/08", "5/1/09"),
   country = c("US", "US", "UK", "UK", "UK"),
   gender  = c("M", "F", "F", "M", "F"),
   age     = c(32, 45, 25, 39, 99),
```

```
  q1      = c(5, 3, 3, 3, 2),
  q2      = c(4, 5, 5, 3, 2),
  q3      = c(5, 2, 5, 4, 1),
  q4      = c(5, 5, 5, NA, 2),
  q5      = c(5, 5, 2, NA, 1)
  )

library(dplyr)                                                    (1)

leadership <- mutate(leadership,                                  (2)
                     total_score = q1 + q2 + q3 + q4 + q5,        (2)
                     mean_score = total_score / 5)                (2)

leadership$gender <- recode(leadership$gender,                    (3)
                            "M" = "male", "F" = "female")         (3)

leadership <- rename(leadership, ID = "manager", sex = "gender")  (4)

leadership <- arrange(leadership, sex, total_score)               (5)

leadership_ratings <- select(leadership, ID, mean_score)          (6)

leadership_men_high <- filter(leadership,                         (7)
                              sex == "male" & total_score > 10)   (7)
```

(1) **Загрузить пакет dplyr.**

(2) **Создать две сводные переменные.**

(3) **Перекодировать значения** `"M"` **и** `"F"` **в значения** `"male"` **и** `"female"`**.**

(4) **Переименовать переменные** `manager` **и** `gender`**.**

(5) **Отсортировать данные по переменной** `sex` **и затем каждый пол по сумме баллов.**

(6) **Создать новую таблицу данных с переменными, отражающими рейтинг.**

(7) **Создать новую таблицу данных с мужчинами, чей рейтинг выше 10.**

Этот пример сначала загружает пакет `dplyr`. Затем вызывает функцию `mutate()`, чтобы создать переменные с суммарным и средним баллом. В общем случае эта функция имеет следующий синтаксис:

```
таблица_данных <- mutate(таблица_данных,
                    новая_переменная_1 = выражение,
                    новая_переменная_2 = выражение, ...).
```

Новые переменные добавляются в таблицу данных.

Далее вызывается функция `recode()`, чтобы изменить значения переменной `gender`. В общем случае эта функция имеет следующий синтаксис:

```
вектор <- recode(вектор,
                 старое_значение_1 = новое_значение_1,
                 старое_значение_2 = новое_значение_2, ...).
```

Старые значения в векторе, для которых не определены новые значения, остаются без изменений. Например:

```
x <- c("a", "b", "c")
x <- recode(x, "a" = "apple", "b" = "banana")
x
[1] "apple" "banana" "c"
```

Для замены старых числовых значений используйте обратные кавычки:

```
> y <- c(1, 2, 3)
> y <- recode(y, `1` = 10, `2` = 15)
> y
[1] 10 15 3
```

Далее вызывается функция `rename()` для изменения имен переменных. В общем случае эта функция имеет следующий синтаксис:

```
таблица_данных <- rename(таблица_данных,
                         новое_имя_1 = "старое_имя_1",
                         новое_имя_2 = "старое_имя_2", ...).
```

Затем данные сортируются с помощью функции `arrange()`. Она сначала сортирует строки в порядке возрастания по полу (сначала женщины, потом мужчины). Затем строки сортируются в порядке возрастания по `total_score` отдельно в каждой половой группе. Чтобы изменить порядок сортировки на обратный, можно использовать функцию `desc()`. Например, инструкция

```
leadership <- arrange(leadership, sex, desc(total_score))
```

отсортирует данные в порядке возрастания по переменной `sex` и в порядке убывания по переменной `total_scores` в пределах каждого пола.

Функция `select()` используется для выбора или исключения определенных переменных. В общем случае эта функция имеет следующий синтаксис:

```
таблица_данных <- select(таблица_данных, список_переменных_1,
                         список_переменных_2, ...)
```

Списки переменных обычно содержат имена переменных без кавычек. Для выбора диапазона переменных можно использовать оператор двоеточия (`:`). Кроме того, `select()` можно использовать для выбора переменных, содержащих определенные текстовые строки. Например, инструкция

```
leadership_subset <- select(leadership,

                            ID, country:age, starts_with("q"))
```

выберет переменные `ID`, `country`, `sex`, `age`, `q1`, `q2`, `q3`, `q4` и `q5`. В справке `help(select_helpers)` вы найдете список функций, которые можно использовать для выбора переменных.

Для исключения переменных используется знак минус (-). Например, инструкция

```
leadership_subset <- select(leadership, -sex, -age)
```

включит в выборку все переменные, кроме sex и age.

Наконец, функция filter() используется для выборки наблюдений, или строк, из таблицы данных, соответствующих заданному критерию. В общем случае эта функция имеет следующий синтаксис:

```
таблица_данных <- filter(таблица_данных, выражение)
```

и извлекает строки, для которых выражение вернет TRUE. Выражение может включать любые операторы из перечисленных в табл. 3.3, а для управления порядком выполнения операторов можно использовать круглые скобки.

Следующая инструкция вернет таблицу данных со списком менеджеров мужчин, для которых средняя оценка (mean_score) меньше 2 или больше 4.

```
extreme_men <- filter(leadership,
                      sex == "male" &
                      (mean_score < 2 | mean_score > 4))
```

3.11.2. Объединение инструкций с помощью оператора конвейера

Пакет magrittr позволяет записывать код более компактно, предлагая оператор конвейера (%>%). Взгляните на следующую последовательность инструкций:

```
high_potentials <- filter(leadership, total_score > 10)
high_potentials <- select(high_potential, ID, country, mean_score)
high_potentials <- arrange(high_potential, country, mean_score)
```

Их можно уместить в одну инструкцию, применив оператор конвейера:

```
high_potentials <- filter(leadership, total_score > 10) %>%
   select(ID, country, mean_score) %>%
   arrange(country, mean_score)
```

Оператор %>% (читается как «ЗАТЕМ») передает результат вызова функции слева через первый параметр функции справа. Реорганизованные таким способом инструкции часто читаются намного проще.

Мы рассмотрели основные функции пакета dplyr, но в нем также имеются функции для обобщения, объединения и реструктуризации данных. Мы познакомимся с ними в главе 5.

3.12. Использование инструкций SQL для работы с таблицами данных

До этого момента для работы с данными использовались функции и операторы R. Однако многие аналитики, пришедшие к R, имели опыт создания запросов на языке структурированных запросов (Structured Query Language, SQL). Было бы обидно не воспользоваться этими накопленными знаниями. Поэтому я хотел бы кратко упомянуть о существовании пакета `sqldf`. (Если вы незнакомы с языком SQL, то можете спокойно пропустить этот раздел.)

Скачав и установив этот пакет (`install.packages("sqldf")`), вы сможете использовать функцию `sqldf()` для выполнения запросов `SELECT` на языке SQL к таблицам данных. В листинге 3.8 показаны два примера.

Листинг 3.8. Использование инструкций SQL для работы с таблицами данных

```
> library(sqldf)                                                    (1)
> newdf <- sqldf("select * from mtcars where carb=1 order by mpg", (1)
                 row.names=TRUE)                                    (1)
> newdf                                                             (1)
                mpg cyl  disp  hp drat   wt qsec vs am gear carb
Valiant        18.1   6 225.0 105 2.76 3.46 20.2  1  0    3    1
Hornet 4 Drive 21.4   6 258.0 110 3.08 3.21 19.4  1  0    3    1
Toyota Corona  21.5   4 120.1  97 3.70 2.46 20.0  1  0    3    1
Datsun 710     22.8   4 108.0  93 3.85 2.32 18.6  1  1    4    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.94 18.9  1  1    4    1
Fiat 128       32.4   4  78.7  66 4.08 2.20 19.5  1  1    4    1
Toyota Corolla 33.9   4  71.1  65 4.22 1.83 19.9  1  1    4    1

> sqldf("select avg(mpg) as avg_mpg, avg(disp) as avg_disp, gear    (2)
              from mtcars where cyl in (4, 6) group by gear")       (2)
  avg_mpg avg_disp gear
1    20.3      201    3
2    24.5      123    4
3    25.4      120    5
```

(1) **Выборка всех переменных (столбцов) из таблицы данных `mtcars` и наблюдений (строк), соответствующих автомобилям с одним карбюратором (`carb`), с сортировкой в порядке возрастания по значениям переменной `mpg`. Полученная выборка сохраняется в новой таблице данных `newdf`. Выражение `row.names=TRUE` требует перенести идентификаторы (имена) строк из исходной таблицы данных в новую.**

(2) **Выводит на экран средние значения переменных `mpg` и `disp` для автомобилей с коробками передач с разным числом ступеней (`gear`) и с 4- или 6-цилиндровыми моторами (`cyl`).**

Продвинутые пользователи SQL сочтут пакет `sqldf` полезным инструментом для работы с данными в R. Получить дополнительную информацию о пакете можно на странице проекта https://github.com/ggrothendieck/sqldf.

Итоги

- Создание новых и перекодирование существующих переменных – важный аспект управления данными.
- Функции позволяют хранить и манипулировать пропущенными значениями и значениями календарных дат.
- Переменные можно преобразовывать из одного типа (например, числового) в другой (например, в строку).
- Основываясь на наборе критериев, можно выбирать (или исключать из выборки) наблюдения и переменные.
- Наборы данных можно объединять по горизонтали (добавляя переменные) или по вертикали (добавляя наблюдения).

4 Начало работы с диаграммами

В этой главе:

- введение в пакет `ggplot2`;
- создание простой диаграммы двух переменных;
- группировка и разделение на категории для создания многомерных диаграмм;
- сохранение диаграмм в разных форматах.

Много раз я показывал клиентам тщательно оформленные результаты статистического анализа в виде чисел и текста, только чтобы увидеть, как у них от недоумения глаза лезут на лоб. Те же самые люди с восторгом восклицали «Ага!», когда я представлял ту же информацию в графическом виде. Часто я мог увидеть закономерности или выявить ошибки в данных, разглядывая диаграммы, – те закономерности и ошибки, которые я совершенно не замечал, выполняя более формализованный статистический анализ.

Люди прекрасно приспособлены для выявления взаимосвязей в данных, представленных в графическом виде. Имея хорошо организованные диаграммы, они способны сопоставлять

тысячи элементов информации, выявлять закономерности, которые не так-то легко обнаружить другими методами. Это одна из причин, по которым достижения в статистической графике оказали такое большое влияние на анализ данных. Исследователи должны *видеть* свои данные, и это одна из областей, где R по-настоящему блистает.

Многие годы язык R продолжал органично развиваться благодаря вкладу многих независимых разработчиков программного обеспечения. Это привело к созданию четырех различных подходов к созданию диаграмм, реализованных в виде пакетов `base`, `grid`, `ggplot2` и `grid`. В этой и во многих других главах мы сосредоточимся на `ggplot2`, самом мощном и популярном подходе, доступном в настоящее время в R.

В пакете ggplot2, созданном Хэдли Уикхэмом (Hadley Wickham, 2009a), реализована графическая система, которая основана на «графическом словаре», описанном в статье Уилкинсона (Wilkinson, 2005) и развитом в публикации Wickham (2009b). Цель этого пакета – предоставить всеобъемлющую систему, основанную на определенной грамматике, для построения диаграмм в единообразном и логическом стиле, что позволит пользователям создавать новые способы визуализации данных.

В этой главе мы обсудим основные идеи и функции для создания диаграмм с помощью `ggplot2`, помогающих визуализировать данные и находить ответы на следующие вопросы:

- Как связаны прошлый опыт работника и его заработная плата?
- Как проще всего обобщить эту связь?
- Различается ли эта связь для мужчин и женщин?
- Имеет ли значение, в какой отрасли работает работник?

Для начала мы создадим простую диаграмму рассеяния, показывающую взаимосвязь между опытом работников и заработной платой. Затем в каждом последующем разделе будем добавлять новые возможности, пока не создадим единый график, отвечающий на эти вопросы. На каждом этапе мы будем все глубже понимать имеющиеся данные.

Для получения ответов на поставленные вопросы мы используем таблицу данных CPS85, входящую в состав пакета `mosaicData`. Таблица данных содержит случайную выборку из 534 человек, отобранных в ходе обследования *Current Population Survey*, проведенного в 1985 году, которая включает информацию об их заработной плате, опыте работы и некоторые другие демографические данные. Прежде чем продолжить, обязательно установите пакеты `mosaicData` и `ggplot2` (`install.packages(c("mosaicData", "ggplot2"))`).

4.1. Создание диаграмм с помощью пакета ggplot2

Пакет `ggplot2` предлагает множество функций для построения диаграмм по слоям. Мы построим сложную диаграмму, начав с самого простого и добавляя элементы по одному. По умолчанию диаграммы `ggplot2` отображаются на сером фоне с белой координатной сеткой.

4.1.1. ggplot

Первой функцией построения диаграмм является функция `ggplot()`. Она принимает следующие аргументы:

- таблицу с данными для построения диаграммы;
- набор переменных, определяющих визуальные свойства диаграммы. Набор переменных передается в функцию `aes()` (от англ. *aesthetics* – эстетика, т. е. что-то, что можно увидеть).

Следующий код создает диаграмму, изображенную на рис. 4.1:

```
library(ggplot2)
library(mosaicData)
ggplot(data = CPS85, mapping = aes(x = exper, y = wage))
```

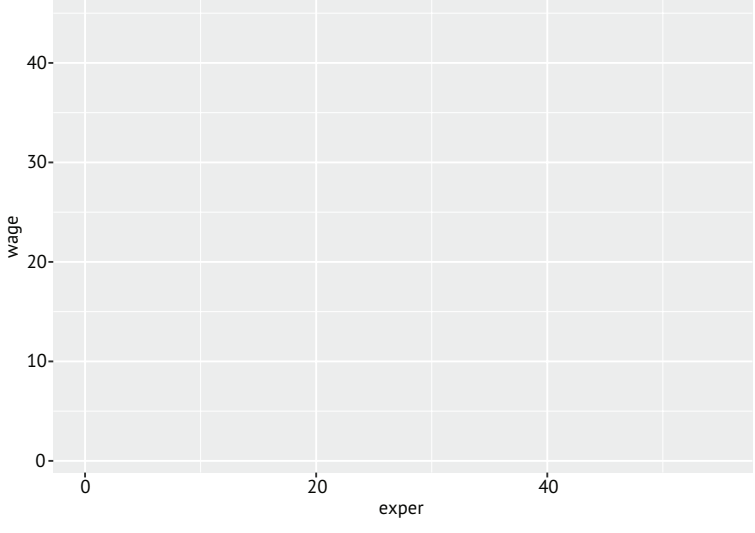


Рис. 4.1. График зависимости размера заработной платы от опыта работника

Почему диаграмма пустая? Мы указали, что переменная `exper` (опыт) должна отображаться на ось *x*, а переменная `wage` (размер заработной платы) – на ось *y*, но не указали, что именно хотим видеть на диаграмме. В данном случае нам нужны точки, представляющие каждую персону.

4.1.2. Геометрические объекты

Геометрические объекты – это объекты (точки, линии, столбики и заштрихованные области), которые можно разместить на диаграмме. Они добавляются с помощью функций с именами, начинающимися с `geom_`. В настоящее время доступно 37 различных геометрических объектов, и этот список растет. В табл. 4.1 описаны наиболее распространенные геометрические объекты, а также параметры соответствующих функций.

Таблица 4.1. Функции создания геометрических объектов

Функция	Добавляет	Параметры
`geom_bar()`	Столбиковую диаграмму	`color, fill, alpha`
`geom_boxplot()`	Коробчатую диаграмму	`color, fill, alpha, notch, width`
`geom_density()`	Диаграмму плотности	`color, fill, alpha, linetype`
`geom_histogram()`	Гистограмму	`color, fill, alpha, linetype, binwidth`
`geom_hline()`	Горизонтальные линии	`color, alpha, linetype, size`
`geom_jitter()`	Точки со смещением	`color, size, alpha, shape`
`geom_line()`	Линейный график	`color, alpha, linetype, size`
`geom_point()`	Диаграмму рассеяния	`color, alpha, shape, size`
`geom_rug()`	Ленточную диаграмму	`color, side`
`geom_smooth()`	Аппроксимирующую линию	`method, formula, color, fill, linetype, size`
`geom_text()`	Текстовую метку	`Смотрите справку по этой функции`
`geom_violin()`	Скрипичную диаграмму	`color, fill, alpha, linetype`
`geom_vline()`	Вертикальные линии	`color, alpha, linetype, size`

Добавим точки с помощью функции `geom_point()` и создадим диаграмму рассеяния. При определении диаграмм средствами `ggplot2` функции объединяются в цепочку с помощью знака `+`:

```
library(ggplot2)
library(mosaicData)
ggplot(data = CPS85, mapping = aes(x = exper, y = wage)) +
  geom_point()
```

Результат показан на рис. 4.2.

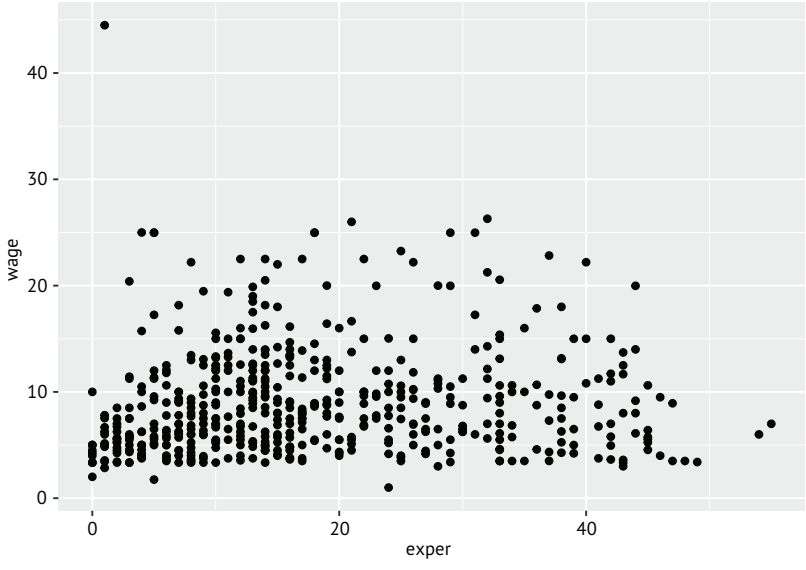


Рис. 4.2. Диаграмма рассеяния зависимости размера заработной платы от опыта работника

Похоже, что с увеличением опыта увеличивается и заработная плата, но связь довольно слабая. На диаграмме также можно заметить выбросы. У одного человека заработная плата намного выше, чем у остальных. Удалим этот случай и воспроизведем диаграмму повторно:

```
CPS85 <- CPS85[CPS85$wage < 40, ]
ggplot(data = CPS85, mapping = aes(x = exper, y = wage)) +
  geom_point()
```

Новая диаграмма показана на рис. 4.3.

Функции `geom_` принимают дополнительные параметры (табл. 4.1). Например, функция `geom_point()` принимает параметры `color`, `size`, `shape` и `alpha`. Они определяют цвет, размер, форму и прозрачность точек соответственно. Цвета можно указывать по именам или в форме шестнадцатеричного кода. Форму и тип линии (`linetype`) можно указывать по именам или номерам, определяющим узор или символ соответственно. Размер точки задается положительными действительными числами начиная с 0. Чем больше число, тем больше размер точек. Степень прозрачности варьируется от 0 (полностью прозрачный) до 1 (полностью непрозрачный). Добавление прозрачности иногда помогает визуализировать перекрывающиеся точки. Каждый из этих параметров подробно будет описан в главе 19.

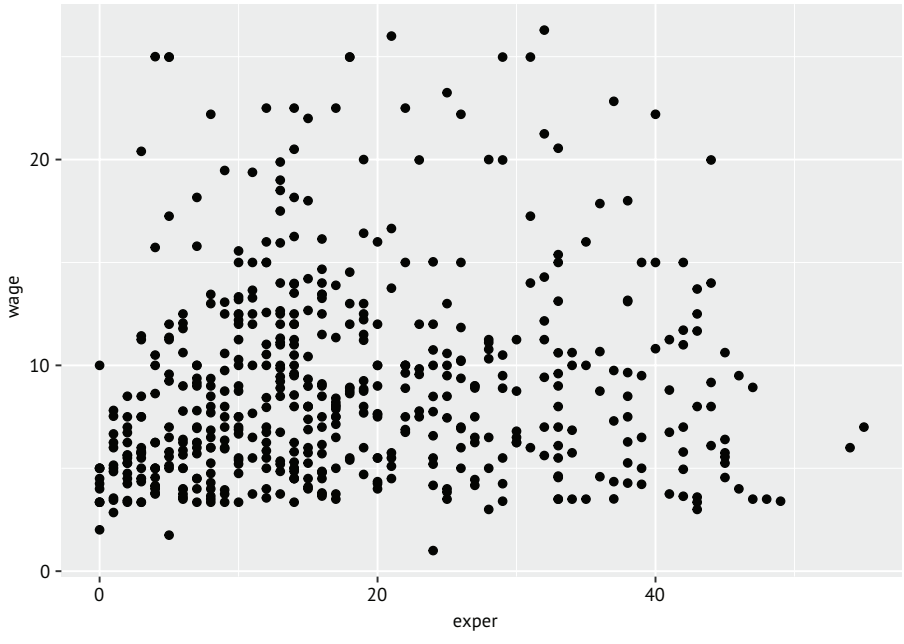


Рис. 4.3. Диаграмма рассеяния зависимости размера заработной платы от опыта работника после удаления выброса

Давайте сделаем точки на рис. 4.3 большими, полупрозрачными и синими. Кроме того, заменим серый фон белым, определив параметр темы `theme` (темы описываются в разделе 4.1.7 и в главе 19). Следующий код создаст диаграмму, изображенную на рис. 4.4:

```
ggplot(data = CPS85, mapping = aes(x = exper, y = wage)) +
  geom_point(color = "cornflowerblue", alpha = .7, size = 1.5) +
  theme_bw()
```

Диаграмма действительно получилась чуть более привлекательной (по крайней мере, вы видите ее в цвете), но она никак не улучшает понимание данных. Было бы полезно увидеть на диаграмме линию, аппроксимирующую общую тенденцию зависимости размера заработной платы от опыта.

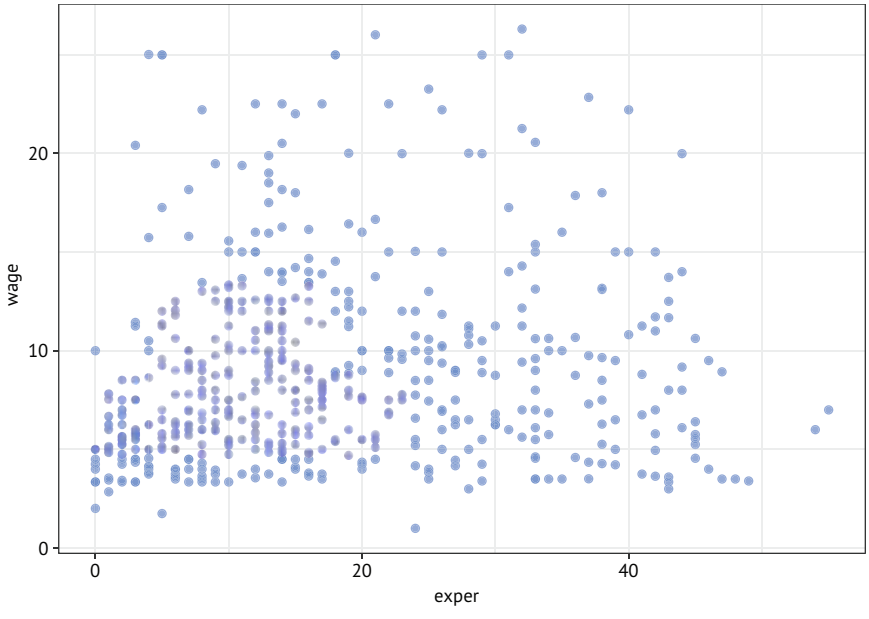


Рис. 4.4. Диаграмма рассеяния зависимости размера заработной платы от опыта работника после удаления выброса и после настройки размеров, цвета и прозрачности точек, а также применения темы bw

Добавить такую линию можно с помощью функции `geom_smooth()`. Она принимает параметры, определяющие тип (линейная, квадратичная, непараметрическая), толщину и цвет линии, а также необязательный доверительный интервал. Все эти параметры обсуждаются в главе 11. Здесь мы выводим линию линейной регрессии (`method = lm`, где `lm` означает linear model – линейная модель):

```
ggplot(data = CPS85, mapping = aes(x = exper, y = wage)) +
  geom_point(color = "cornflowerblue", alpha = .7, size = 1.5) +
  geom_smooth(method = "lm") +
  theme_bw()
```

Результат показан на рис. 4.5.

Судя по этой линии, средняя заработная плата несколько увеличивается с увеличением опыта. В этой главе используются только два геометрических объекта. Но в следующих главах мы используем многие другие и посмотрим, как с их помощью создавать самые разные диаграммы, столбиковые диаграммы, гистограммы, коробчатые диаграммы, графики плотности и др.

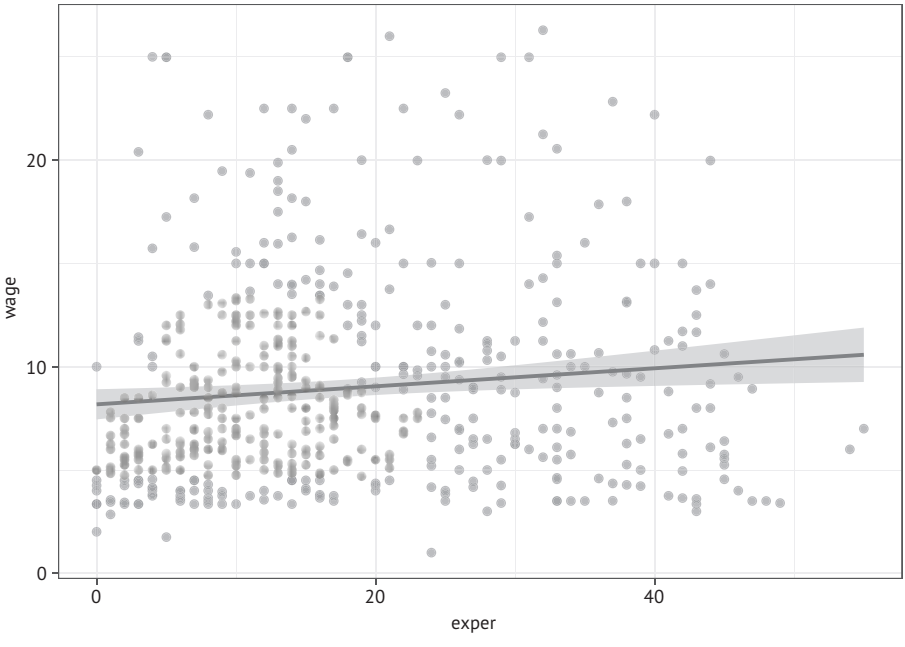


Рис. 4.5. Диаграмма рассеяния зависимости размера заработной платы от опыта работника с аппроксимирующей линией

4.1.3. Группировка

В предыдущем разделе мы настроили такие характеристики диаграммы, как цвет и прозрачность, используя *постоянное* значение. Однако цвет, форму, размер, прозрачность, стиль линий и другие визуальные характеристики можно поставить в зависимость от значений переменных. Это позволяет изображать группы наблюдений на одном графике (это называется группировкой).

Давайте добавим в диаграмму переменную `sex` и используем ее для управления цветом, формой и типом линий:

```
ggplot(data = CPS85,
       mapping = aes(x = exper, y = wage,
                     color = sex, shape = sex, linetype = sex)) +
  geom_point(alpha = .7, size = 1.5) +
  geom_smooth(method = "lm", se = FALSE, size = 1.5) +
  theme_bw()
```

По умолчанию первая группа (женщины) представлена кружками с розовой заливкой и сплошной розовой линией, а вторая группа (мужчины) – треугольниками с бирюзовой заливкой и пунктирной бирюзовой линией. Новая диаграмма показана на рис. 4.6.

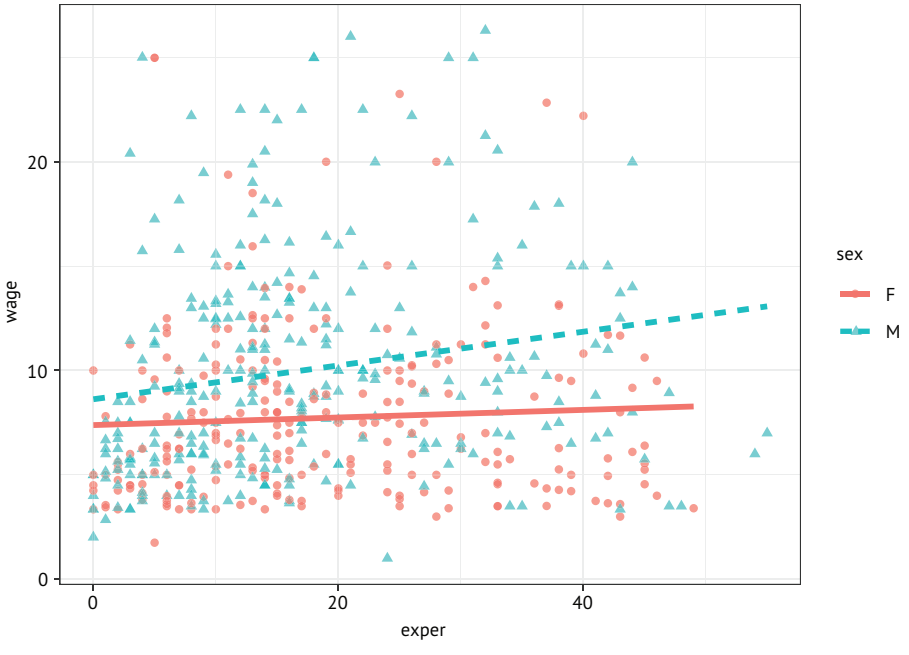


Рис. 4.6. Диаграмма рассеяния зависимости размера заработной платы от опыта работника с разделением по полу и отдельными аппроксимирующими линиями для мужчин и женщин

Обратите внимание, что параметры `color=sex`, `shape=sex` и `linetype=sex` помещены в функцию `aes()`, потому что выбранная нами переменная управляет эстетикой (внешним видом). Параметр `se = FALSE` в вызове `geom_smooth` был добавлен для подавления доверительных интервалов, что делает график не таким перегруженным и более удобным для восприятия. Параметр `size = 1.5` немного увеличивает толщину линий.

Упрощение диаграмм

В общем случае мы должны стремиться создавать как можно более простые диаграммы, точно передающие информацию, содержащуюся в данных. В реальной жизни, подготавливая диаграмму из предыдущего примера, я, скорее всего, поставил в зависимость от переменной `sex` только цвет. Добавление зависимости для формы и типа линии делает диаграмму излишне перегруженной. Я добавил эти зависимости, только чтобы создать диаграмму, которую одинаково легко было бы читать как в цветном (электронном), так и в черно-белом (печатном) исполнении этой книги.

Теперь отчетливо видно, что мужчины, как правило, зарабатывают больше, чем женщины (верхняя линия). Кроме того, связь опыта с размером заработной платы у мужчин явно более сильная (более крутая линия), чем у женщин.

4.1.4. Масштабирование

Как мы видели, функция `aes()` используется для сопоставления переменных с визуальными характеристиками диаграммы. Масштабирование помогает определить, как отображаются данные. Например, `ggplot2` автоматически создает оси графика с засечками, метками засечек и метками осей. Обычно он прекрасно справляется с этой задачей, но иногда бывает желательно подправить их внешний вид. Цвета, представляющие группы, выбираются автоматически, но при желании и в зависимости от ваших вкусов и требований к публикации можно выбрать другой набор цветов.

Функции масштабирования (с именами, начинающимися с `scale_`) позволяют изменить масштабирование по умолчанию. В табл. 4.2 перечислены некоторые часто используемые функции масштабирования.

Таблица 4.2. Некоторые часто используемые функции масштабирования

Функция	Описание
`scale_x_continuous()`, `scale_y_continuous()`	Масштабирует оси *x* и *y* для количественных переменных. Принимает параметры, управляющие визуальным представлением засечек, меток и диапазоном отображаемых значений
`scale_x_discrete()`, `scale_y_discrete()`	То же, что и выше, но для осей, представляющих категориальные переменные
`scale_color_manual()`	Задает цвета для представления уровней категориальной переменной. Параметр `values` определяет цвета. Таблицу цветов можно найти в http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

Давайте изменим масштабы осей *x* и *y*, а также цвета, представляющие мужчин и женщин. Для оси *x*, представляющей опыт (`exper`), зададим диапазон от 0 до 60 с шагом 10, а для оси *y*, представляющей размер заработной платы, зададим диапазон от 0 до 30 с шагом 5. Женщины будут представлены грязно-красным цветом, а мужчины – грязно-синим. Следующий код создает диаграмму, изображенную на рис. 4.7:

```
ggplot(data = CPS85,
       mapping = aes(x = exper, y = wage,
                     color = sex, shape=sex, linetype=sex)) +
   geom_point(alpha = .7, size = 3) +
   geom_smooth(method = "lm", se = FALSE, size = 1.5) +
   scale_x_continuous(breaks = seq(0, 60, 10)) +
   scale_y_continuous(breaks = seq(0, 30, 5)) +
   scale_color_manual(values = c("indianred3", "cornflowerblue")) +
 theme_bw()
```

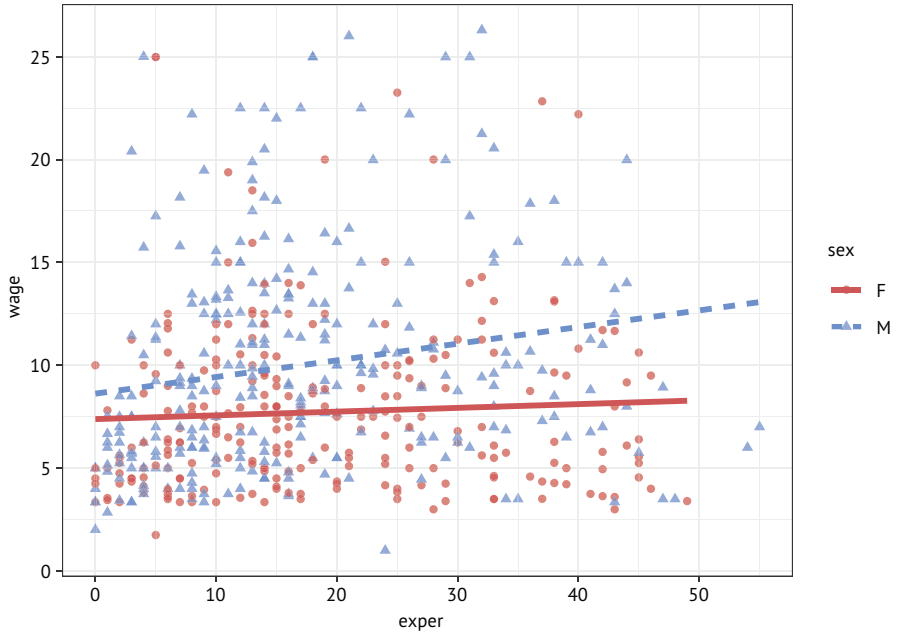


Рис. 4.7. Диаграмма рассеяния зависимости размера заработной платы от опыта работника с разделением по полу и измененными цветами и масштабом по осям *x* и *y*

Засечки определяются вектором значений. Здесь используется функция `seq()`, упрощающая создание такого вектора. Например, `seq(0, 60, 10)` генерирует числовой вектор, начиная с 0 и заканчивая 60 с шагом 10.

Оси *x* и *y* теперь кажутся нагляднее, а цвета – привлекательнее (на мой взгляд). Однако зарплата указана в долларах. Мы можем изменить метки на оси *y*, чтобы отметить этот факт, используя пакет `scales`, позволяющий форматировать метки и отображать метки, включающие знак доллара, евро, процента и т. д.

Установите пакет `scales` (`install.packages("scales")`), а затем выполните следующий код:

```
ggplot(data = CPS85,
       mapping = aes(x = exper, y = wage,
                                color = sex, shape=sex, linetype=sex)) +
    geom_point(alpha = .7, size = 3) +
    geom_smooth(method = "lm", se = FALSE, size = 1.5) +
    scale_x_continuous(breaks = seq(0, 60, 10)) +
    scale_y_continuous(breaks = seq(0, 30, 5),
                       label = scales::dollar) +
    scale_color_manual(values = c("indianred3", "cornflowerblue")) +
  theme_bw()
```

Результат показан на рис. 4.8.

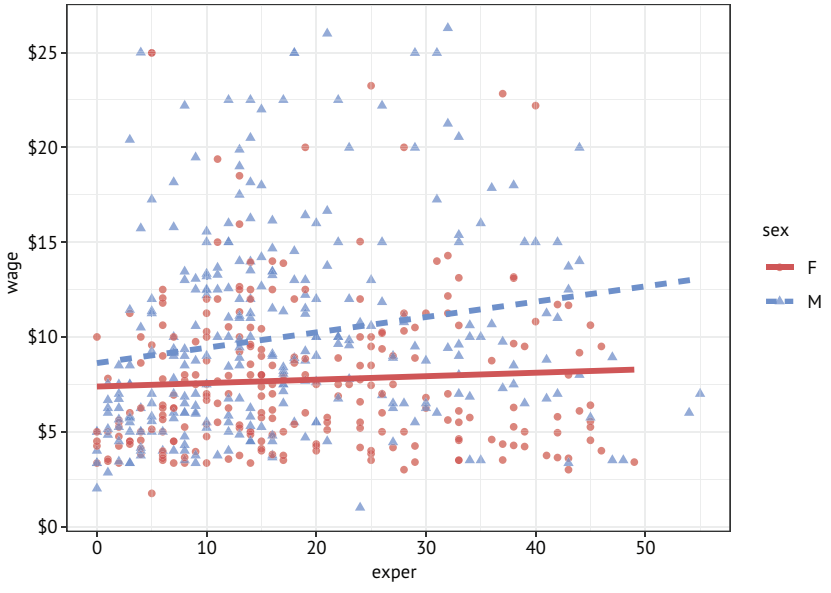


Рис. 4.8. Диаграмма рассеяния зависимости размера заработной платы от опыта работника с разделением по полу, измененными цветами и масштабом по осям *x* и *y*. Метки на оси *y* подсказывают, что зарплата измеряется в долларах

Мы неплохо продвинулись вперед. Теперь перейдем к следующему вопросу: различается ли связь между опытом и размером заработной платы для мужчин и женщин и для разных отраслей? Давайте построим ту же диаграмму отдельно для каждой отрасли.

4.1.5. Категоризованные диаграммы

Иногда взаимосвязи становятся более отчетливыми, если отобразить группы на параллельных диаграммах. Категоризованные диаграммы позволяют воспроизвести отдельный график для каждого значения заданной переменной (или их комбинации). Категоризованные диаграммы можно создавать с помощью функций `facet_wrap()` и `facet_grid()`. Их синтаксис приводится в табл. 4.3, где `var`, `rowvar` и `colvar` являются факторами.

Таблица 4.3. Функции создания категоризованных диаграмм в пакете ggplot2

Функция	Описание
`facet_wrap(~var, ncol=n)`	Создает отдельную диаграмму для каждого значения переменной `var`, размещая их в `n` столбцах
`facet_wrap(~var, nrow=n)`	Создает отдельную диаграмму для каждого значения переменной `var`, размещая их в `n` строках

Окончание табл. 4.3

Функция	Описание
`facet_grid(rowvar~colvar)`	Создает отдельную диаграмму для каждого значения комбинации переменных `rowvar` и `colvar`, где `rowvar` представляет строки, а `colvar` – столбцы
`facet_grid(rowvar~.)`	Создает отдельную диаграмму для каждого значения переменной `rowvar` в одном столбце
`facet_grid(.~colvar)`	Создает отдельную диаграмму для каждого значения переменной `colvar` в одной строке

В нашем примере категории будут определяться восемью значениями переменной `sector`. Поскольку каждая отдельная диаграмма получится меньше, мы опустим параметр `size=3` в вызове `geom_point()` и `size=1.5` в вызове `geom_smooth()`. Это уменьшит размеры точек и толщину линий и сделает диаграммы более удобочитаемыми. Следующий код создает диаграмму, изображенную на рис. 4.9:

```
ggplot(data = CPS85,
       mapping = aes(x = exper, y = wage,
                     color = sex, shape = sex, linetype = sex)) +
  geom_point(alpha = .7) +
  geom_smooth(method = "lm", se = FALSE) +
  scale_x_continuous(breaks = seq(0, 60, 10)) +
  scale_y_continuous(breaks = seq(0, 30, 5),
                     label = scales::dollar) +
  scale_color_manual(values = c("indianred3", "cornflowerblue")) +
  facet_wrap(~sector) +
  theme_bw()
```

Похоже, что различия между мужчинами и женщинами по-разному проявляются в разных отраслях. Например, сильная положительная связь между опытом и заработной платой имеет место для менеджеров-мужчин, но не для менеджеров-женщин. В меньшей степени это верно и в торговле. По всей видимости, в сфере обслуживания заработная плата никак не зависит от опыта ни у мужчин, ни у женщин, однако мужчины зарабатывают немного больше. Заработная плата увеличивается с опытом у женщин-бухгалтеров, но может снижаться у мужчин (в данном случае зависимость может быть несущественной). На данный момент мы получили более глубокое понимание зависимости между опытом и размером заработной платы.

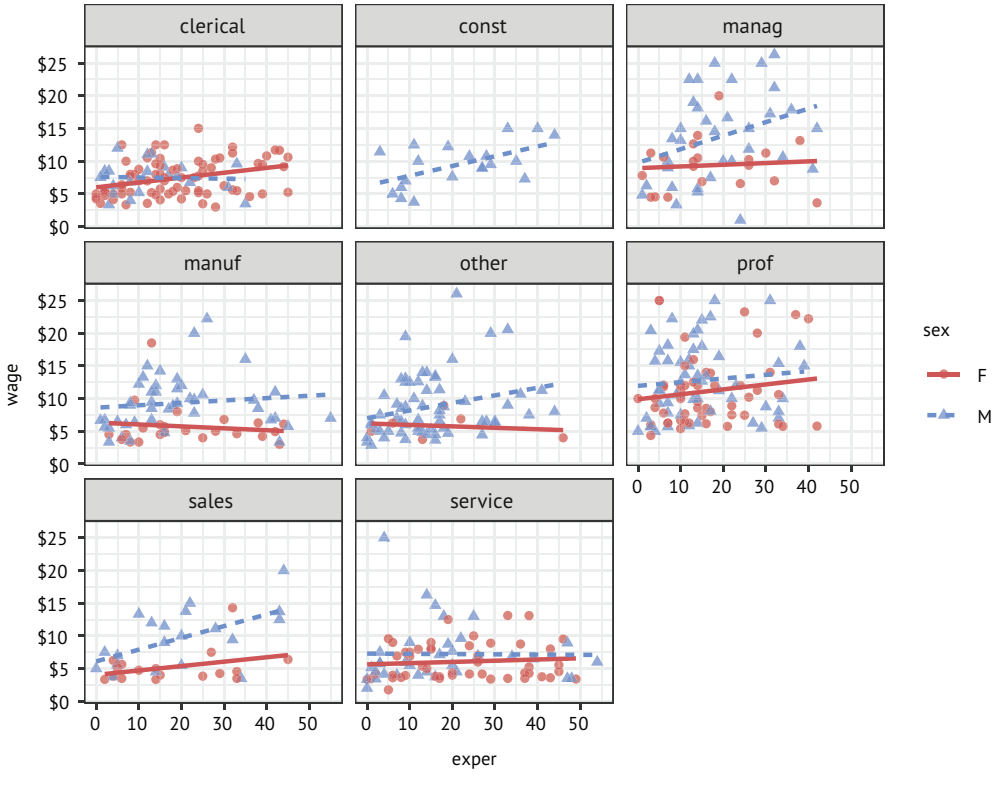


Рис. 4.9. Диаграмма рассеяния зависимости размера заработной платы от опыта работника с разделением по полу, измененными цветами и масштабом по осям *x* и *y*. Зависимость для каждой отрасли отображена на отдельной диаграмме

4.1.6. Метки

Диаграммы должны легко интерпретироваться, и информативные метки здорово помогают в этом. Функция `labs()` позволяет настроить метки для осей и легенд. Также с ее помощью можно добавить свой заголовок, подзаголовок и подпись внизу. Давайте добавим все эти метки, как показано ниже:

```
ggplot(data = CPS85,
       mapping = aes(x = exper, y = wage,
                     color = sex, shape=sex, linetype=sex)) +
    geom_point(alpha = .7) +
    geom_smooth(method = "lm", se = FALSE) +
    scale_x_continuous(breaks = seq(0, 60, 10)) +
    scale_y_continuous(breaks = seq(0, 30, 5),
                       label = scales::dollar) +
    scale_color_manual(values = c("indianred3",
                                  "cornflowerblue")) +
    facet_wrap(~sector) +
```

```
  labs(title = "Relationship between wages and experience",
     subtitle = "Current Population Survey",
     caption = "source: http://mosaic-web.org/",
     x = " Years of Experience",
     y = "Hourly Wage",
     color = "Gender", shape = "Gender", linetype = "Gender") +
theme_bw()
```

Результат показан на рис. 4.10.

Рис. 4.10. Диаграмма рассеяния зависимости размера заработной платы от опыта работника с отдельными графиками для разных отраслей и дополнительными метками

Теперь не нужно гадать, что означают метки `expr` и `wage` или откуда взяты данные.

4.1.7. Темы

Наконец, внешний вид диаграммы можно точно настроить с помощью тем. Функции управления темами (с именами, начинающимися с `theme_`) позволяют задавать цвет фона, шрифта, линий сетки, размещение легенды и другое видимое содержимое диаграмм, не связанное с данными. Давайте использовать наиболее чистую тему. Начиная с примера, изображенного на рис. 4.4, мы использовали тему с белым фоном и светло-серыми линиями сетки. Теперь попробуем другую тему, более минималистическую. Следующий код создает диаграмму, представленную на рис. 4.11:

```
ggplot(data = CPS85,
       mapping = aes(x = exper, y = wage,
                     color = sex, shape=sex, linetype=sex)) +
    geom_point(alpha = .7) +
    geom_smooth(method = "lm", se = FALSE) +
    scale_x_continuous(breaks = seq(0, 60, 10)) +
    scale_y_continuous(breaks = seq(0, 30, 5),
                       label = scales::dollar) +
    scale_color_manual(values = c("indianred3",
                                  "cornflowerblue")) +
    facet_wrap(~sector) +
    labs(title = "Relationship between wages and experience",
       subtitle = "Current Population Survey",
       caption = "source: http://mosaic-web.org/",
       x = " Years of Experience",
       y = "Hourly Wage",
       color = "Gender", shape = "Gender", linetype = "Gender") +
  theme_minimal()
```

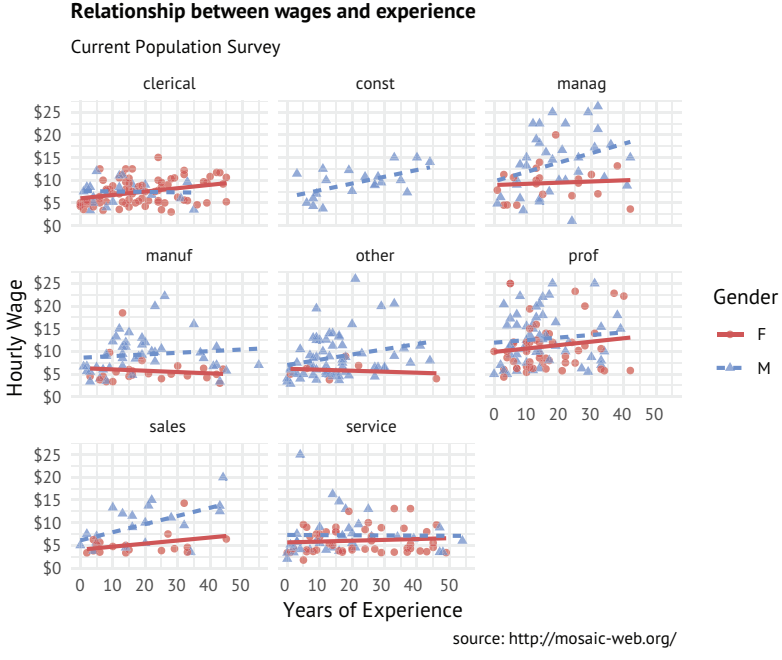


Рис. 4.11. Диаграмма рассеяния зависимости размера заработной платы от опыта работника с отдельными графиками для разных отраслей и дополнительными метками, оформленная с использованием более чистой темы

Эта диаграмма готова к публикации. Конечно, аналитические выводы условны, потому что основаны на выборке ограниченного размера и не включают оценку статистической значимости. Соответствующие проверки для получения таких оценок будут описаны в главе 8, а более подробное описание тем вы найдете в главе 19.

4.2. Особенности пакета ggplot2

Прежде чем закончить эту главу, необходимо рассмотреть три важных вопроса: применение функции `aes()`, работа с диаграммами ggplot2 как с объектами R и различные методы сохранения графиков для использования в отчетах и на веб-страницах.

4.2.1. Параметры с данными и настройками визуального представления

Создание диаграмм с помощью `ggplot2` всегда начинается с вызова функции `ggplot()`. В предыдущих примерах этой функции передавались параметры с данными и настройками визуального представления. В таком случае они применяются ко всем последующим вызовам функций `geom`.

Однако эти параметры можно передавать непосредственно в вызовы функций `geom`. В этом случае они применяются только к конкретному геометрическому объекту. Рассмотрим следующий фрагмент кода:

```
ggplot(CPS85, aes(x = exper, y = wage, color = sex)) +
          geom_point(alpha = .7, size = 1.5) +
          geom_smooth(method = "lm", se = FALSE, size = 1)  +
          scale_color_manual(values = c("lightblue", "midnightblue")) +
          theme_bw()
```

Он сгенерирует диаграмму, изображенную на рис. 4.12.

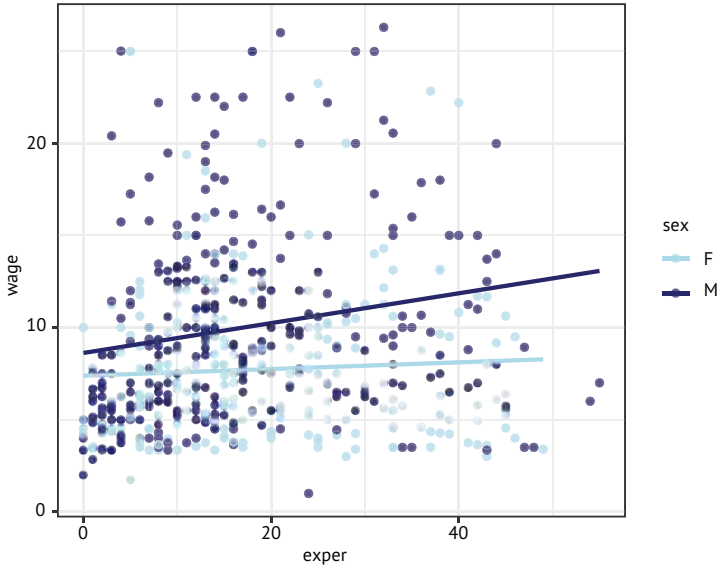


Рис. 4.12. Диаграмма рассеяния зависимости размера заработной платы от опыта работника с разделением по полу, где aes(color=sex) находится в вызове `ggplot()`. Параметры, управляющие визуальным представлением, применяются к обеим функциям, `geom_point()` и `geom_smooth()`, что обеспечивает окрашивание в разные цвета точек и аппроксимирующих линий, соответствующих мужчинам и женщинам

Поскольку зависимость цвета от переменной `sex` определяется в функции `ggplot()`, она применяется также к вызовам обеих функций, `geom_point` и `geom_smooth`. Цвет точки указывает на пол, а кроме того, для мужчин и женщин создаются отдельные цветные аппроксимирующие линии, отражающие тренд. Вот другой фрагмент кода для сравнения:

```
ggplot(CPS85,   aes(x = exper, y = wage)) +
          geom_point(aes(color = sex), alpha = .7, size = 1.5) +
          geom_smooth(method = "lm", se = FALSE, size = 1) +
          scale_color_manual(values = c("lightblue", "midnightblue")) +
          theme_bw()
```

Получившаяся диаграмма показана на рис. 4.13.

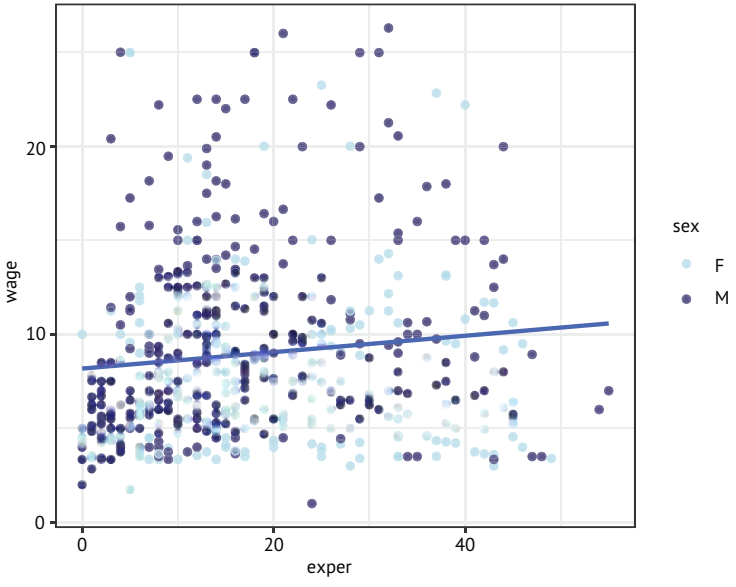


Рис. 4.13. Диаграмма рассеяния зависимости размера заработной платы от опыта работника с разделением по полу, где aes(color=sex) находится в вызове `geom_point()`. Параметры, управляющие визуальным представлением, в этом случае влияют только на цвет точек, но не применяются к вызову geom_smooth(), поэтому на диаграмме отображается лишь одна аппроксимирующая линия, соответствующая тренду для всех работников без разделения по полу

Поскольку зависимость цвета от пола задается только в функции `geom_point()`, она применяется лишь там, а `geom_smooth()` выводит единую для всех наблюдений линию аппроксимации.

В большинстве примеров в этой книге параметры с данными и настройками визуального представления помещаются в вызов функции `ggplot()`. Кроме того, имена `data=` и `mapping=` в большинстве случаев будут опускаться, потому что первый параметр всегда представляет данные, а второй – настройки визуального представления.

4.2.2. Диаграммы как объекты

Диаграмму `ggplot2` можно сохранить как именованный объект R (список), выполнить с ним некоторые операции, а затем распечатать или сохранить на диск. Взгляните на код в листинге 4.1.

Листинг 4.1. Использование диаграммы ggplot2 в роли объекта

```
data(CPS85 , package = "mosaicData")                              (1)
CPS85 <- CPS85[CPS85$wage < 40,]                                  (1)

myplot <- ggplot(data = CPS85,                                    (2)
            aes(x = exper, y = wage)) +                           (2)
       geom_point()                                               (2)

myplot                                                            (3)

myplot2 <- myplot + geom_point(size = 3, color = "blue")          (4)
myplot2                                                           (4)

myplot + geom_smooth(method = "lm") +                             (5)
  labs(title = "Mildly interesting graph")                        (5)
```

(1) **Подготовка данных.**

(2) **Создание диаграммы рассеяния и сохранение в объекте** `myplot`**.**

(3) **Отображение** `myplot`**.**

(4) **Увеличение размеров точек и окрашивание их в синий цвет, сохранение полученного результата в виде объекта** `myplot2` **и его отображение.**

(5) **Отображение диаграммы** `myplot` **с дополнительной аппроксимирующей линией тренда и заголовком.**

Сначала выполняются импортирование данных и удаление выбросов. Затем создается простая диаграмма рассеяния зависимости размера заработной платы от опыта, которая сохраняется как объект `myplot`. Далее диаграмма выводится на экран. Затем в ней изменяются размер и цвет точек, результат сохраняется как `myplot2` и отображается на экране. Наконец, в исходную диаграмму добавляются линия аппроксимации и заголовок, и полученный результат отображается на экране. Обратите внимание, что эти последние изменения не сохраняются.

Возможность сохранять диаграммы в виде объектов позволяет продолжить работу с ними и вносить изменения. Она может помочь сэкономить массу времени (и избежать запястного синдрома), а также пригодится для сохранения диаграмм программным способом, как будет показано в следующем разделе.

4.2.3. Сохранение диаграмм

Диаграммы, созданные средствами `ggplot2`, можно сохранять, используя графический интерфейс RStudio или свой программный код. Чтобы сохранить график с помощью меню RStudio, перейдите на вкладку Plots (Диаграммы) и распахните раскрывающийся список Export (Экспорт), как показано на рис. 4.14.

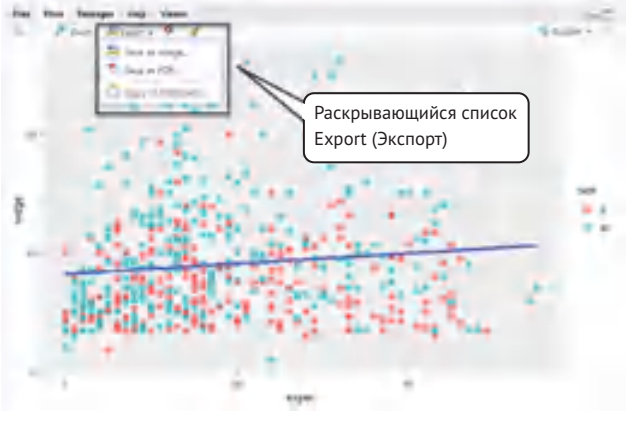


Рис. 4.14. Сохранение диаграммы в RStudio

Диаграммы также можно сохранять программно, с помощью функции `ggsave()`. Функции можно передать объект диаграммы, ее размер и формат, а также путь к месту для его сохранения. Например:

```
ggsave(file="mygraph.png", plot=myplot, width=5, height=4)
```

Этот вызов сохранит `myplot` в формате PNG размером 5"×4" в файл с именем *mygraph.png* в текущем рабочем каталоге. При желании диаграмму можно сохранить в другом формате, изменив расширение файла. В табл. 4.4 перечислены некоторые наиболее часто используемые форматы

Таблица 4.4. Форматы файлов изображений

Расширение	Формат
pdf	Portable Document Format (переносимый формат документов)
jpeg	JPEG
ti	Tagged Image File Format (формат файлов изображений с признаками)
png	Portable Network Graphics (переносимая сетевая графика)
svg	Scalable Vector Graphics (масштабируемая векторная графика)
wmf	Windows Metafile (метафайл среды Windows)

Форматы PDF, SVG и WMF являются векторными форматами – они могут изменять размеры без появления эффектов размытости или пикселизации. Другие форматы являются растровыми изображениями – при изменении их размера проявляется эффект пикселизации. Особенно это заметно при увеличении маленьких изображений. Формат PNG широко используется для изображений, отображаемых в веб-страницах. Форматы JPEG и TIF обычно используются для фотографий.

Формат WMF рекомендуется для диаграмм, которые будут отображаться в документах Microsoft Word или PowerPoint. MS Office не поддерживает файлы PDF или SVG, а формат WMF хорошо масштабируется. Однако формат WMF не поддерживает настройки прозрачности.

Если опустить параметр `plot=`, то будет сохранен последний созданный график. Следующий код сохраняет график на диск в виде документа PDF:

```
ggplot(data=mtcars, aes(x=mpg)) + geom_histogram()
ggsave(file="mygraph.pdf")
```

Дополнительные подробности ищите в справке `help(ggsave)`.

4.2.4. Типичные ошибки

Поработав с `ggplot2` в течение многих лет, я обнаружил, что часто допускаю две ошибки. Первая – пропуск или неправильная установка закрывающей скобки. Чаще всего это происходит после функции `aes()`. Взгляните на следующий код:

```
ggplot(CPS85, aes(x = exper, y = wage, color = sex) +
  geom_point()
```

Обратите внимание на отсутствие закрывающей скобки в конце первой строки. Я не могу сказать, сколько раз совершал эту ошибку.

Вторая ошибка – путаница присваивания с отображением. Следующий код создает диаграмму, изображенную на рис. 4.15:

```
ggplot(CPS85, aes(x = exper, y = wage, color = "blue")) +
  geom_point()
```

Функция `aes()` используется для *группировки*, она применяет назначенные переменные для определения визуальных характеристик диаграммы. Присваивание постоянных значений должно производиться вне функции `aes()`. Вот как должен выглядеть правильный код:

```
ggplot(CPS85, aes(x = exper, y = wage) +
  geom_point(color = "blue")
```

Точки выводятся красным цветом (а не синим), и что за странная легенда (рис. 4.15)?

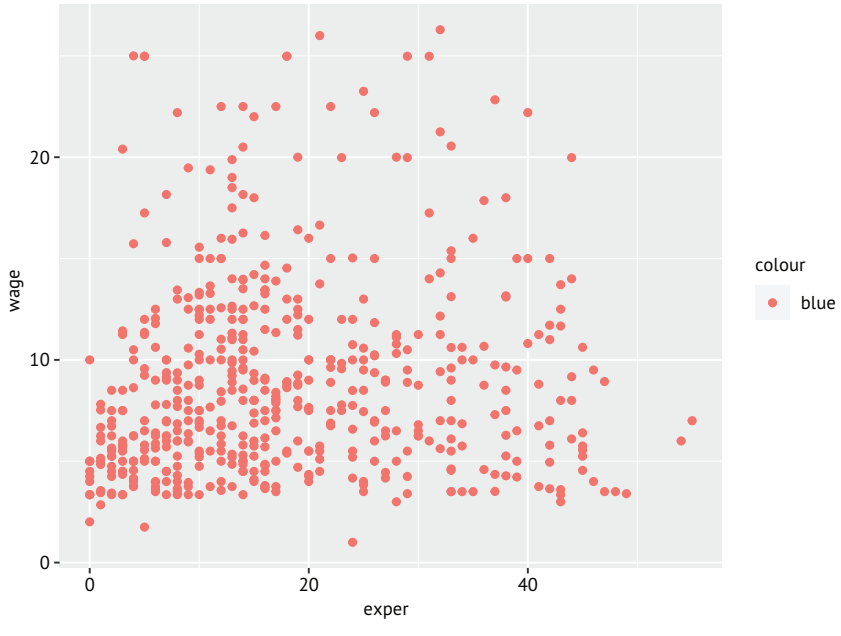


Рис. 4.15. Результат помещения инструкции присваивания в вызов функции aes()

Итоги

- Пакет `ggplot2` предоставляет язык и грамматику для создания диаграмм, отображающих данные.
- Диаграммы рассеяния описывают взаимосвязь между двумя количественными переменными. Для обобщения взаимосвязи в диаграммы можно добавлять аппроксимирующие линии (линии тренда).
- Для представления групп наблюдений можно использовать цвет, форму и размер.
- Категориальные диаграммы удобны для построения графиков, отображающих зависимости в нескольких группах.
- Диаграммы можно настраивать, определяя масштабы, метки и темы.
- Диаграммы можно сохранять во многих форматах.

5 Дополнительные приемы управления данными

В этой главе:

- математические и статистические функции;
- текстовые функции;
- циклы и условные операторы;
- пользовательские функции;
- агрегирование и преобразование данных.

В главе 3 мы рассмотрели основные способы управления наборами данных в R. В этой главе мы сосредоточимся на более сложных подходах. Глава делится на три основные части. В первой части мы кратко познакомимся с многочисленными функциями, выполняющими математические и статистические вычисления, а также преобразование текстовых значений. Для придания этой части большей актуальности мы начнем с описания задачи по преобразованию данных, которую можно решить с использованием этих функций. После знакомства с функциями мы рассмотрим одно из возможных решений данной задачи.

Затем мы поговорим о том, как писать свои собственные функции для управления данными и их анализа. Сначала мы исследу-

ем способы управления потоком выполнения инструкций с использованием циклов и условных операторов. Потом познакомимся со структурой пользовательских функций и узнаем, как их применять.

Наконец, мы рассмотрим способы агрегирования, обобщения и преобразования наборов данных. При агрегировании данных можно использовать любые встроенные или написанные пользователем функции, поэтому знания, полученные в первых двух разделах этой главы, вам действительно пригодятся.

5.1. Задача по управлению данными

Прежде чем начинать обсуждение числовых и текстовых функций, определим задачу, которую затем будем решать. Группа студентов сдавала экзамены по математике, естественным наукам и английскому языку. Полученные ими баллы по трем предметам нужно объединить и получить единый показатель успеваемости для каждого студента. Кроме того, необходимо поставить оценку `A` первым по успеваемости 20 % студентов, оценку `B` – следующим по успеваемости 20 % и т. д. Наконец, список студентов нужно отсортировать в алфавитном порядке. Данные представлены в табл. 5.1.

Таблица 5.1. Результаты студенческих экзаменов

Студент	Математика	Естественные науки	Английский язык
John Davis	502	95	25
Angela Williams	600	99	22
Bullwinkie Moose	412	80	18
David Jones	358	82	15
Janice Markhammer	495	75	20
Cheryl Cushing	512	85	28
Reuven Ytzrhak	410	80	15
Greg Knox	625	95	30
Joel England	573	89	27
Mary Rayburn	522	86	18

При взгляде на эти данные сразу можно заметить несколько проблем. Во-первых, баллы, полученные на экзаменах по разным предметам, несопоставимы между собой. Их средние значения и стандартные отклонения сильно различаются, поэтому усреднять их не имеет смысла. Для вычисления единого показателя успеваемости

необходимо преобразовать эти баллы так, чтобы их можно было сопоставлять между собой. Во-вторых, нам понадобится метод для определения мест студентов в общем рейтинге успеваемости, чтобы поставить им итоговую оценку. В-третьих, для нормальной сортировки студентов в алфавитном порядке нужно разбить первый столбец на два – с именем и фамилией.

Все перечисленные проблемы можно устранить с помощью числовых и текстовых функций в R. После знакомства с функциями в следующем разделе мы сможем найти подходящее решение для нашей задачи по управлению данными.

5.2. Числовые и текстовые функции

В этом разделе перечисляются функции R, которые можно использовать для управления данными. Их можно разделить на числовые (математические, статистические, вероятностные) и текстовые. После знакомства с функциями обоих типов я покажу вам, как применять их к столбцам (переменным) и строкам (наблюдениям) таблиц данных (раздел 5.2.6).

5.2.1. Математические функции

В табл. 5.2 перечислены наиболее распространенные математические функции вместе с короткими примерами.

Таблица 5.2. Математические функции

Функция	Описание
`abs(x)`	Абсолютное значение `abs(-4)` вернет 4
`sqrt(x)`	Квадратный корень `sqrt(25)` вернет 5 То же самое, что и `25^(0.5)`
`ceiling(x)`	Ближайшее целое число, не меньшее, чем *x* `ceiling(3.457)` вернет 4
`floor(x)`	Ближайшее целое число, не большее, чем *x* `floor(3.457)` вернет 3
`trunk(x)`	Целое число, полученное округлением *x* в сторону нуля `trunk(5.99)` вернет 5
`round(x, digits=n)`	Округляет *x* до заданного числа знаков *n* после запятой `round(3.475, digits=2)` вернет 3.48
`signif(x, digits=n)`	Округляет *x* до заданного числа *n* значащих цифр `signif(3.475, digits=2)` вернет 3.5
`cos(x), sin(x), tan(x)`	Косинус, синус и тангенс `cos(2)` вернет -0.416

Окончание табл. 5.2

Функция	Описание
`acos(x)`, `asin(x)`, `atan(x)`	Арккосинус, арксинус и арктангенс `acos(-0.416)` вернет `2`
`cosh(x)`, `sinh(x)`, `tanh(x)`	Гиперболические косинус, синус и тангенс `sinh(2)` вернет `3.627`
`acosh(x)`, `asinh(x)`, `atanh(x)`	Гиперболические арккосинус, арксинус и арктангенс `asinh(3.627)` вернет `2`
`log(x, base=n)` `log(x)` `log10(x)`	Логарифм *x* по основанию *n* Для удобства: `log(x)` – натуральный логарифм `log10(x)` – десятичный логарифм `log(10)` вернет `2.3026` `log10(10)` вернет `1`
`exp(x)`	Экспоненциальная функция `exp(2.3026)` вернет `10`

В основном эти функции применяются для преобразования данных. К примеру, данные с положительно асимметричным распределением перед дальнейшей обработкой обычно логарифмируют. Математические функции также используют при реализации вычислений по формулам, создании графиков (например, кривая зависимости `x` от `sin(x)`) и форматировании числовых значений перед выводом на экран.

В табл. 5.2 показаны примеры применения математических функций к *скалярам* (отдельным числам). Когда эти функции применяются к числовым векторам, матрицам или таблицам данных, они преобразуют каждое число отдельно. Например, `sqrt(c(4, 16, 25))` вернет вектор `c(2, 4, 5)`.

5.2.2. Статистические функции

Самые распространенные статистические функции перечислены в табл. 5.3. Многие из них принимают дополнительные параметры, влияющие на результат. Например,

```
y <- mean(x)
```

вычислит среднее арифметическое по всем элементам объекта `x`, а

```
z <- mean(x, trim = 0.05, na.rm=TRUE)
```

вычислит усеченное среднее, исключив 5 % наибольших и 5 % наименьших значений в выборке, а также пропущенные значения. Используйте `help()`, чтобы узнать больше о каждой функции и ее аргументах.

Таблица 5.3. Статистические функции

Функция	Описание
`mean(x)`	Среднее арифметическое `mean(c(1,2,3,4))` вернет `2.5`
`median(x)`	Медиана `median(c(1,2,3,4))` вернет `2.5`
`sd(x)`	Стандартное отклонение `sd(c(1,2,3,4))` вернет `1.29`
`var(x)`	Дисперсия `var(c(1,2,3,4))` вернет `1.67`
`mad(x)`	Абсолютное отклонение медианы `mad(c(1,2,3,4))` вернет `1.48`
`quantile(x, probs)`	Квантили, где *x* – числовой вектор, для которого вычисляются квантили, а *probs* – числовой вектор с вероятностями в диапазоне [0; 1] # 30-й и 84-й процентили *x* y <- quantile(x, c(.3,.84))
`range(x)`	Размах значений x <- c(1,2,3,4) `range(x)` вернет `c(1,4)`. `diff(range(x))` вернет `3`
`sum(x)`	Сумма `sum(c(1,2,3,4))` вернет `10`
`diff(x, lag=n)`	Разности между значениями в выборке, взятыми с заданным интервалом *lag*. По умолчанию интервал равен 1. x <- c(1,5,23,29) `diff(x)` вернет `c(4, 18, 6)`
`min(x)`	Минимум `min(c(1,2,3,4))` вернет `1`
`max(x)`	Максимум `max(c(1,2,3,4))` вернет `4`
`scale(x, center=TRUE, scale=TRUE)`	Значения объекта *x*, центрированные (`center=TRUE`) или стандартизированные (`center=TRUE, scale=TRUE`) по столбцам. Пример использования приводится в листинге 5.6

Чтобы увидеть все эти функции в действии, загляните в листинг 5.1. В нем показаны два способа вычисления среднего арифметического и стандартного отклонения для числового вектора.

Листинг 5.1. Вычисление среднего арифметического и стандартного отклонения

```
> x <- c(1,2,3,4,5,6,7,8)

> mean(x)                    (1)
[1] 4.5                      (1)
> sd(x)                      (1)
[1] 2.449490                 (1)
```

```
> n <- length(x)                    (2)
> meanx <- sum(x)/n                 (2)
> css <- sum((x - meanx)^2)         (2)
> sdx <- sqrt(css / (n-1))          (2)
> meanx                             (2)
[1] 4.5                             (2)
> sdx                               (2)
[1] 2.449490                        (2)
```

(1) **Быстрый способ**

(2) **Долгий способ**

Будет поучительно рассмотреть порядок вычисления суммы квадратов отклонений от среднего (corrected sum of squares – `css`) вторым способом.

1. `x` – это вектор `c(1,2,3,4,5,6,7,8)`, а среднее арифметическое значение равно `4.5` (`length(x)` возвращает число элементов в векторе `x`).
2. Выражение `x - meanx` вычитает `4.5` из каждого элемента вектора `x`, в результате чего получается вектор `c(-3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5)`.
3. Выражение `(x - meanx)^2` возводит в квадрат каждый элемент вектора `(x - meanx)`, в результате чего получается вектор `c(12.25, 6.25, 2.25, 0.25, 0.25, 2.25, 6.25, 12.25)`.
4. Инструкция `sum((x - meanx)^2)` вычисляет сумму всех элементов `(x - meanx)^2`, равную 42.

Создание формул в R имеет много общего с языками матричных вычислений, таких как MATLAB (более детально мы рассмотрим решение задач матричной алгебры в приложении D).

Стандартизация данных

По умолчанию функция `scale()` стандартизирует заданный столбец матрицы или таблицы данных так, чтобы его среднее арифметическое стало равно нулю, а стандартное отклонение – единице[1]:

```
newdata <- scale(mydata)
```

Для стандартизации каждого столбца так, чтобы его среднее арифметическое и стандартное отклонение приобрели заданные значения, можно использовать примерно такой программный код:

```
newdata <- scale(mydata)*SD + M
```

где `M` – желаемое значение среднего арифметического, а `SD` – стандартного отклонения.

[1] В русскоязычной литературе часто вместо термина «стандартизация» используется термин «нормализация». – *Прим. перев.*

Применение функции `scale()` к столбцам с нечисловыми данными вызывает сообщение об ошибке. Чтобы стандартизировать определенный столбец, а не всю матрицу или таблицу данных целиком, можно использовать такой программный код:

```
newdata <- transform(mydata, myvar = scale(myvar)*10+50)
```

Этот код преобразует вектор `myvar` так, что его среднее арифметическое становится равным `50`, а стандартное отклонение — `10`. Мы будем использовать функцию `scale()` для решения описанной выше задачи в разделе 5.3.

5.2.3. Функции распределения вероятности

Вы можете задаться вопросом: почему функции распределения вероятности не были рассмотрены вместе со статистическими (это действительно странно, не правда ли?). Хотя функции распределения вероятности являются статистическими, они настолько своеобразны, что заслуживают отдельного обсуждения. Эти функции обычно используются для создания искусственных данных с известными параметрами и для вычисления вероятностей в пользовательских статистических функциях.

В программе R функции распределения вероятности имеют вид:

```
[dpqr]distribution_abbreviation()
```

где первая буква в имени обозначает параметр распределения данных:

d = плотность;
p = функция распределения;
q = функция, определяющая квантили;
r = генератор случайных отклонений.

Наиболее распространенные функции распределения вероятности перечислены в табл. 5.4.

Таблица 5.4. Типы распределений вероятностей

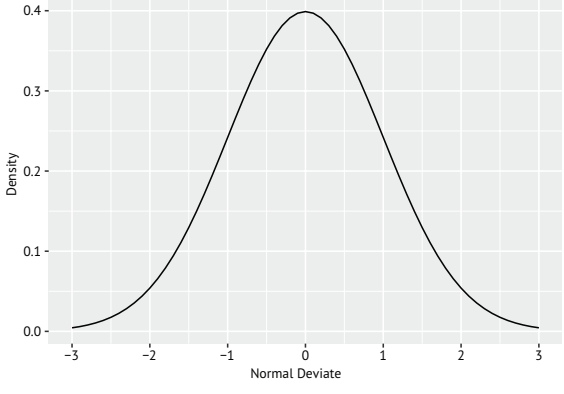
Распределение	Краткое название	Распределение	Краткое название
Бета	`beta`	Логистическое	`logis`
Биномиальное	`binom`	Мультиномиальное	`multinom`
Коши	`cauchy`	Отрицательное биномиальное	`nbinom`
Хи-квадрат (асимметричное)	`chisq`	Нормальное	`norm`
Экспоненциальное	`exp`	Пуассоновское	`pois`
F	`f`	Знаковых рангов Вилкоксона	`signrank`

Окончание табл. 5.4

Распределение	Краткое название	Распределение	Краткое название
Гамма	gamma	T	t
Геометрическое	geom	Равномерное	unif
Гипергеометрическое	hyper	Вейбулла	weibull
Логнормальное	lnorm	Суммы рангов Вилкоксона	wilcox

Чтобы понять, как они работают, рассмотрим функцию нормального распределения. Если значения среднего арифметического и стандартного отклонения не указаны явно, то по умолчанию будет воспроизводиться стандартное нормальное распределение (среднее арифметическое равно `0`, стандартное отклонение равно `1`). Примеры функций плотности (`dnorm`), распределения (`pnorm`), квантилей (`qnorm`) и генератора случайных отклонений (`rnorm`) приведены в табл. 5.5.

Таблица 5.5. Функции нормального распределения

Задача	Решение
Как нарисовать кривую стандартного нормального распределения в диапазоне значений [–3, 3] (см. рис. ниже)? 	`library(ggplot2)` `x <- seq(from = -3, to = 3, by = 0.1)` `y = dnorm(x)` `data <- data.frame(x = x, y=y)` `ggplot(data, aes(x, y)) +` `geom_line() +` `labs(x = "Normal Deviate",` `y = "Density") +` `scale_x_continuous(` `breaks = seq(-3, 3, 1))`
Как определить площадь под кривой стандартного нормального распределения слева от `z=1.96`?	`pnorm(1.96)` вернет `0.975`
Как определить значение 90-го процентиля нормального распределения со средним значением 500 и стандартным отклонением 100?	`qnorm(.9, mean=500, sd=100)` вернет `628.16`
Как создать 50 случайных чисел, принадлежащих нормальному распределению со средним значением 50 и стандартным отклонением 10?	`rnorm(50, mean=50, sd=10)`

Выбор начального значения для генератора случайных чисел

Каждый раз, когда генерируется новая последовательность псевдослучайных чисел, используется новое начальное число, чтобы на выходе получались разные результаты. Однако, чтобы сделать результаты воспроизводимыми, можно это начальное число задать явно при помощи функции `set.seed()`, как показано в листинге 5.2. Здесь используется функция `runif()`, генерирующая псевдослучайные числа, принадлежащие однородному распределению в интервале от 0 до 1.

Листинг 5.2. Генерирование псевдослучайных чисел из однородного распределения

```
> runif(5)
[1] 0.8725344 0.3962501 0.6826534 0.3667821 0.9255909
> runif(5)
[1] 0.4273903 0.2641101 0.3550058 0.3233044 0.6584988
> set.seed(1234)
> runif(5)
[1] 0.1137034 0.6222994 0.6092747 0.6233794 0.8609154
> set.seed(1234)
> runif(5)
[1] 0.1137034 0.6222994 0.6092747 0.6233794 0.8609154
```

Задавая начальное число вручную, можно получать воспроизводимые результаты. Это может пригодиться при создании примеров, к которым вы и другие люди будете возвращаться впоследствии.

Генерирование многомерных данных с нормальным распределением

В исследованиях с использованием искусственных данных и методов Монте-Карло часто бывает необходимо генерировать многомерные данные с нормальным распределением, соответствующие заданным вектору средних значений и ковариационной матрице.

Функция `draw.d.variate.normal()` из пакета `MultiRNG` позволяет легко справиться с этой задачей. Эта функция имеет следующий синтаксис:

```
draw.d.variate.normal(n, nvar, mean, sigma)
```

где *n* – требуемый объем выборки, *nvar* – количество переменных, *mean* – вектор средних значений, а *sigma* – ковариационная (или корреляционная) матрица. Код в листинге 5.3 создаст выборку из 500 наблюдений с многомерным нормальным распределением трех переменных со следующими параметрами:

Вектор средних значений	230,7	146,7	3,6
Ковариационная матрица	15 360,8	6721,2	–47,1
	6721,2	4700,9	–16,5
	–47,1	–16,5	0,3

Листинг 5.3. Генерирование данных с многомерным нормальным распределением

```
> install.packages("MultiRNG")
> library(MultiRNG)
> options(digits=3)
> set.seed(1234)                                                    (1)

> mean <- c(230.7, 146.7, 3.6)
> sigma <- matrix(c(15360.8, 6721.2, -47.1,
                    6721.2, 4700.9, -16.5,                          (2)
                     -47.1,  -16.5,   0.3), nrow=3, ncol=3)         (2)

> mydata <- draw.d.variate.normal(500, 3, mean, sigma)              (3)
> mydata <- as.data.frame(mydata)                                   (3)
> names(mydata) <- c("y","x1","x2")                                 (3)

> dim(mydata)                                                       (4)
[1] 500 3                                                           (4)
> head(mydata, n=10)                                                (4)
       y    x1   x2
1   81.1 122.6 3.69
2  265.1 110.4 3.49
3  365.1 235.3 2.67
4  -60.0  14.9 4.72
5  283.9 244.8 3.88
6  293.4 163.9 2.66
7  159.5  51.5 4.03
8  163.0 137.7 3.77
9  160.7 131.0 3.59
10 120.4  97.7 4.11
```

(1) **Задается случайное начальное число**

(2) **Задаются вектор средних значений и ковариационная матрица**

(3) **Генерирование данных**

(4) **Вывод результатов**

Код в листинге 5.3 задает начальное число для генератора случайных чисел, чтобы потом можно было воспроизвести полученный результат. Определяет вектор средних значений и ковариационную матрицу и генерирует 500 псевдослучайных чисел. Для удобства результаты преобразованы из матрицы в таблицу данных,

а переменным присвоены имена. В заключение код проверяет, действительно ли было сгенерировано 500 наблюдений по трем переменным, и выводит первые 10 наблюдений. Учтите, что поскольку корреляционная матрица также является ковариационной матрицей, вы можете напрямую задать структуру взаимосвязей (корреляций).

Пакет `MultiRNG` позволяет генерировать случайные данные из 10 многомерных распределений, включая многомерные версии распределений T, равномерного, Бернулли, гипергеометрического, мультиномиального, Лапласа, Уишарта.

С помощью функций распределения вероятности можно генерировать синтетические (искусственные) данные, подчиняющиеся распределениям с известными параметрами. Число статистических методов, которые используют искусственные данные, в настоящее время лавинообразно растет, и вы увидите несколько примеров их применения в следующих главах.

5.2.4. Текстовые функции

В то время как математические и статистические функции оперируют числовыми данными, текстовые функции извлекают информацию из текстовых данных или изменяют формат текстовых данных для вывода на экран и составления отчетов. Например, вам может понадобиться объединить имя и фамилию человека в одной ячейке таблицы и гарантировать, что они начинаются с прописных букв. Некоторые из наиболее часто используемых текстовых функций перечислены в табл. 5.6.

Таблица 5.6. Текстовые функции

Функция	Описание
`nchar(`*`x`*`)`	Подсчитывает число элементов в *x*. x <- c("ab", "cde", "fghij") `length(x)` вернет 3 (табл. 5.7). `nchar(x[3])` вернет 5
`substr(`*`x, start, stop`*`)`	Извлекает или замещает часть текстового вектора. x <- "abcdef" `substr(x, 2, 4)` вернет `"bcd"`. substr(x, 2, 4) <- "22222" (x теперь содержит текст `"a222ef"`)
`grep(`*`pattern, x,`* `ignore.case=FALSE, fixed=FALSE)`	Ищет совпадения с шаблоном *`pattern`* в *x*. Если fixed=FALSE, то *`pattern`* интерпретируется как регулярное выражение. Если fixed=TRUE, то *`pattern`* интерпретируется как простая текстовая строка. Возвращает индексы найденных совпадений. `grep("A", c("b","A","c"), fixed=TRUE)` вернет 2

Окончание табл. 5.6

Функция	Описание
`sub(pattern, replacement, x, ignore.case=FALSE, fixed=FALSE)`	Ищет совпадения с шаблоном *pattern* в *x* и замещает найденное совпадение текстом *replacement*. Если fixed=FALSE, то *pattern* интерпретируется как регулярное выражение. Если fixed=TRUE, то *pattern* интерпретируется как простая текстовая строка. `sub("\\s",".","Hello There")` вернет `Hello.There`. Обратите внимание, что `"\s"` – это регулярное выражение, совпадающее с пробелами; здесь используется `"\\s"`, потому что обратный слеш `"\"` в R – это экранирующий символ (раздел 1.3.4)
`strsplit(x, split, fixed=FALSE)`	Разбивает текстовый вектор *x* на элементы по значению *split*. Если fixed=FALSE, то *split* интерпретируется как регулярное выражение. Если fixed=TRUE, то *split* интерпретируется как простая текстовая строка. `y <- strsplit("abc", "")` вернет один компонент – список с тремя элементами: `"a" "b" "c"`. `unlist(y)[2]` и `sapply(y, "[", 2)` вернут `"b"`
`paste(..., sep="")`	Объединяет строки, разделяя их заданной строкой sep. `paste("x", 1:3, sep="")` вернет `c("x1", "x2", "x3")`. `paste("x",1:3,sep="M")` вернет `c("xM1","xM2" "xM3")`. `paste("Today is", date())` вернет `Today is Thu Jul 22 10:36:14 2021`
`toupper(x)`	Преобразует строку *x* в верхний регистр. `toupper("abc")` вернет `"ABC"`
`tolower(x)`	Преобразует строку *x* в нижний регистр. `tolower("ABC")` вернет `"abc"`

Обратите внимание, что функции `grep()`, `sub()` и `strsplit()` способны осуществлять поиск текстовых строк (fixed=TRUE) или регулярных выражений (fixed=FALSE, по умолчанию). Регулярные выражения имеют простой и последовательный синтаксис описания искомой комбинации символов. Например, регулярное выражение

```
^[hc]?at
```

соответствует всем строкам, которые начинаются с 0 или с одной буквы `h` или `c`, за которой следует `at`. Таким образом, это выражение позволит отыскать такие слова, как `hat`, `cat` и `at`, но не `bat`. Чтобы узнать больше, поищите в Википедии статьи по теме «регулярные выражения». Также можно порекомендовать статью «Regular Expression Tutorial» (https://ryanstutorials.net/regular-expressions-tutorial/) и интерактивное руководство «RegexOne» (https://regexone.com/).

5.2.5. *Другие полезные функции*

Функции, перечисленные в табл. 5.7, тоже могут пригодиться для управления данными и их преобразования, однако их нельзя уверенно отнести ни к одной из названных выше категорий.

Таблица 5.7. Другие полезные функции

Функция	Описание
`length(x)`	Число элементов в объекте *x*. x <- c(2, 5, 6, 9) `length(x)` вернет 4
`seq(from, to, by)`	Создает последовательность элементов. indices <- seq(1,10,2) в `indices` будет помещен вектор `c(1, 3, 5, 7, 9)`
`rep(x, n)`	Повторяет *x n* раз. y <- rep(1:3, 2) в y будет помещен вектор `c(1, 2, 3, 1, 2, 3)`
`cut(x, n)`	Преобразует непрерывную переменную *x* в фактор с *n* уровнями. Для создания упорядоченного фактора можно добавить параметр `ordered_result = TRUE`
`cat(... , file = "myfile", append = FALSE)`	Объединяет и выводит объекты в `...` на экран или в файл (если указано его имя). name <- c("Jane") cat("Hello" , name, "\n")

Последний пример в табл. 5.7 показывает, как используется символ экранирования при выводе результатов на экран: `\n` добавляет перевод строки, `\t` – символ табуляции, `\b` – забой (возврат к предыдущему символу) и т. д. (введите `?Quotes` для получения дополнительной информации). Например, следующий код:

```
name <- "Bob"
cat( "Hello", name, "\b.\n", "Isn\'t R", "\t", "GREAT?\n")
```

выведет:

```
Hello Bob.
 Isn't R        GREAT?
```

Обратите внимание, что вторая строка смещена на один символ вправо. Когда функция `cat()` объединяет объекты для вывода, она разделяет их пробелами. Вот почему был добавлен символ забоя (`\b`) перед точкой. Иначе получилось бы `"Hello Bob"`.

Способы применения изученных на данный момент функций к числам, строкам и векторам интуитивно понятны и просты, но как применять их к матрицам и таблицам данных? Это и станет темой следующего раздела.

5.2.6. Применение функций к матрицам и таблицам данных

Одно из интересных свойств функций R – возможность применять их к объектам разных типов (скалярам, векторам, матрицам, массивам и таблицам данных). Код в листинге 5.4 послужит хорошей иллюстрацией.

Листинг 5.4. Применение функций к объектам

```
> a <- 5
> sqrt(a)
[1] 2.236068
> b <- c(1.243, 5.654, 2.99)
> round(b)
[1] 1 6 3
> c <- matrix(runif(12), nrow=3)
> c
       [,1]  [,2]  [,3]  [,4]
[1,] 0.4205 0.355 0.699 0.323
[2,] 0.0270 0.601 0.181 0.926
[3,] 0.6682 0.319 0.599 0.215
> log(c)
       [,1]   [,2]   [,3]   [,4]
[1,] -0.866 -1.036 -0.358 -1.130
[2,] -3.614 -0.508 -1.711 -0.077
[3,] -0.403 -1.144 -0.513 -1.538
> mean(c)
[1] 0.444
```

Обратите внимание, что в листинге 5.4 среднее значение для матрицы `c` равно скаляру (`0.444`). Функция `mean()` вычисляет среднее арифметическое для всех 12 элементов матрицы. А что, если понадобится вычислить средние значения для каждой из трех строк или каждого из четырех столбцов?

В R есть функция `apply()`, которая позволяет применить любую функцию к любой части матрицы, массива или таблицы данных. Она имеет следующий синтаксис:

```
apply(x, MARGIN, FUN, ...)
```

где `x` – это объект, `MARGIN` – индекс измерения, к которому применяется функция (столбцы или строки), `FUN` – функция и `...` – другие параметры. Для матрицы или таблицы данных `MARGIN=1` обозначает строки, а `MARGIN=2` – столбцы. Взгляните на примеры в листинге 5.5.

Листинг 5.5. Применение функции к строкам (столбцам) матрицы

```
> mydata <- matrix(rnorm(30), nrow=6)                    (1)
> mydata
         [,1]   [,2]    [,3]   [,4]   [,5]
[1,]  0.71298  1.368 -0.8320 -1.234 -0.790
```

```
[2,] -0.15096 -1.149 -1.0001 -0.725  0.506
[3,] -1.77770  0.519 -0.6675  0.721 -1.350
[4,] -0.00132 -0.308  0.9117 -1.391  1.558
[5,] -0.00543  0.378 -0.0906 -1.485 -0.350
[6,] -0.52178 -0.539 -1.7347  2.050  1.569
> apply(mydata, 1, mean)                          (2)
[1] -0.155 -0.504 -0.511  0.154 -0.310  0.165
> apply(mydata, 2, mean)                          (3)
[1] -0.2907  0.0449 -0.5688 -0.3442  0.1906
> apply(mydata, 2, mean, trim=0.2)                (4)
[1] -0.1699  0.0127 -0.6475 -0.6575  0.2312
```

(1) Генерирует данные

(2) Вычисляются средние значения по строкам

(3) Вычисляются средние значения по столбцам

(4) Вычисляются усеченные средние значения по столбцам

Здесь сначала генерируется матрица 6×5, содержащая случайные числа, подчиняющиеся нормальному распределению (1). Затем вычисляются средние значения для каждой из шести строк (2) и каждого из пяти столбцов (3). Наконец, вычисляются усеченные средние значения для каждого столбца (в данном случае усреднены «центральные» 60 % данных, без учета 20 % наибольших и 20 % наименьших значений) (4).

Поскольку в параметре `FUN` можно передать любую функцию, в том числе и написанную вами (раздел 5.4), функция `apply()` действительно оказывается мощным средством. Функция `apply()` применяется к строкам или столбцам массива данных, а ее аналоги, `lapply()` и `sapply()`, применяют заданную функцию к целому списку. Вы увидите пример применения функции `sapply()` (более удобной версии функции `lapply()`) в следующем разделе.

Теперь у вас есть все необходимые инструменты для управления данными, представленными в разделе 5.1, так что давайте попробуем сделать это.

5.2.7. Решение задачи по управлению данными

В задаче из раздела 5.1 требовалось объединить результаты экзаменов по каждому предмету в единый балл успеваемости для каждого студента, каждому поставить оценку от `A` до `F` в зависимости от позиции в общем рейтинге (верхние 20 %, следующие 20 % и т. д.), отсортировать строки в списке по фамилии студентов, а затем и по имени. Решение этой задачи представлено в листинге 5.6.

Листинг 5.6. Решение учебной задачи

```
> options(digits=2)                                                    (1)

> Student <- c("John Davis", "Angela Williams", "Bullwinkle Moose",
               "David Jones", "Janice Markhammer", "Cheryl Cushing",
               "Reuven Ytzrhak", "Greg Knox", "Joel England",
               "Mary Rayburn")
> Math <- c(502, 600, 412, 358, 495, 512, 410, 625, 573, 522)
> Science <- c(95, 99, 80, 82, 75, 85, 80, 95, 89, 86)
> English <- c(25, 22, 18, 15, 20, 28, 15, 30, 27, 18)
> roster <- data.frame(Student, Math, Science, English,
                       stringsAsFactors=FALSE)
> z <- scale(roster[,2:4])                                             (2) (4)
> score <- apply(z, 1, mean)                                           (3) (4)
> roster <- cbind(roster, score)                                       (3) (4)

> y <- quantile(score, c(.8,.6,.4,.2))                                 (5) (7)
> roster$grade <- NA                                                   (6) (7)
> roster$grade[score >= y[1]] <- "A"                                   (6) (7)
> roster$grade[score < y[1] & score >= y[2]] <- "B"                    (6) (7)
> roster$grade[score < y[2] & score >= y[3]] <- "C"                    (6) (7)
> roster$grade[score < y[3] & score >= y[4]] <- "D"                    (6) (7)
> roster$grade[score < y[4]] <- "F"                                    (6) (7)

> name <- strsplit((roster$Student), " ")                              (8) (10)
> Lastname <- sapply(name, "[", 2)                                     (9) (10)
> Firstname <- sapply(name, "[", 1)                                    (9) (10)
> roster <- cbind(Firstname,Lastname, roster[,-1])                     (9) (10)

> roster <- roster[order(Lastname,Firstname),]                         (11) (12)

> roster
    Firstname   Lastname Math Science English score grade
6      Cheryl    Cushing  512      85      28  0.35     C
1        John      Davis  502      95      25  0.56     B
9        Joel    England  573      89      27  0.70     B
4       David      Jones  358      82      15 -1.16     F
8        Greg       Knox  625      95      30  1.34     A
5      Janice Markhammer  495      75      20 -0.63     D
3   Bullwinkle      Moose  412      80      18 -0.86     D
10       Mary    Rayburn  522      86      18 -0.18     C
2      Angela   Williams  600      99      22  0.92     A
7      Reuven    Ytzrhak  410      80      15 -1.05     F
```

(1) **Шаг 1**

(2) **Шаг 2**

(3) **Шаг 3**

(4) **Вычисление объединенного показателя успеваемости**

(5) **Шаг 4**

6 **Шаг 5**
7 **Оценивание студентов**
8 **Шаг 6**
9 **Шаг 7**
10 **Извлечение фамилий и имен**
11 **Шаг 8**
12 **Сортировка по фамилиям и именам**

Это очень насыщенный пример, поэтому рассмотрим его подробнее, шаг за шагом.

1 Создается исходный набор данных. Инструкция `options(digits=2)` сокращает число знаков после запятой до двух у всех выводимых на экран чисел, это упрощает их восприятие:

```
> options(digits=2)
> roster
              Student Math Science English
1         John Davis  502      95      25
2    Angela Williams  600      99      22
3   Bullwinkle Moose  412      80      18
4        David Jones  358      82      15
5  Janice Markhammer  495      75      20
6     Cheryl Cushing  512      85      28
7     Reuven Ytzrhak  410      80      15
8          Greg Knox  625      95      30
9        Joel England  573      89      27
10      Mary Rayburn  522      86      18
```

2 Поскольку оценки по математике, естественным наукам и английскому языку выставлены по разным шкалам (их средние и стандартные отклонения заметно различаются), их необходимо сделать сопоставимыми перед объединением. Один из подходов – стандартизировать переменные так, чтобы каждая оценка выражалась в стандартных отклонениях, а не в абсолютных баллах. Это можно сделать при помощи функции `scale()`:

```
> z <- scale(roster[,2:4])
> z
        Math  Science  English
 [1,]  0.013   1.078    0.587
 [2,]  1.143   1.591    0.037
 [3,] -1.026  -0.847   -0.697
 [4,] -1.649  -0.590   -1.247
 [5,] -0.068  -1.489   -0.330
 [6,]  0.128  -0.205    1.137
 [7,] -1.049  -0.847   -1.247
 [8,]  1.432   1.078    1.504
 [9,]  0.832   0.308    0.954
[10,]  0.243  -0.077   -0.697
```

3 Рассчитать показатель успеваемости студентов можно, усреднив значения стандартизированных баллов для каждой строки с помощью функции `mean()` и добавив их в список при помощи функции `cbind()`:

```
> score <- apply(z, 1, mean)
> roster <- cbind(roster, score)
> roster
              Student Math Science English score
1         John Davis  502      95      25  0.56
2    Angela Williams  600      99      22  0.92
3   Bullwinkle Moose  412      80      18 -0.86
4        David Jones  358      82      15 -1.16
5  Janice Markhammer  495      75      20 -0.63
6     Cheryl Cushing  512      85      28  0.35
7     Reuven Ytzrhak  410      80      15 -1.05
8          Greg Knox  625      95      30  1.34
9       Joel England  573      89      27  0.70
10      Mary Rayburn  522      86      18 -0.18
```

4 Функция `quantile()` позволяет вычислить границы 20%-ных интервалов (процентилей) показателя успеваемости. Как видите, нижняя граница для оценки `A` равна `0.74`, для оценки `B` – `0.44` и т. д.:

```
> y <- quantile(roster$score, c(.8,.6,.4,.2))
> y
  80%   60%   40%   20%
 0.74  0.44 -0.36 -0.89
```

5 Используя логические операторы, можно перекодировать значения показателя успеваемости в новую категориальную переменную итоговой оценки. Для этого в таблице данных `roster` создается переменная `grade`:

```
> roster$grade <- NA
> roster$grade[score >= y[1]] <- "A"
> roster$grade[score < y[1] & score >= y[2]] <- "B"
> roster$grade[score < y[2] & score >= y[3]] <- "C"
> roster$grade[score < y[3] & score >= y[4]] <- "D"
> roster$grade[score < y[4]] <- "F"
> roster
              Student Math Science English score grade
1         John Davis  502      95      25  0.56     B
2    Angela Williams  600      99      22  0.92     A
3   Bullwinkle Moose  412      80      18 -0.86     D
4        David Jones  358      82      15 -1.16     F
5  Janice Markhammer  495      75      20 -0.63     D
6     Cheryl Cushing  512      85      28  0.35     C
7     Reuven Ytzrhak  410      80      15 -1.05     F
8          Greg Knox  625      95      30  1.34     A
9       Joel England  573      89      27  0.70     B
10      Mary Rayburn  522      86      18 -0.18     C
```

6 Отделить друг от друга имена и фамилии студентов, разделенные пробелом, можно с помощью функции `strsplit()`. Применение этой функции к вектору из строк приводит к созданию списка:

```
> name <- strsplit((roster$Student), " ")
> name

[[1]]
[1] "John"  "Davis"

[[2]]
[1] "Angela"   "Williams"

[[3]]
[1] "Bullwinkle" "Moose"

[[4]]
[1] "David" "Jones"

[[5]]
[1] "Janice"     "Markhammer"

[[6]]
[1] "Cheryl"  "Cushing"

[[7]]
[1] "Reuven"  "Ytzrhak"

[[8]]
[1] "Greg" "Knox"

[[9]]
[1] "Joel"    "England"

[[10]]
[1] "Mary"    "Rayburn"
```

7 Применение функции `sapply()`, чтобы поместить первый элемент каждого компонента в вектор с именами `Firstname`, а второй – в вектор с фамилиями `Lastname`. "[" – это функция, извлекающая часть объекта – в данном случае первый или второй элемент списка `name`. Добавить эти элементы в набор данных `roster` можно с помощью функции `cbind()`. Поскольку переменная `student` с именем и фамилией каждого студента больше не нужна, от нее можно избавиться (указав индекс `-1` в обращении к таблице данных `roster`):

```
> Firstname <- sapply(name, "[", 1)
> Lastname <- sapply(name, "[", 2)
> roster <- cbind(Firstname, Lastname, roster[,-1])
```

```
> roster
    Firstname   Lastname Math Science English score grade
1        John      Davis  502      95      25  0.56     B
2      Angela   Williams  600      99      22  0.92     A
3  Bullwinkle      Moose  412      80      18 -0.86     D
4       David      Jones  358      82      15 -1.16     F
5      Janice Markhammer  495      75      20 -0.63     D
6      Cheryl    Cushing  512      85      28  0.35     C
7      Reuven    Ytzrhak  410      80      15 -1.05     F
8        Greg       Knox  625      95      30  1.34     A
9        Joel    England  573      89      27  0.70     B
10       Mary    Rayburn  522      86      18 -0.18     C
```

8 Наконец, можно отсортировать список по именам и фамилиям студентов при помощи функции `order()`:

```
> roster[order(Lastname,Firstname),]
    Firstname   Lastname Math Science English score grade
6      Cheryl    Cushing  512      85      28  0.35     C
1        John      Davis  502      95      25  0.56     B
9        Joel    England  573      89      27  0.70     B
4       David      Jones  358      82      15 -1.16     F
8        Greg       Knox  625      95      30  1.34     A
5      Janice Markhammer  495      75      20 -0.63     D
3  Bullwinkle      Moose  412      80      18 -0.86     D
10       Mary    Rayburn  522      86      18 -0.18     C
2      Angela   Williams  600      99      22  0.92     A
7      Reuven    Ytzrhak  410      80      15 -1.05     F
```

Вот и все! Пара пустяков!

Существует много других способов решить эту же задачу, но приведенный пример позволяет понять идею применения этих функций. Теперь настала пора обратиться к условным операторам и функциям, которые пишутся самими пользователями.

5.3. Управление потоком выполнения

Обычно инструкции в R выполняются последовательно, от начала программы до конца. Однако иногда некоторые инструкции нужно выполнить несколько раз, а другие – только при определенных условиях. В таких ситуациях следует использовать специальные конструкции управления потоком выполнения.

В R реализованы многие стандартные конструкции такого типа, которые можно встретить в любом другом современном языке программирования. Сначала мы рассмотрим условные конструкции, а затем – конструкции, используемые для циклического выполнения.

Во всех примерах этого раздела будут использоваться следующие сокращения:

- `statement` – простая или составная инструкция (группа инструкций, заключенных в фигурные скобки {} и разделенных точкой с запятой);
- `cond` – условное выражение, которое может возвращать `TRUE` или `FALSE`;
- `expr` – выражение, возвращающее число или текстовую строку;
- `seq` – последовательность чисел или текстовых строк.

После обсуждения конструкций управления потоком выполнения мы перейдем к созданию своих собственных функций.

5.3.1. Циклы

Конструкции циклов многократно выполняют одну и ту же инструкцию или их набор, пока не выполнится заданное условие. К таким конструкциям относятся циклы `for` и `while`.

for

Конструкция цикла `for` выполняет инструкцию `statement` для каждого значения в последовательности `seq`. Она имеет следующий синтаксис:

```
for (var in seq) statement
```

Следующий код:

```
for (i in 1:10)  print("Hello")
```

выведет слово `Hello` 10 раз.

while

Конструкция цикла `while` повторно выполняет инструкцию, пока заданное условие остается истинным. Она имеет следующий синтаксис:

```
while (cond) statement
```

Следующий код:

```
i <- 10
while (i > 0) {print("Hello"); i <- i - 1}
```

тоже выведет слово `Hello` 10 раз. Убедитесь, что инструкции внутри фигурных скобок изменяют переменную, проверяемую в условном выражении `cond` так, что рано или поздно оно перестанет быть истинным, иначе цикл никогда не завершится! В предыдущем примере инструкция

```
i <- i - 1
```

вычитает 1 из объекта `i` в каждом цикле, поэтому после десятого цикла значение `i` не будет больше 0. Если после каждого цикла, на-

оборот, прибавлять по единице, то программа никогда не перестала бы выводить Hello. Вот почему while может быть опаснее других циклических конструкций.

Циклы в R могут быть неэффективными и занимать много времени, особенно при обработке больших объемов данных. При любой возможности лучше вместо циклов использовать встроенные числовые и текстовые функции в сочетании с функциями из семейства apply.

5.3.2. Выполнение по условию

Выполнение по условию означает, что инструкции выполняются, только если соблюдается определенное условие. К таким конструкциям относятся if-else, ifelse и switch.

if-else

Условный оператор if-else выполняет инструкцию, если заданное условие верно. Также есть возможность выполнить другую инструкцию, если условие не верно. Он имеет следующий синтаксис:

```
if (cond) statement
if (cond) statement1 else statement2
```

Например:

```
if (is.character(grade)) grade <- as.factor(grade)
if (!is.factor(grade)) grade <- as.factor(grade) else print("Grade already
    is a factor")
```

В первом случае, если переменная grade – текстовый вектор, она преобразуется в фактор. Во втором случае выполняется одна из двух инструкций. Если переменная grade – не фактор (обратите внимание на символ !), то она преобразуется в него. Если эта переменная – уже фактор, то выводится сообщение об этом.

ifelse

Условный оператор ifelse – компактная и векторизованная версия оператора if-else. Он имеет следующий синтаксис:

```
ifelse(cond, statement1, statement2)
```

Инструкция statement1 выполняется, если условие cond истинно. Если условие ложно – выполняется вторая инструкция statement2. Например:

```
ifelse(score > 0.5, print("Passed"), print("Failed"))
outcome <- ifelse (score > 0.5, "Passed", "Failed")
```

Эту конструкцию можно использовать для выполнения одного из двух действий или для ввода/вывода векторов.

switch

Оператор выбора `switch` выбирает инструкцию для выполнения в зависимости от значения выражения `expr`. Он имеет следующий синтаксис:

```
switch(expr, ...)
```

где многоточие (`...`) означает инструкции, соответствующие возможным значениям `expr`. Проще всего понять принцип работы оператора выбора на примере (листинг 5.7).

Листинг 5.7. Пример применения оператора выбора switch

```
> feelings <- c("sad", "afraid")
> for (i in feelings)
    print(
      switch(i,
        happy  = "I am glad you are happy",
        afraid = "There is nothing to fear",
        sad    = "Cheer up",
        angry  = "Calm down now"
      )
    )

[1] "Cheer up"
[1] "There is nothing to fear"
```

Это надуманный пример, зато он демонстрирует основные принципы применения оператора выбора. Как можно его использовать в собственных функциях, вы узнаете в следующем разделе.

5.4. Пользовательские функции

Одно из существенных преимуществ R – возможность создавать свои функции. Структура таких функций выглядит примерно так:

```
myfunction <- function(arg1, arg2, ... ){
  statements
  return(object)
}
```

Объекты, объявленные внутри функции, недоступны снаружи. Объект, который возвращается из функции, может быть любого типа – от скаляра до списка. Рассмотрим пример.

Допустим, нам понадобилась функция, вычисляющая основную тенденцию и разброс данных. Эта функция должна предоставлять выбор между параметрическими (среднее арифметическое и стандартное отклонение) и непараметрическими (медиана и ее абсолютное отклонение) статистиками. Результат должен быть представлен в виде списка с именами. Кроме того, пользователь должен иметь возможность выбора – выводить ли

результаты автоматически или нет. По умолчанию функция должна вычислять параметрические статистики и не выводить результаты на экран. Одно из возможных решений представлено в листинге 5.8.

Листинг 5.8. mystats() – пользовательская функция для вычисления обобщенных статистик

```
mystats <- function(x, parametric=TRUE, print=FALSE) {
  if (parametric) {
    center <- mean(x); spread <- sd(x)
  } else {
    center <- median(x); spread <- mad(x)
  }   if (print & parametric) {
    cat("Mean=", center, "\n", "SD=", spread, "\n")
  } else if (print & !parametric) {
    cat("Median=", center, "\n", "MAD=", spread, "\n")
  }
  result <- list(center=center, spread=spread)
  return(result)
}
```

Чтобы увидеть эту функцию в действии, создадим некоторый набор данных (случайная выборка из 500 чисел, подчиняющихся нормальному распределению):

```
set.seed(1234)
x <- rnorm(500)
```

После выполнения инструкции

```
y <- mystats(x)
```

элемент `y$center` будет содержать среднее арифметическое (`0.00184`), а элемент `y$spread` – стандартное отклонение (`1.03`). На экран ничего не выводится. Если выполнить инструкцию

```
y <- mystats(x, parametric=FALSE, print=TRUE)
```

то элемент `y$center` будет содержать медиану (`-0.0207`), а элемент `y$spread` – ее абсолютное отклонение (`1.001`). Кроме того, результат автоматически появится на экране:

```
Median= -0.0207
MAD= 1
```

В качестве следующего шага рассмотрим пользовательскую функцию, использующую оператор `switch`. Эта функция позволяет выбрать формат представления сегодняшней даты. Значения, присвоенные параметрам функции в ее объявлении, используются как значения по умолчанию. Функция `mydate()` использует значение `"long"` параметра `type` как формат даты по умолчанию, если в вызове функции не указано другое значение:

```
mydate <- function(type="long") {
  switch(type,
    long =  format(Sys.time(), "%A %B %d %Y"),
    short = format(Sys.time(), "%m-%d-%y"),
    cat(type, "is not a recognized type\n")
   )
}
```

Вот как работает эта функция:

```
> mydate("long")
[1] "Saturday July 24 2021"
> mydate("short")
[1] "07-24-21"
> mydate()
[1] "Saturday July 24 2021"
> mydate("medium")
medium is not a recognized type
```

Обратите внимание, что функция `cat()` выполняется, только если в параметре `type` передано значение, отличное от `"long"` и `"short"`. Часто бывает полезно включить в функцию выражение, «отлавливающее» ошибочные аргументы.

Существует несколько функций, помогающих отлавливать и исправлять ошибки в пользовательских функциях. Для вывода сообщений об ошибке можно использовать функцию `warning()`. Функция `message()` выводит диагностические сообщения, а функция `stop()` останавливает выполнение текущей функции и выводит сообщение об ошибке. За дополнительной информацией по этим функциям обращайтесь к встроенной справке.

Вы уже можете сделать очень многое, используя знания, полученные в этой главе. Управление потоком выполнения и другие вопросы программирования более подробно рассматриваются в главе 20. Создание пакетов обсуждается в главе 22. Если же вам понадобится изучить разработку функций во всех деталях или писать программный код на профессиональном уровне, чтобы распространять его среди других людей, то я советую прочитать две прекрасные книги, ссылки на которые вы найдете в списке литературы в конце этой книги: Venables & Ripley (2000) и Chambers (2008). Вместе они помогут вам детально разобраться в вопросе на большом количестве разнообразных примеров.

Теперь, рассмотрев создание пользовательских функций, закончим эту главу описанием способов агрегирования и реструктуризации данных.

5.5. Агрегирование и реструктуризация данных

Под *реструктуризацией* данных понимается изменение структуры (строк и столбцов), определяющей организацию данных.

В этом разделе описаны три основных метода реструктуризации: транспонирование набора данных, преобразование широкого набора данных в длинный и преобразование длинного набора данных в широкий. Все они описываются в следующих далее разделах.

5.5.1. Транспонирование

Транспонирование (переворот таблицы на 90° так, что строки и столбцы меняются местами) – самый простой, пожалуй, способ реструктуризации данных. Выполнить транспонирование матрицы или таблицы данных можно с помощью функции `t()`. В последнем случае названия строк становятся именами переменных (столбцов).

Проиллюстрируем транспонирование с использованием набора данных `mtcars`, входящего в дистрибутив R. Эти данные, взятые из журнала «Motor Trend» за 1974 год, содержат информацию о технических характеристиках (число цилиндров, объем и мощность двигателя, расход топлива и т. д.) 34 марок автомобилей. Чтобы узнать больше об этом наборе данных, наберите `help(mtcars)`.

В листинге 5.9 показан пример применения операции транспонирования. Для экономии места используется лишь ограниченное подмножество данных из этого набора.

Листинг 5.9. Транспонирование данных

```
> cars <- mtcars[1:5,1:4]
> cars
                   mpg cyl disp  hp
Mazda RX4         21.0   6  160 110
Mazda RX4 Wag     21.0   6  160 110
Datsun 710        22.8   4  108  93
Hornet 4 Drive    21.4   6  258 110
Hornet Sportabout 18.7   8  360 175
> t(cars)
     Mazda RX4 Mazda RX4 Wag Datsun 710 Hornet 4 Drive Hornet Sportabout
mpg         21            21       22.8           21.4              18.7
cyl          6             6        4.0            6.0               8.0
disp       160           160      108.0          258.0             360.0
hp         110           110       93.0          110.0             175.0
```

Функция `t()` всегда возвращает матрицу. Поскольку матрица может хранить данные только одного типа (числовые, текстовые или логические), операция транспонирования работает эффективнее, когда все переменные в исходном наборе данных являются числовыми или логическими. Если в наборе данных есть какие-либо текстовые переменные, то в процессе транспонирования *все* данные в наборе будут преобразованы в текстовые значения.

5.5.2. Преобразование широкого набора данных в длинный и обратно

Нередко набор данных имеет широкий или длинный формат. В *широком* формате каждая строка представляет уникальное наблюдение. В табл. 5.8 показан пример такого набора данных. Он содержит оценки ожидаемой продолжительности жизни для четырех стран в 1990, 2000 и 2010 годах и является частью огромного набора данных, доступного на сайте Our World in Data (https://ourworldindata.org/life-expectancy). Обратите внимание, что каждая строка представляет данные для одной страны.

Таблица 5.8. Ожидаемая продолжительность жизни по годам и странам

ID	Country	LExp1990	LExp2000	LExp2010
AU	Australia	76,9	79,6	82,0
CN	China	69,3	72,0	75,2
PRK	North Korea	69,9	65,3	69,6

В *длинном* формате каждая строка представляет уникальное измерение. В табл. 5.9 показан пример с теми же данными в длинном формате.

Таблица 5.9. Ожидаемая продолжительность жизни по годам и странам – длинный формат

ID	Country	Variable	LifeExp
AU	Australia	LExp1990	76,9
CN	China	LExp1990	69,3
PRK	North Korea	LExp1990	69,9
AU	Australia	LExp2000	79,6
CN	China	LExp2000	72,0
PRK	North Korea	LExp2000	65,3
AU	Australia	LExp2010	82,0
CN	China	LExp2010	75,2
PRK	North Korea	LExp2010	69,6

Для разных видов анализа могут потребоваться данные в разных форматах. Например, чтобы определить страны, в которых наблюдаются схожие тенденции изменения ожидаемой продолжительности жизни с течением времени, можно использовать кластерный анализ (глава 16). Для кластерного анализа желательно,

чтобы данные были представлены в широком формате. С другой стороны, для предсказания ожидаемой продолжительности жизни по стране и году можно использовать множественную регрессию (глава 8). В этом случае желательно, чтобы данные были представлены в длинном формате.

Большинство функций в R принимают таблицы данных в широком формате, но некоторые требуют, чтобы данные находились в длинном формате. К счастью, пакет `tidyr` предоставляет функции для преобразования таблиц данных из одного формата в другой. Прежде чем продолжить, не забудьте установить пакет инструкцией `install.packages("tidyr")`.

Функция `gather()` из пакета `tidyr` преобразует таблицу данных с широким форматом в таблицу данных с длинным форматом. Она имеет следующий синтаксис:

```
longdata <- gather(widedata, key, value, variable_list)
```

где:

- `widedata` – исходная таблица данных, подлежащая преобразованию;
- `key` – имя для столбца с переменной (в данном примере `Variable`);
- `value` – имя для столбца со значением (в данном примере `LifeExp`);
- `variable_list` – список переменных для реорганизации в один столбец (в данном примере `LExp1990`, `LExp2000`, `LExp2010`).

Пример такого преобразования показан в листинге 5.10.

Листинг 5.10. Преобразование таблицы данных из широкого формата в длинный

```
> library(tidyr)

> data_wide <- data.frame(ID = c("AU", "CN", "PRK"),
                          Country = c("Australia", "China", "North Korea"),
                            LExp1990 = c(76.9, 69.3, 69.9),
                            LExp2000 = c(79.6, 72.0, 65.3),
                            LExp2010 = c(82.0, 75.2, 69.6))
> data_wide
   ID     Country LExp1990 LExp2000 LExp2010
1  AU   Australia     76.9     79.6     82.0
2  CN       China     69.3     72.0     75.2
3 PRK North Korea     69.9     65.3     69.6

> data_long <- gather(data_wide, key="Variable", value="Life_Exp",
                      c(LExp1990, LExp2000, LExp2010))
> data_long
   ID     Country Variable Life_Exp
```

```
1  AU   Australia LExp1990     76.9
2  CN       China LExp1990     69.3
3 PRK North Korea LExp1990     69.9
4  AU   Australia LExp2000     79.6
5  CN       China LExp2000     72.0
6 PRK North Korea LExp2000     65.3
7  AU   Australia LExp2010     82.0
8  CN       China LExp2010     75.2
9 PRK North Korea LExp2010     69.6
```

Функция `spread()` из пакета `tidyr` преобразует таблицу данных из длинного формата в широкий. Она имеет следующий синтаксис:

```
widedata <- spread(longdata, key, value)
```

где:

- `longdata` – исходная таблица данных, подлежащая преобразованию;
- `key` – имя для столбца с именами переменных;
- `value` – имя для столбца со значениями переменных.

В продолжение примера код в листинге 5.11 преобразует таблицу данных в длинном формате в таблицу данных в широком формате.

Листинг 5.11. Преобразование таблицы данных из длинного формата в широкий

```
> data_wide <- spread(data_long, key=Variable, value=Life_Exp)
> data_wide
   ID     Country LExp1990 LExp2000 LExp2010
1  AU   Australia     76.9     79.6     82.0
2  CN       China     69.3     72.0     75.2
3 PRK North Korea     69.9     65.3     69.6
```

Узнать больше о длинных и широких форматах данных можно в замечательном руководстве Симона Эджемира (Simon Ejdemyr), доступном по адресу: https://sejdemyr.github.io/r-tutorials/basics/wide-and-long/.

5.6. Агрегирование данных

Агрегирование данных заключается в замене групп наблюдений сводной статистикой, основанной на этих наблюдениях. Агрегирование данных может предшествовать статистическому анализу или обобщению данных для представления в виде таблиц или диаграмм.

В R данные агрегируются с помощью функции `aggregate()`, принимающей список переменных для агрегирования и функцию, собственно выполняющую агрегирование. Она имеет следующий синтаксис:

```
aggregate(x, by, FUN)
```

где x – объект с исходными данными, by – список переменных, на основе которых будут формироваться новые наблюдения, а FUN – функция для вычисления обобщенных статистик, которые послужат значениями в новых наблюдениях.

Для примера давайте агрегируем данные mtcars по числу цилиндров и передач, вычислив средние арифметические всех числовых переменных (листинг 5.12).

Листинг 5.12. Агрегирование данных с помощью функции aggregate()

```
> options(digits=3)
> aggdata <-aggregate(mtcars,
                      by=list(mtcars$cyl,mtcars$gear),
                      FUN=mean, na.rm=TRUE)
> aggdata
  Group.1 Group.2  mpg cyl disp  hp drat   wt qsec  vs   am gear carb
1       4       3 21.5   4  120  97 3.70 2.46 20.0 1.0 0.00    3 1.00
2       6       3 19.8   6  242 108 2.92 3.34 19.8 1.0 0.00    3 1.00
3       8       3 15.1   8  358 194 3.12 4.10 17.1 0.0 0.00    3 3.08
4       4       4 26.9   4  103  76 4.11 2.38 19.6 1.0 0.75    4 1.50
5       6       4 19.8   6  164 116 3.91 3.09 17.7 0.5 0.50    4 4.00
6       4       5 28.2   4  108 102 4.10 1.83 16.8 0.5 1.00    5 2.00
7       6       5 19.7   6  145 175 3.62 2.77 15.5 0.0 1.00    5 6.00
8       8       5 15.4   8  326 300 3.88 3.37 14.6 0.0 1.00    5 6.00
```

В представленных результатах Group.1 – это число цилиндров (4, 6 или 8), а Group.2 – это число передач (3, 4 или 5). Например, машинам с четырьмя цилиндрами и тремя передачами хватает одного галлона топлива, чтобы проехать в среднем 21.5 мили. Здесь использована функция mean(), но вообще можно использовать любую функцию, стандартную или написанную пользователем, вычисляющую сводную статистику.

В этом примере есть два недостатка. Во-первых, имена переменных Group.1 и Group.2 ужасно неинформативны.

Во-вторых, исходные переменные cyl и gear включаются в агрегированную таблицу данных, в чем нет необходимости.

R предлагает возможность преодолеть эти недостатки. Во-первых, определяя список переменных для агрегирования, им можно присвоить свои имена. Например, by=list(Cylinders=cyl, Gears=gear) заменит Group.1 и Group.2 на Cylinders и Gears. Во-вторых, лишние столбцы можно удалить, используя скобочную нотацию (mtcars[-c(2, 10]). В листинге 5.13 показана улучшенная версия.

Листинг 5.13. Агрегирование данных с помощью функции aggregate() (улучшенная версия)

```
> aggdata <-aggregate(mtcars[-c(2, 10)],
            by=list(Cylinders=mtcars$cyl, Gears=mtcars$gear),
            FUN=mean, na.rm=TRUE)
> aggdata
  Cylinders Gears  mpg disp  hp drat   wt qsec  vs   am carb
1         4     3 21.5  120  97 3.70 2.46 20.0 1.0 0.00 1.00
2         6     3 19.8  242 108 2.92 3.34 19.8 1.0 0.00 1.00
3         8     3 15.1  358 194 3.12 4.10 17.1 0.0 0.00 3.08
4         4     4 26.9  103  76 4.11 2.38 19.6 1.0 0.75 1.50
5         6     4 19.8  164 116 3.91 3.09 17.7 0.5 0.50 4.00
6         4     5 28.2  108 102 4.10 1.83 16.8 0.5 1.00 2.00
7         6     5 19.7  145 175 3.62 2.77 15.5 0.0 1.00 6.00
8         8     5 15.4  326 300 3.88 3.37 14.6 0.0 1.00 6.00
```

Пакет `dplyr` предлагает более естественный метод агрегирования данных. Взгляните на код в листинге 5.14.

Листинг 5.14. Агрегирование с помощью пакета dplyr

```
> mtcars %>%
    group_by(cyl, gear) %>%
    summarise_all(list(mean), na.rm=TRUE)
# Таблица данных: 8 x 11
# Группы:   cyl [3]
    cyl  gear   mpg  disp    hp  drat    wt  qsec    vs    am  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     4     3  21.5  120.   97   3.7   2.46  20.0   1    0     1
2     4     4  26.9  103.   76   4.11  2.38  19.6   1    0.75  1.5
3     4     5  28.2  108.  102   4.1   1.83  16.8   0.5  1     2
4     6     3  19.8  242.  108.  2.92  3.34  19.8   1    0     1
5     6     4  19.8  164.  116.  3.91  3.09  17.7   0.5  0.5   4
6     6     5  19.7  145   175   3.62  2.77  15.5   0    1     6
7     8     3  15.0  358.  194.  3.12  4.10  17.1   0    0     3.08
8     8     5  15.4  326   300.  3.88  3.37  14.6   0    1     6
```

Агрегированные переменные сохраняют свои имена и не дублируются в выходных данных. Мы еще вернемся к пакету `dplyr` и подробнее остановимся на его возможностях обобщения при обсуждении приемов обобщения и получения описательных статистик в главе 7.

Теперь у вас есть все инструменты, необходимые для преобразования данных в нужный вид. Мы готовы распрощаться с первой частью книги и перейти в волнующий мир анализа данных! В следующих главах мы начнем исследовать разнообразные статистические и графические методы извлечения информации из данных.

Итоги

- Язык R содержит сотни математических, статистических и вероятностных функций, которые с успехом можно использовать для обработки данных и применять к широкому спектру объектов данных, включая векторы, матрицы и таблицы данных.
- Условные операторы и циклы позволяют выполнять инструкции только при определенных условиях или многократно.
- R позволяет пользователям писать свои функции и тем самым расширять возможности их программ.
- Часто перед проведением дальнейшего анализа данные необходимо агрегировать и/или реструктурировать.

Часть II

Базовые методы

В первой части мы исследовали среду R и узнали, как получить данные из самых разных источников, агрегировать и подготовить эти данные к дальнейшему анализу. После импортирования данных в программу и подготовки к обработке наступает следующий этап – изучение каждой переменной в отдельности, чтобы узнать, как распределены значения каждой переменной. Это помогает узнать характеристики выборки, обнаружить неожиданные или проблемные значения и выбрать подходящие статистические методы. Затем, как правило, переменные изучаются попарно. При этом выявляются взаимосвязи между переменными и начинается построение более сложных моделей.

Вторая часть посвящена графическим и статистическим методам получения основных сведений о данных. В главе 6 описаны методы визуализации распределения отдельных переменных. Для категориальных данных это столбиковые и круговые диаграммы, а также относительно новые диаграммы типа «плоское дерево» (tree map). Для числовых переменных – гистограммы, диаграммы распределения плотности, коробчатые диаграммы (или диаграммы размахов), точечные диаграммы и менее известные «скрипич-

ные диаграммы». Все эти диаграммы нужны, чтобы понять, как распределены значения одной переменной.

В главе 7 описаны статистические методы вычисления обобщенных характеристик отдельных переменных и взаимосвязей между парами переменных. Глава начинается с описания способов определения обобщенных характеристик числовых данных (как для всего набора данных, так и для его части). Затем описывается построение сводных и частотных таблиц для обобщения категориальных данных. И в завершение главы обсуждаются основные приемы изучения взаимосвязей между двумя переменными, включая корреляцию, критерий хи-квадрат, критерий Стьюдента и непараметрические методы.

К концу второй части книги вы научитесь использовать основные графические и статистические методы, доступные в R, для описания своих данных, исследования различий между группами и выявления значимых взаимосвязей между переменными.

6 Базовые диаграммы

В этой главе:

- столбиковые, коробчатые и точечные диаграммы;
- круговые диаграммы и диаграммы вида «плоское дерево»;
- гистограммы и диаграммы распределения плотности.

Какие бы данные ни анализировались, прежде всего нужно *посмотреть* на них. Какие значения у каждой переменной встречаются чаще всего? Каков диапазон разброса данных? Есть ли какие-нибудь необычные наблюдения? В R реализовано множество функций для визуализации данных. В этой главе мы рассмотрим диаграммы, построенные на основе одной категориальной или непрерывной переменной и помогающие:

- увидеть характер распределения значений переменой;
- сравнить распределение переменной с двумя или более группами.

В обоих случаях переменная может быть непрерывной (например, расход топлива машиной в милях на галлон) или категориальной (например, результат лечения: отсутствует, незначительный, заметный). В следующих главах мы познакомимся с диаграммами, отображающими более сложные взаимосвязи между переменными.

Разделы этой главы посвящены столбиковым и круговым диаграммам, диаграммам вида «плоское дерево», гистограммам, диаграммам распределения плотности, коробчатым, скрипичным и точечным диаграммам. Какие-то из них могут быть уже знакомы вам, а какие-то (как, например, плоские деревья и скрипичные диаграммы) окажутся новыми. Нашей задачей будет, как всегда, лучше понять данные и поделиться своими открытиями с другими. Начнем со столбиковых диаграмм.

6.1. Столбиковые диаграммы

Столбиковые диаграммы (bar plot) отражают распределение (частоту разных значений) категориальной переменной в виде вертикальных или горизонтальных столбиков. Создать столбиковую диаграмму можно с помощью пакета `ggplot2`, используя такой код:

```
ggplot(data, aes(x=catvar) + geom_bar()
```

где *data* – таблица данных, а `catvar` – категориальная переменная.

В примерах, следующих ниже, демонстрируются результаты исследования нового метода лечения ревматоидного артрита. Данные хранятся в таблице `Arthritis`, распространяемой с пакетом `vcd`. Этот пакет не входит в базовый дистрибутив R, поэтому перед использованием его нужно скачать и установить, выполнив инструкцию `install.packages("vcd")`. Учтите, что пакет `vcd` не нужен для построения столбиковых диаграмм. Мы загружаем его, только чтобы получить доступ к таблице данных `Arthritis`.

6.1.1. Простые столбиковые диаграммы

В таблице данных `Arthritis` в переменной `Improved` указана реакция пациентов на плацебо или лекарство:

```
> data(Arthritis, package="vcd")
> table(Arthritis$Improved)

  None   Some Marked
    42     14     28
```

Здесь видно, что здоровье 28 пациентов заметно улучшилось (`Marked`), 14 пациентам стало немного лучше (`Some`) и 42 пациента так и не поправились (`None`). В главе 7 мы более подробно обсудим применение функции `table()` для подсчета значений в ячейках.

Результаты подсчетов можно графически отобразить в виде вертикальной или горизонтальной столбиковой диаграммы. Эту задачу решает программный код в листинге 6.1, а диаграмма, получившаяся в результате его выполнения, показана на рис. 6.1.

Листинг 6.1. Простые столбиковые диаграммы

```
library(ggplot2)
ggplot(Arthritis, aes(x=Improved)) + geom_bar() +          (1)
  labs(title="Simple Bar chart",                           (1)
       x="Improvement",                                    (1)
       y="Frequency")                                      (1)

ggplot(Arthritis, aes(x=Improved)) + geom_bar() +          (2)
  labs(title="Horizontal Bar chart",                       (2)
       x="Improvement",                                    (2)
       y="Frequency") +                                    (2)
  coord_flip()                                             (2)
```

(1) **Простая столбиковая диаграмма**

(2) **Горизонтальная столбиковая диаграмма**

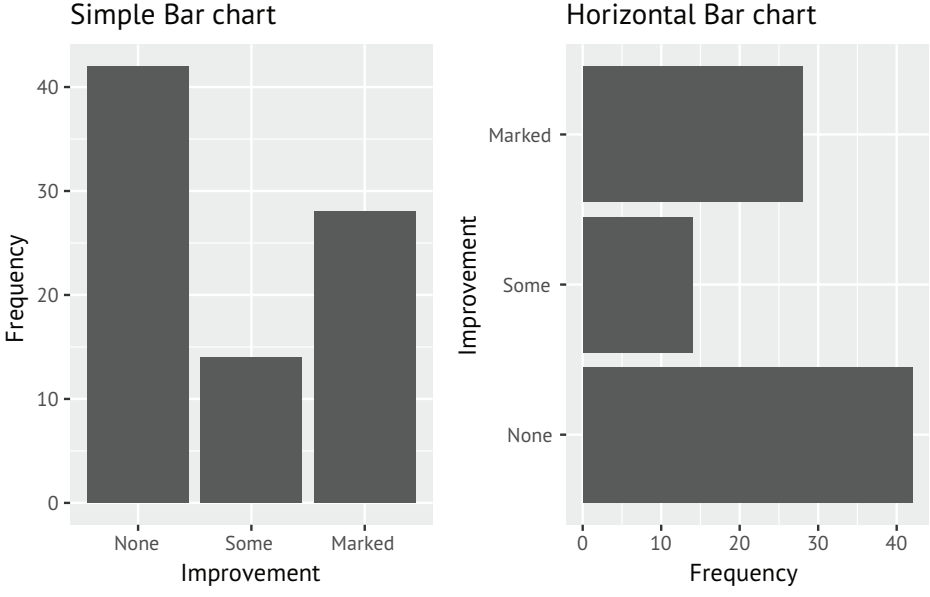


Рис. 6.1. Простые вертикальная и горизонтальная столбиковые диаграммы

Но что получится, если метки окажутся слишком длинными? В разделе 6.1.4 вы увидите, как настроить отображение меток, чтобы они не перекрывались на диаграмме.

6.1.2. Столбиковые диаграммы: составные, с группировкой и спинограммы

Главный вопрос в исследовании артрита: «Как степень улучшения состояния коррелирует с получением пациентами плацебо и ле-

карства?» Для этого можно создать сводную таблицу переменных с помощью функции `table()`:

```
> table(Arthritis$Improved, Arthritis$Treatment)

            Treatment
Improved Placebo Treated
  None        29      13
  Some         7       7
  Marked       7      21
```

Табличное представление, конечно, может пригодиться, но результаты будет проще понять, если изобразить их с помощью гистограммы. Отношения между двумя категориальными переменными можно изобразить с помощью столбиковых диаграмм *составных*, *с группировкой* или *спинограмм*. В листинге 6.2 приводится соответствующий код, а на рис. 6.2 показана получившаяся диаграмма.

Листинг 6.2. Столбиковые диаграммы составные, с группировкой и спинограммы

```
library(ggplot2)
ggplot(Arthritis, aes(x=Treatment, fill=Improved)) +      1
  geom_bar(position = "stack") +                          1
  labs(title="Stacked Bar chart",                         1
       x="Treatment",                                     1
       y="Frequency")                                     1

ggplot(Arthritis, aes(x=Treatment, fill=Improved)) +      2
  geom_bar(position = "dodge") +                          2
  labs(title="Grouped Bar chart",                         2
       x="Treatment",                                     2
       y="Frequency")                                     2

ggplot(Arthritis, aes(x=Treatment, fill=Improved)) +      3
  geom_bar(position = "fill") +                           3
  labs(title="Filled Bar chart",                          3
       x="Treatment",                                     3
       y="Proportion")                                    3
```

1 **Составная столбиковая диаграмма.**

2 **Столбиковая диаграмма с группировкой.**

3 **Спинограмма.**

В составной столбиковой диаграмме (stacked bar chart) каждый сегмент представляет частоту или долю случаев в данной комбинации Treatment (Лечение) – Placebo (плацебо) и Treated (лекарство) – и Improvement (Улучшение) – None (нет), Some (некоторое) и Marked (заметное). Сегменты откладываются отдельно для каждого способа лечения. Столбиковая диаграмма с группировкой

(grouped bar chart) позволяет сравнить степень улучшения для обоих способов лечения. Спинограмма (filled bar chart) – та же самая составная столбиковая диаграмма, масштаб которой изменен так, что высота каждого столбца равна 1, а высоты сегментов представляют их пропорции.

Спинограммы особенно удобны для сравнения пропорций категориальных переменных. Например, спинограмма на рис. 6.2 ясно показывает больший процент пациентов с заметным улучшением, получавших лекарство, по сравнению с пациентами, получавшими плацебо.

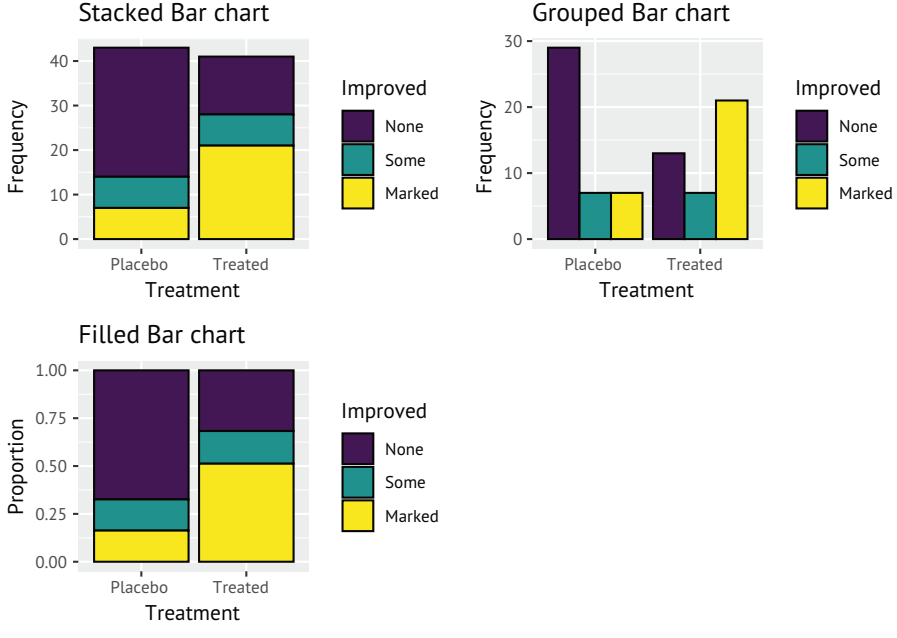


Рис. 6.2. Столбиковые диаграммы: составные, с группировкой и спинограммы

6.1.3. Столбиковые диаграммы средних значений

Столбиковая диаграмма не обязательно отражает количество или частоту значений. Столбиковые диаграммы можно строить на основе средних значений, медиан, процентилей, стандартных отклонений и т. д., обобщая данные с использованием соответствующих статистик и передавая результаты пакету `ggplot2`.

На следующем графике мы построим средний уровень неграмотности для регионов США в 1970 году. Распространяемый с R набор данных `state.x77` содержит уровни неграмотности по штатам, а набор данных `state.region` – названия регионов в каждом штате. В листинге 6.3 показан код, создающий диаграмму на рис. 6.3.

Листинг 6.3. Столбиковая диаграмма на основе отсортированных средних значений

```
> states <- data.frame(state.region, state.x77)
> library(dplyr)
> plotdata <- states %>%
    group_by(state.region) %>%                                    (1)
    summarize(mean = mean(Illiteracy))                            (1)
  plotdata                                                        (1)

# Таблица данных: 4×2
  state.region   mean
  <fct>         <dbl>
1 Northeast      1
2 South          1.74
3 North Central  0.7
4 West           1.02

> ggplot(plotdata, aes(x=reorder(state.region, mean), y=mean)) +  (2)
    geom_bar(stat="identity") +                                   (2)
    labs(x="Region",                                              (2)
         y="",                                                    (2)
         title = "Mean Illiteracy Rate")                          (2)
```

(1) **Генерируются средние значения по регионам**

(2) **Вывод столбиковой диаграммы на основе отсортированных средних значений**

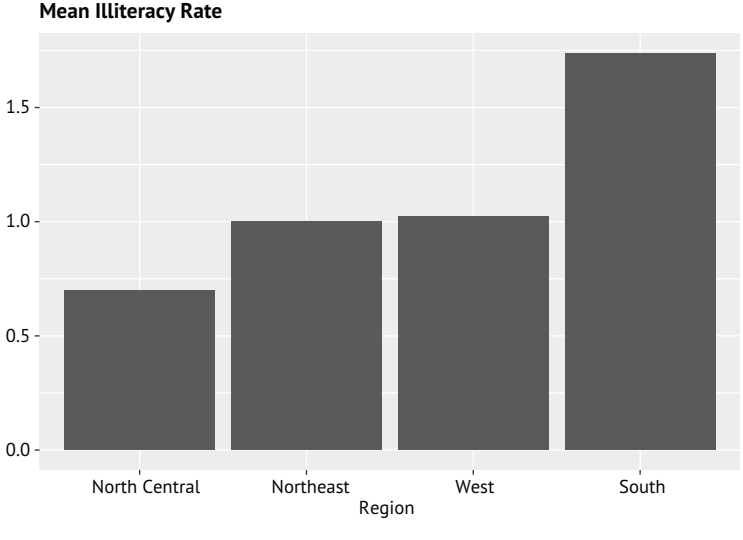


Рис. 6.3. Столбиковая диаграмма на основе отсортированных средних значений неграмотности в разных регионах США

Здесь сначала вычисляются средние уровни неграмотности для каждого региона (1). Затем полученные средние значения сортиру-

ются в порядке возрастания и отображаются в виде столбцов (2). Обычно функция `geom_bar()` вычисляет и отображает количества наблюдений, но добавление параметра `stat="identity"` заставляет ее отображать указанные числа (в данном случае средние значения). Функция `reorder()` используется для упорядочения столбцов по возрастанию среднего значения неграмотности.

При выводе на диаграммах обобщающих статистик, таких как средние значения, рекомендуется указывать степень их изменчивости. Одной из мер изменчивости является стандартная ошибка статистики – оценка ожидаемой дисперсии в гипотетических повторяющихся выборках. Код в листинге 6.4 выводит дополнительно размах погрешностей, используя в качестве меры стандартную ошибку среднего (рис. 6.4).

Листинг 6.4. Столбиковая диаграмма на основе отсортированных средних значений с интервалами стандартной ошибки

```
> plotdata <- states %>%                                          (1)
    group_by(state.region) %>%                                    (1)
    summarize(n=n(),                                              (1)
              mean = mean(Illiteracy),                            (1)
              se = sd(Illiteracy)/sqrt(n))                        (1)

> plotdata

# Таблица данных: 4×4
  state.region      n  mean     se
  <fct>         <int> <dbl>  <dbl>
1 Northeast         9  1    0.0928
2 South            16  1.74 0.138
3 North Central    12  0.7  0.0408
4 West             13  1.02 0.169

> ggplot(plotdata, aes(x=reorder(state.region, mean), y=mean)) +  (2)
    geom_bar(stat="identity", fill="skyblue") +
    geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=0.2) +   (3)
    labs(x="Region",
         y="",
         title = "Mean Illiteracy Rate",
         subtitle = "with standard error bars")
```

(1) **Генерируются средние значения и стандартные ошибки по регионам**

(2) **Вывод столбиковой диаграммы на основе отсортированных средних значений**

(3) **Вывод интервалов стандартной ошибки**

Здесь сначала для каждого региона вычисляются средние значения и стандартные ошибки (1). Затем полученные средние значения сортируются в порядке возрастания. Цвет изменился с темно-серого по умолчанию на более светлый (небесно-голубой),

чтобы «усы» (планки), отражающие погрешности, которые будут добавлены на следующем шаге, выделялись на их фоне (2). Наконец, в диаграмму добавляются планки погрешностей (3). Параметр `width` в функции `geom_errorbar()` управляет горизонтальной шириной отображаемых планок погрешностей и служит чисто эстетическим целям – он не имеет статистического значения. На рис. 6.4 можно видеть, что среднее значение уровня неграмотности для Северо-Центрального (North Central) региона не только самое низкое, но и самое надежное (имеет наименьшую изменчивость), а среднее значение для Западного (West) региона является наименее надежным (имеет наибольшую изменчивость).

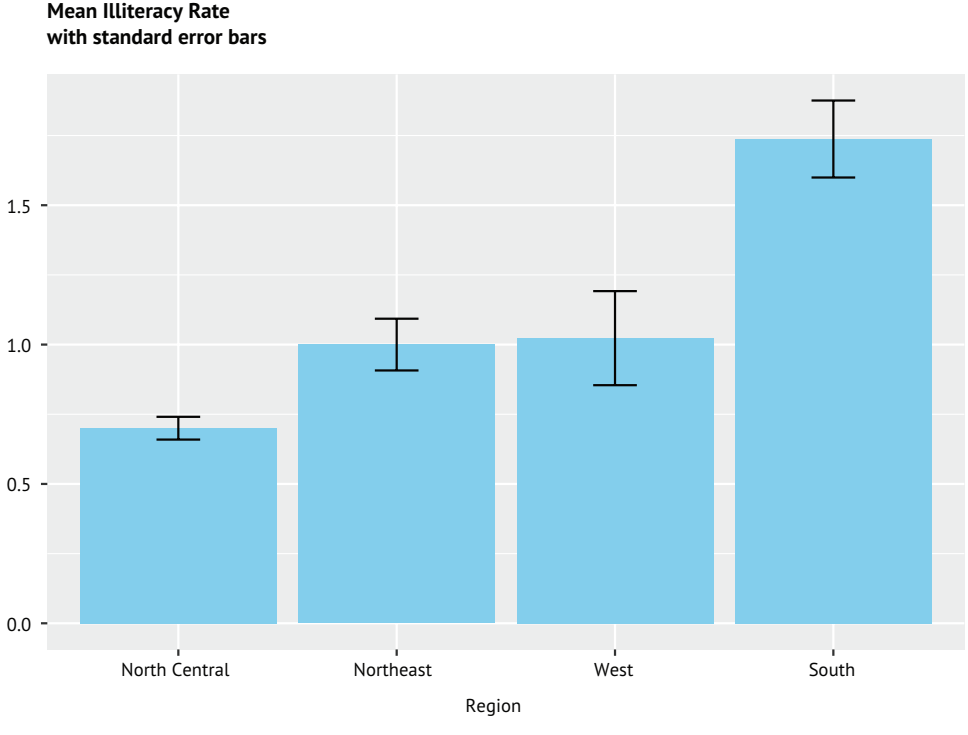


Рис. 6.4. Столбиковая диаграмма среднего уровня неграмотности в регионах США. К каждому столбику добавлена планка, отражающая стандартную ошибку среднего

6.1.4. Настройка столбиковых диаграмм

Столбиковые диаграммы имеют несколько параметров, управляющих внешним видом, которые можно настроить. Чаще всего настраиваются цвет и подписи. Мы рассмотрим оба по очереди.

Цвета

Пользователь имеет возможность назначать свои цвета заливки и рамок. Функция `geom_bar()` принимает параметр `fill="цвет"` – цвет заливки, а `color="цвет"` – цвет рамок.

Параметры fill и color

Многие функции в пакете ggplot2 принимают параметр fill, определяющий цвет площадных геометрических объектов (таких как столбики, секторы круга, прямоугольники), и параметр color, определяющий цвет геометрических объектов, не имеющих площади (линий, точек и рамок).

Например, следующий код:

```
data(Arthritis, package="vcd")
ggplot(Arthritis, aes(x=Improved)) +
   geom_bar(fill="gold", color="black") +
   labs(title="Treatment Outcome")
```

создаст диаграмму, изображенную на рис. 6.5.

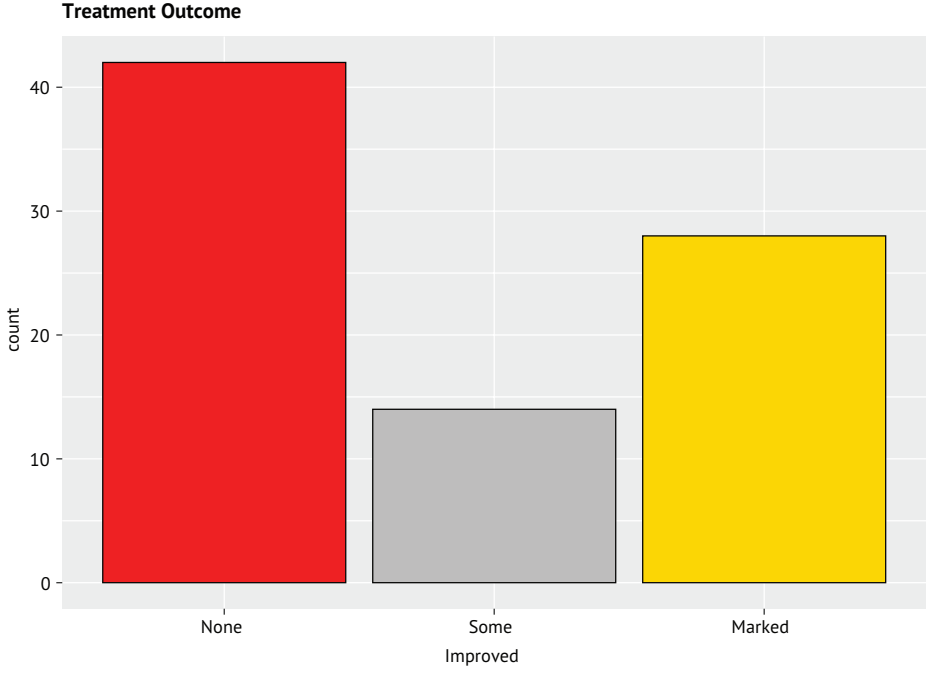


Рис. 6.5. Столбиковая диаграмма с настроенными цветами заливки и рамок

В предыдущем примере каждый столбик окрашивался в один цвет. Однако есть возможность назначать цвета разным уровням категориальной переменной. Например, следующий код:

```
ggplot(Arthritis, aes(x=Treatment, fill=Improved)) +
  geom_bar(position = "stack", color="black") +
  scale_fill_manual(values=c("red", "grey", "gold")) +
  labs(title="Stacked Bar chart",
       x="Treatment",
       y="Frequency")
```

создаст диаграмму, изображенную на рис. 6.6.

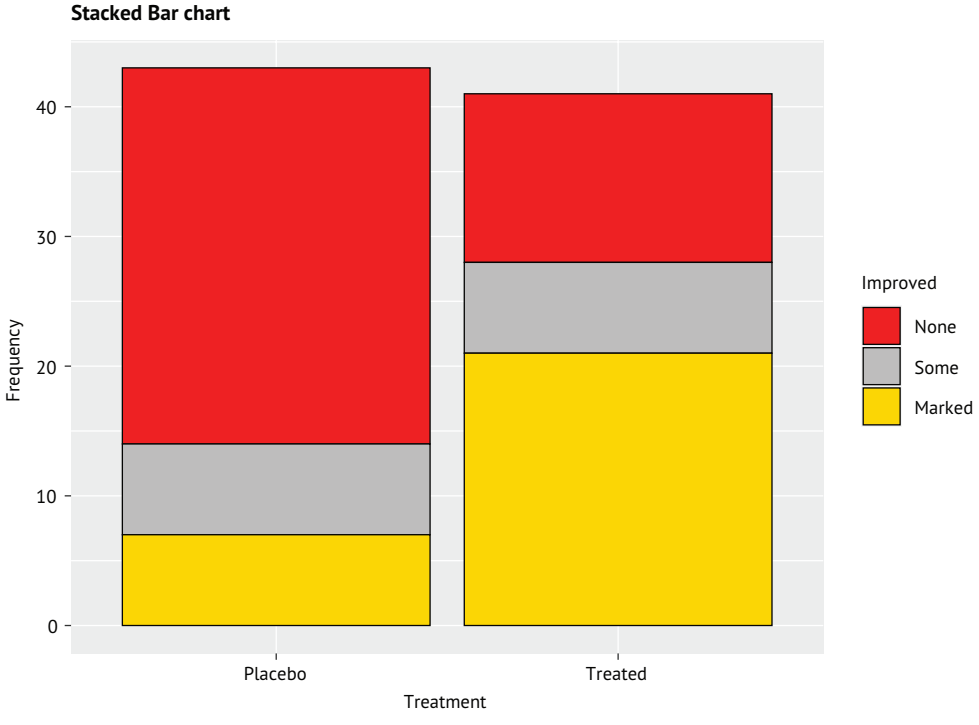


Рис. 6.6. Составная столбиковая диаграмма с настроенными цветами заливки для разных уровней переменной Improvement

Здесь цвета заливки столбиков зависят от уровней переменной `Improved`. Функция `scale_fill_manual()` назначает красный цвет для значения `None`, серый – для значения `Some` и желтый – для значения `Marked`. Список с названиями цветов можно найти по адресу: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf. В главе 19 мы обсудим другие методы назначения цветов.

Подписи

Когда на диаграмме присутствует много столбцов, их подписи могут перекрываться. Рассмотрим пример. Набор данных `mpg` в пакете `ggplot2` описывает расход топлива для 38 популярных моделей автомобилей, выпускавшихся с 1999 по 2008 год. Все модели характеризуются набором параметров (тип трансмиссии, количество цилиндров в двигателе и т. д.). Допустим, мы решили подсчитать количество экземпляров каждой модели, включенных в набор данных. Следующий код:

```
ggplot(mpg, aes(x=model)) +
   geom_bar() +
   labs(title="Car models in the mpg dataset",
        y="Frequency", x="")
```

создаст диаграмму, изображенную на рис. 6.7.

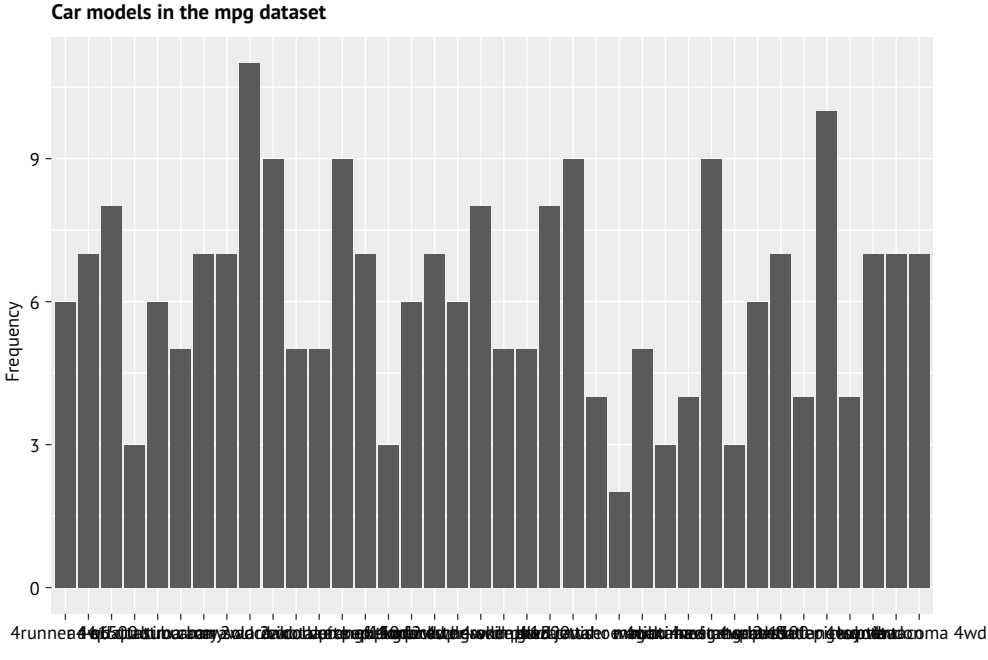


Рис. 6.7. Столбиковая диаграмма с перекрывающимися метками

Даже я в своих очках не могу это прочитать. Две простые настройки сделают подписи читабельными. Во-первых, данные можно представить в виде горизонтальной столбиковой диаграммы (рис. 6.8).

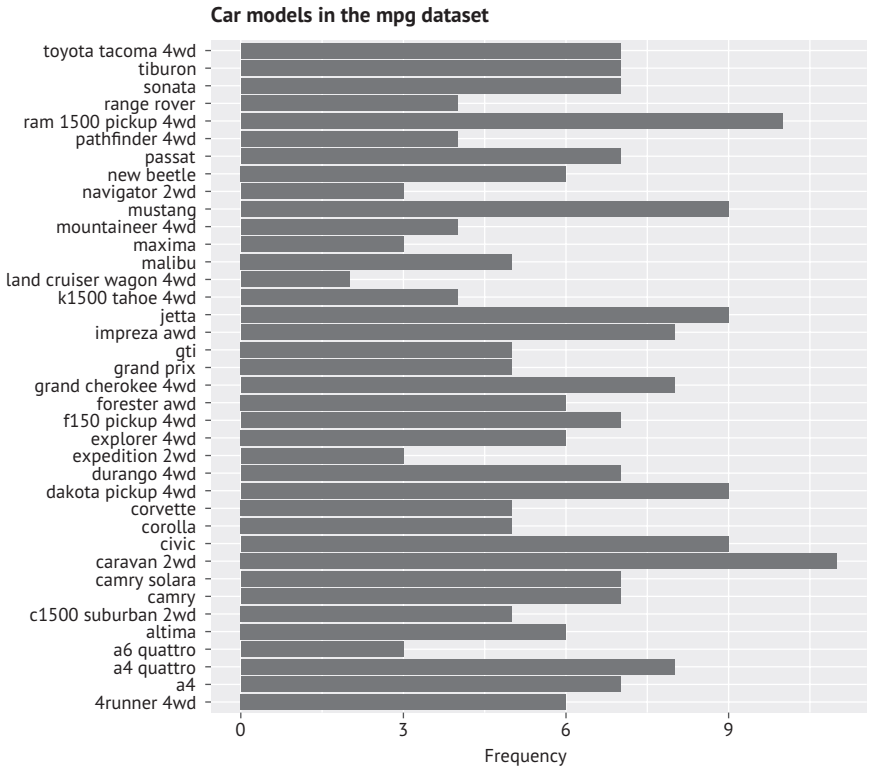


Рис. 6.8. Горизонтальная столбиковая диаграмма

Во-вторых, можно повернуть подписи на некоторый угол и использовать шрифт меньшего размера (рис. 6.9):

```
ggplot(mpg, aes(x=model)) +
   geom_bar() +
   labs(title="Model names in the mpg dataset",
        y="Frequency", x="") +
   theme(axis.text.x = element_text(angle = 45, hjust = 1, size=8))
```

В главе 19 мы более подробно обсудим функцию `theme()`. Наряду со столбиковыми диаграммами для визуализации категориальных данных не менее часто используются круговые диаграммы. Рассмотрим их далее.

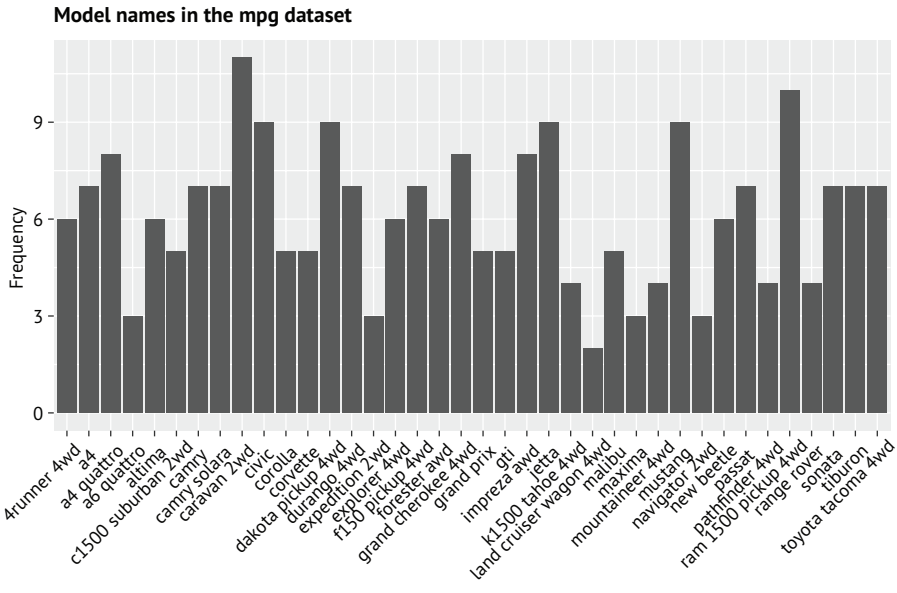


Рис. 6.9. Столбиковая диаграмма с подписями, повернутыми на некоторый угол и с уменьшенным шрифтом

6.2. Круговые диаграммы

Несмотря на то что круговые диаграммы широко используются в деловом мире, они критикуются большинством статистиков, включая авторов документации R. Они рекомендуют применять столбиковые или точечные диаграммы вместо круговых, потому что людям легче сравнивать длины, чем площади и объемы. Возможно, именно из этих соображений круговые диаграммы имеют довольно ограниченную поддержку в R по сравнению с другими статистическими программами.

Однако иногда круговые диаграммы могут быть по-настоящему полезными. В частности, они хорошо передают отношения часть–целое. Например, круговую диаграмму можно использовать для отображения процента штатных преподавателей-женщин в университете.

Создать круговую диаграмму в R можно с помощью функции `pie()`, но, как я уже сказал, эта функция имеет довольно ограниченные возможности, а создаваемые ею диаграммы смотрятся непривлекательно. Чтобы решить эту проблему, я создал пакет `ggpie`, позволяющий создавать разнообразные круговые диаграммы с помощью `ggplot2` (только, пожалуйста, не шлите мне гневных писем!). Его можно установить из моего репозитория на GitHub, как показано ниже:

```
if(!require(remotes)) install.packages("remotes")
remotes::install_github("rkabacoff/ggpie")
```

Базовый синтаксис функции:

```
ggpie(data, x, by, offset, percent, legend, title)
```

где

- `data` – таблица данных;
- `x` – категориальная переменная для отображения на диаграмме;
- `by` – необязательная вторая категориальная переменная, если имеется; для каждого уровня этой переменной будет построена отдельная круговая диаграмма;
- `offset` – расстояние между подписями секторов круговой диаграммы и началом координат. Значение `0.5` поместит подписи в центры секторов, а значения больше `1.0` – за пределами секторов;
- `percent` – логическое значение: `FALSE` подавляет вывод символа процента;
- `legend` – логическое значение: `FALSE` подавляет вывод легенды и добавляет подписи к секторам на самой диаграмме;
- `title` – необязательный заголовок.

Используя дополнительные параметры (описаны на веб-сайте проекта `ggpie`), можно настроить внешний вид круговой диаграммы в довольно широких пределах. Давайте создадим круговую диаграмму, отображающую распределение автомобилей по классам в таблице данных `mpg`:

```
library(ggplot2)
library(ggpie)
ggpie(mpg, class)
```

Результат показан на рис. 6.10. Как показывает диаграмма, 26 % автомобилей относятся к классу кроссоверов и только 2 % – к классу двухместных автомобилей.

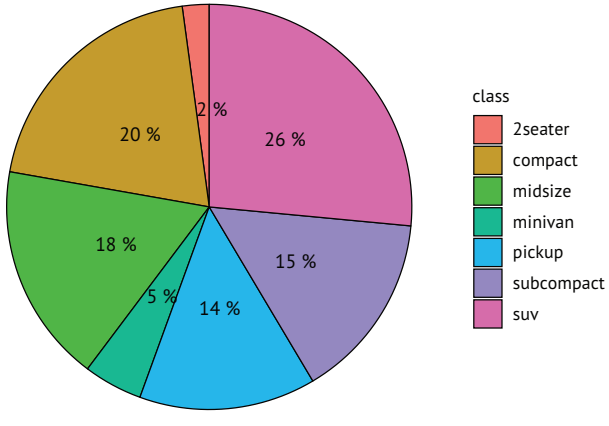


Рис. 6.10. Круговая диаграмма, отображающая распределение автомобилей по классам

Следующий код выводит (рис. 6.11) круговую диаграмму без легенды и с подписями для каждого сектора. Кроме того, подписи вынесены за пределы секторов, а также добавлен заголовок:

```
ggpie(mpg, class, legend=FALSE, offset=1.3,
        title="Automobiles by Car Class")
```

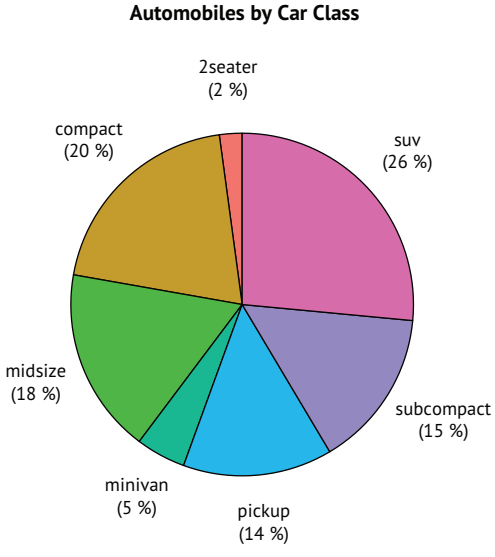


Рис. 6.11. Круговая диаграмма с подписями за пределами секторов

В заключительном примере (рис. 6.12) показано распределение автомобилей по классам в разные годы.

```
ggpie(mpg, class, year,
      legend=FALSE, offset=1.3, title="Car Class by Year")
```

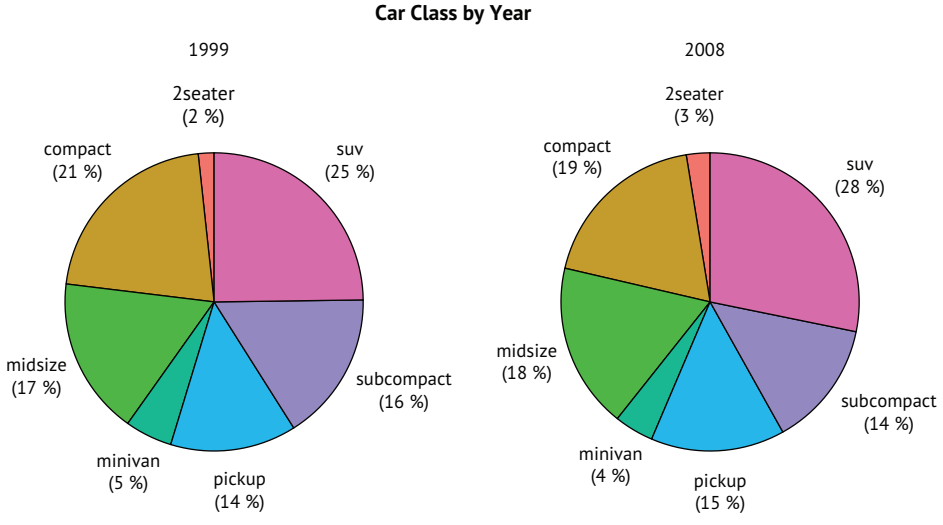


Рис. 6.12. Круговая диаграмма распределения автомобилей по классам в разные годы

Как видите, в период с 1999 по 2008 год распределение автомобилей по классам оставалось довольно постоянным. Пакет `ggpie` может создавать более сложные круговые диаграммы, настраиваемые в широких пределах. Подробности ищите в документации (https://rkabacoff.github.io/ggpie).

6.3. Диаграммы «плоское дерево»

Круговые диаграммы имеют альтернативу – диаграммы вида «плоское дерево» (tree map), отображающие распределение категориальной переменной с использованием прямоугольников с площадями, пропорциональными уровням переменных. Я покажу, как создавать диаграммы «плоское дерево» с помощью пакета `treemapify`. Обязательно установите его, прежде чем продолжить (`install.packages("treemapify")`).

Начнем с создания диаграммы «плоское дерево», отображающей распределение производителей автомобилей в таблице данных mpg. В листинге 6.5 показан код, а на рис. 6.13 – созданная им диаграмма.

Листинг 6.5. Простая диаграмма «плоское дерево»

```
library(ggplot2)
library(dplyr)
library(treemapify)

plotdata <- mpg %>% count(manufacturer)      (1)
```

```
ggplot(plotdata,                                    (2)
       aes(fill = manufacturer,                     (2)
           area = n,                                (2)
           label = manufacturer)) +                 (2)
geom_treemap() +                                    (2)
geom_treemap_text() +                               (2)
theme(legend.position = "none")                     (2)
```

(1) Получение обобщенных данных

(2) Создание диаграммы «плоское дерево»

Здесь сначала вычисляется количество автомобилей, произведенных каждым производителем, на основе переменной `manufacturer` (1). Затем эта информация передается в `ggplot2` для создания диаграммы (2). В вызове функции `aes()` в параметре `fill` указана категориальная переменная, в параметре `area` – число производителей, а в необязательном параметре `label` – переменная, которая должна использоваться для создания подписей ячеек. Функция `geom_treemap()` создает диаграмму, а функция `geom_treemap_text()` добавляет подписи к ячейкам. Функция `theme()` используется для подавления вывода легенды, которая здесь не нужна, потому что все ячейки и без того подписаны.

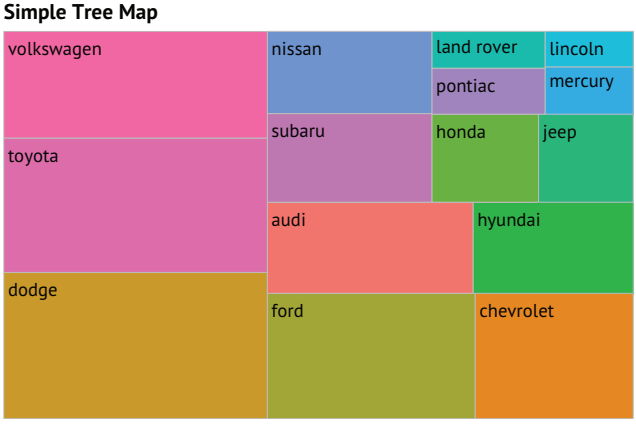


Рис. 6.13. Диаграмма «плоское дерево», отображающая доли производителей автомобилей в наборе данных mpg. Размеры прямоугольников пропорциональны количествам автомобилей, произведенных производителями

Как видите, диаграммы «плоское дерево» можно использовать для визуализации категориальных переменных с *несколькими* уровнями (в отличие от круговых диаграмм). В следующем примере добавлена вторая переменная – `drivetrain`. На ней автомобили делятся не только по производителям, но также по типу привода: на переднеприводные (front-wheel), заднеприводные (rear-wheel) и полноприводные (four-wheel). Код показан в листинге 6.6, а сама диаграмма – на рис. 6.14.

Листинг 6.6. Диаграмма «плоское дерево» с дополнительной разбивкой на подгруппы

```
library(ggplot2)
library(dplyr)
library(treemapify)

plotdata <- mpg %>%                                                    (1)
  count(manufacturer, drv)                                             (1)
  plotdata$drv <- factor(plotdata$drv,                                 (2)
                         levels=c("4", "f", "r"),                      (2)
                         labels=c("4-wheel", "front-wheel", "rear"))   (2)

ggplot(plotdata,                                                       (3)
       aes(fill = manufacturer,                                        (3)
           area = n,                                                   (3)
           label = manufacturer,                                       (3)
           subgroup=drv)) +                                            (3)
  geom_treemap() +                                                     (3)
  geom_treemap_subgroup_border() +                                     (3)
  geom_treemap_subgroup_text(                                          (3)
    place = "middle",                                                  (3)
    colour = "black",                                                  (3)
    alpha = 0.5,                                                       (3)
    grow = FALSE) +                                                    (3)
  geom_treemap_text(colour = "white",                                  (3)
                    place = "centre",                                  (3)
                    grow=FALSE) +                                      (3)
  theme(legend.position = "none")                                      (3)
```

(1) **Вычисляется количество ячеек**

(2) **Замена подписей для уровней переменной** `drivetrains`

(3) **Создание диаграммы «плоское дерево»**

Сначала определяются количества автомобилей для разных комбинаций производитель/привод (1). Затем для уровней переменной `drivetrain` (2) определяются более подходящие подписи. Полученная в результате таблица данных передается пакету `ggplot2` для создания диаграммы (3). Параметр `subgroup` в функции `aes()` создает отдельные поддиаграммы для каждого типа привода. Функции `geom_treemap_border()` и `geom_treemap_subgroup_text()` добавляют рамки и подписи для подгрупп соответственно. Параметры в каждой из этих функций определяют внешний вид диаграммы. Подписи, соответствующие подгруппам, размещаются по центру и выводятся полупрозрачным шрифтом (`alpha=0.5`). Размер шрифта подписей остается постоянного размера и не увеличивается для заполнения всей области (`grow=FALSE`). Подписи ячеек выводятся белым шрифтом по центру каждой ячейки и тоже не увеличиваются для заполнения всей ячейки.

На диаграмме (рис. 6.14) отчетливо видно, например, что Hyundai выпускает автомобили с передним приводом, но не выпускает автомобилей с задним или полным приводом. Основную массу автомобилей с задним приводом производят Ford и Chevrolet. Больше всего полноприводных автомобилей произведено компанией Dodge.

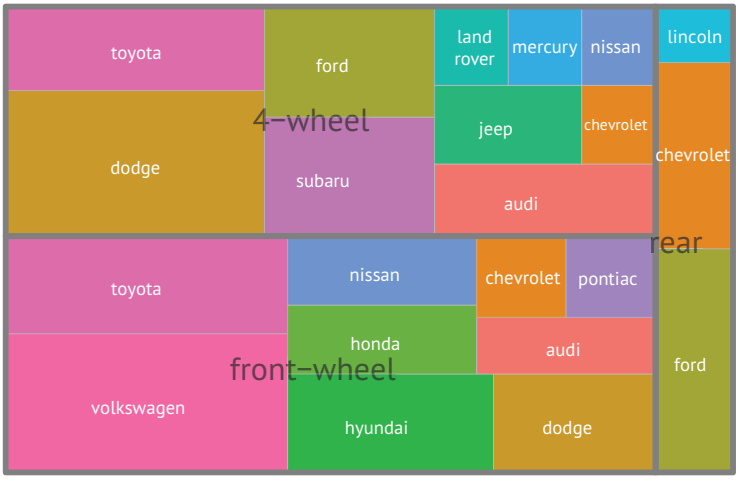


Рис. 6.14. Диаграмма «плоское дерево», отображающая распределение автомобилей по производителям и типам привода

Теперь перейдем к гистограммам. В отличие от столбиковых, круговых и древовидных диаграмм, гистограммы описывают распределение значений непрерывной переменной.

6.3. Гистограммы

Гистограммы отображают распределение значений непрерывных переменных, разделяя диапазон значений на заданное число отрезков по оси *x* и отображая частоту значений внутри каждого отрезка на оси *y*. Гистограмму можно создать при помощи функции:

```
ggplot(data, aes(x = contvar)) + geom_histogram()
```

где `data` – это таблица данных, а `contvar` – непрерывная переменная. Давайте снова используем наборы данных из пакета `ggplot` и изучим распределение расхода топлива в милях на галлон в городском цикле (`cty`) для 117 типов автомобилей в 2008 году. Код в листинге 6.7 создает четыре гистограммы, а на рис. 6.15 показаны получившиеся диаграммы.

Листинг 6.7. Гистограммы

```
library(ggplot2)
library(scales)

data(mpg)
cars2008 <- mpg[mpg$year == 2008, ]

ggplot(cars2008, aes(x=cty)) +                                    (1)
   geom_histogram() +                                             (1)
   labs(title="Default histogram")                                (1)

ggplot(cars2008, aes(x=hwy)) +                                    (2)
   geom_histogram(bins=20, color="white", fill="steelblue") +     (2)
   labs(title="Colored histogram with 20 bins",                   (2)
       x="City Miles Per Gallon",                                 (2)
       y="Frequency")

ggplot(cars2008, aes(x=hwy, y=..density..)) +                     (3)
   geom_histogram(bins=20, color="white", fill="steelblue") +     (3)
   scale_y_continuous(labels=scales::percent) +                   (3)
  labs(title="Histogram with percentages",                        (3)
       y= "Percent".                                              (3)
       x="City Miles Per Gallon")                                 (3)

ggplot(cars2008, aes(x=cty, y=..density..)) +                     (4)
   geom_histogram(bins=20, color="white", fill="steelblue") +     (4)
   scale_y_continuous(labels=scales::percent) +                   (4)
   geom_density(color="red", size=1) +                            (4)
   labs(title="Histogram with density curve",                     (4)
        y="Percent" ,                                             (4)
       x="Highway Miles Per Gallon")                              (4)
```

(1) **Простая гистограмма**

(2) **Цветная гистограмма с 20 столбиками**

(3) **Гистограмма с процентами**

(4) **Гистограмма с кривой плотности**

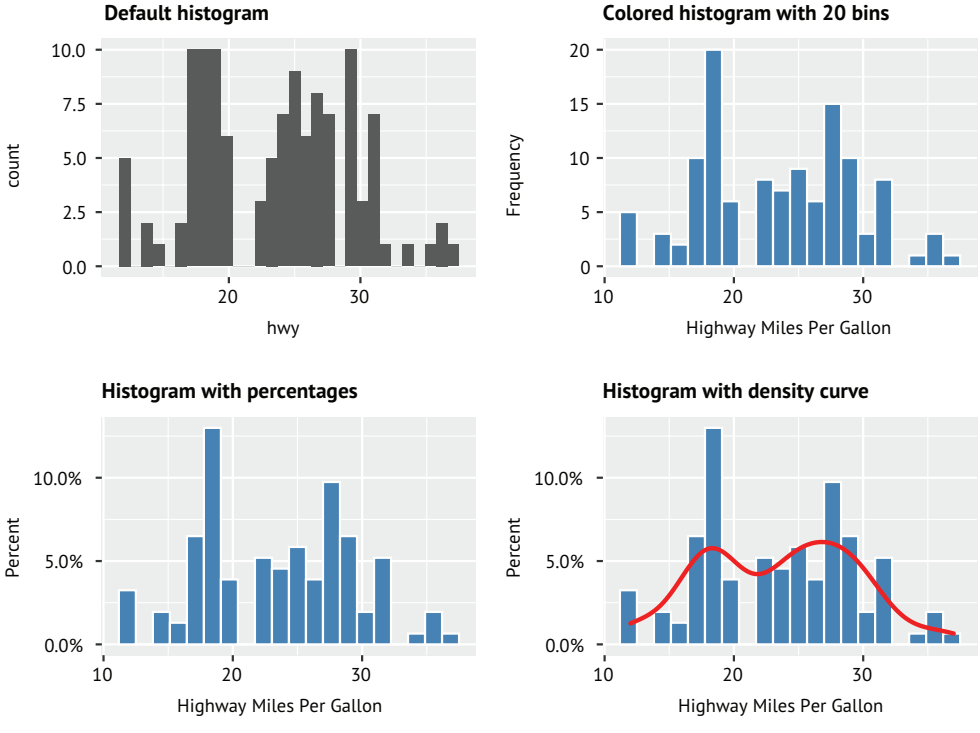


Рис. 6.15. Примеры гистограмм

Первая гистограмма (1) – это то, что получается по умолчанию, когда функция `ggplot()` вызывается без дополнительных параметров. В этом случае диапазон значений разбивается на 30 диапазонов. Для второй гистограммы (2) заданы 20 диапазонов, серо-голубой цвет для столбиков и белый цвет для рамки. Дополнительно определены более информативные подписи. Количество диапазонов сильно влияет на внешний вид гистограммы. Поэтому я рекомендую поэкспериментировать со значением `bins`, чтобы подобрать значение, хорошо отражающее распределение. При разбивке распределения на 20 диапазонов можно видеть два пика: один около расхода 13 миль на галлон, а другой – около 20,5 мили на галлон.

Третья гистограмма (3) выводит данные в форме процентов вместо частот. Это достигается связыванием внутренней переменной `..density..` с осью *y*. Для преобразования данных для оси *y* в проценты используется пакет `scales`. Обязательно установите его (`install.packages("scales")`), прежде чем запускать этот фрагмент кода.

Четвертая гистограмма (4) похожа на предыдущую, но добавляет кривую плотности. Кривая плотности – это ядерная оценка функции плотности, которая описана в следующем разделе. Она позволяет изобразить распределение значений в сглаженном виде. Параметры

функции `geom_density()`, которая строит эту кривую, задают красный цвет и толщину чуть больше толщины по умолчанию. Кривая плотности тоже показывает наличие двух пиков в распределении.

6.5. Диаграммы ядерной оценки функции плотности

В предыдущем разделе вы видели диаграмму ядерной оценки функции плотности, добавленную к гистограмме. С технической точки зрения ядерная оценка функции плотности – это непараметрический метод оценки плотности вероятности случайной переменной. По сути, он состоит в создании сглаженной гистограммы с площадью под кривой, равной 1. Обсуждение математического аппарата выходит за рамки этой книги, поэтому отмечу лишь, что в целом ядерная оценка функции плотности – хороший способ изобразить распределение значений непрерывной переменной. Вот общий синтаксис построения кривой плотности:

```
ggplot(data, aes(x = contvar)) + geom_density()
```

где `data` – таблица данных, а `contvar` – непрерывная переменная. Давайте вернемся к примеру построения диаграммы распределения расхода топлива в милях на галлон в городском цикле (`cty`) разными автомобилями в 2008 году. Код в листинге 6.8 строит три кривые ядерной оценки функции плотности, а на рис. 6.16 показаны получившиеся диаграммы.

Листинг 6.8. Диаграммы ядерной оценки функции плотности

```
library(ggplot2)
data(mpg)
cars2008 <- mpg[mpg$year == 2008, ]

ggplot(cars2008, aes(x=cty)) +                    1
   geom_density() +                               1
   labs(title="Default kernel density plot")      1

ggplot(cars2008, aes(x=cty)) +                    2
   geom_density(fill="red") +                     2
   labs(title="Filled kernel density plot",       2
        x="City Miles Per Gallon)                 2

> bw.nrd0(cars2008$cty)                           3
1.408                                             3

ggplot(cars2008, aes(x=cty)) +                    4
   geom_density(fill="red", bw=.5) +              4
   labs(title="Kernel density plot with bw=0.5",  4
        x="City Miles Per Gallon")                4
```

1 График плотности по умолчанию

2 График плотности с заливкой

3 Выводит полосу пропускания, используемую по умолчанию для оценки плотности ядра

4 График плотности с уменьшенной полосой пропускания

Первым выводится график плотности по умолчанию 1. Во втором примере область под кривой залита красным цветом. Гладкость кривой определяется параметром полосы пропускания `bw`, который вычисляется на основе отображаемых данных 2. Вызов `bw.nrd0(cars2008$cty)` выводит это значение (`1.408`) 3. Чем больше полоса пропускания, тем более гладкой получается кривая. Меньшее значение даст менее пологую кривую. В третьем примере используется меньшая полоса пропускания (`bw=0.5`), что позволяет увидеть больше деталей 4. Как и в случае с параметром `bins`, рекомендуется попробовать несколько значений пропускной способности, чтобы увидеть, какое поможет более эффективно визуализировать данные.

Диаграммы ядерной оценки функции плотности можно использовать для сравнения групп. Этот подход редко используется на практике, но это обусловлено не низкой эффективностью метода, а, вероятно, отсутствием доступного программного обеспечения. К счастью, пакет `ggplot2` прекрасно восполняет данный недостаток.

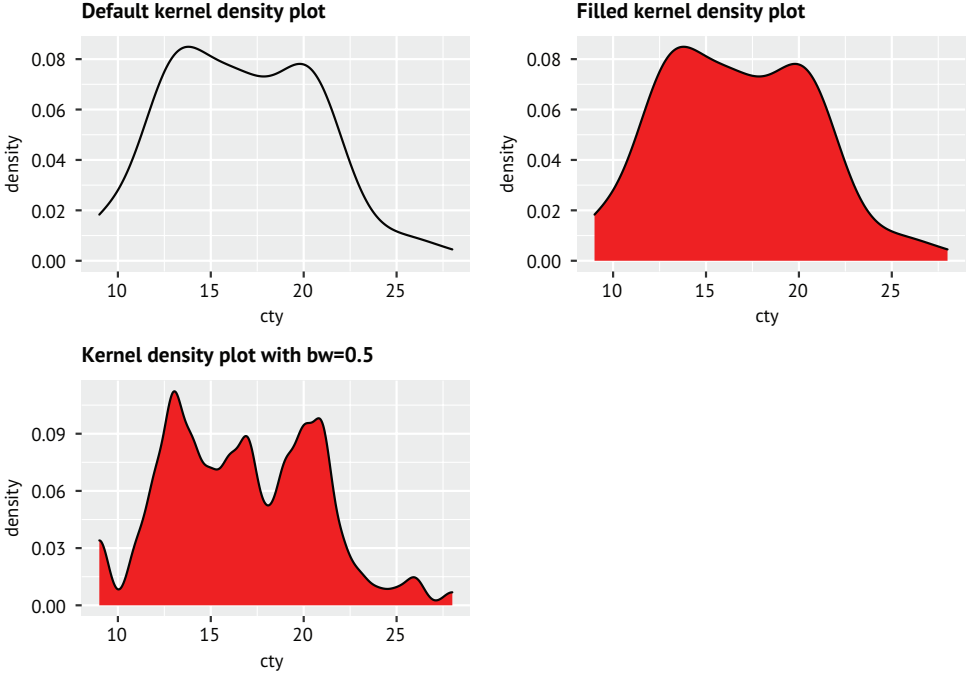


Рис. 6.16. Диаграммы ядерной оценки функции плотности

В этом примере сравниваются расходы топлива в городском цикле за 2008 год для автомобилей с четырьмя, шестью и восемью цилиндрами. Автомобилей с 5 цилиндрами лишь несколько штук, поэтому мы исключим их из анализа. Код реализации показан в листинге 6.9, а на рис. 6.17 и 6.18 – получившиеся диаграммы.

Листинг 6.9. Сравнительные графики ядерных функций плотности

```
data(mpg, package="ggplot2")                                          (1)
cars2008 <- mpg[mpg$year == 2008 & mpg$cyl != 5,]                     (1)
cars2008$Cylinders <- factor(cars2008$cyl)                            (1)

ggplot(cars2008, aes(x=cty, color=Cylinders, linetype=Cylinders)) +   (2)
  geom_density(size=1)  +                                             (2)
  labs(title="Fuel Efficiecy by Number of Cylinders",                 (2)
       x = "City Miles per Gallon")                                   (2)

ggplot(cars2008, aes(x=cty, fill=Cylinders)) +
  geom_density(alpha=.4) +                                            (3)
  labs(title="Fuel Efficiecy by Number of Cylinders",                 (3)
       x = "City Miles per Gallon")                                   (3)
```

(1) **Подготовка данных**

(2) **Вывод кривых плотности**

(3) **Вывод кривых плотности с заливкой**

Сначала код загружает новую копию данных и сохраняет данные за 2008 год для автомобилей с четырьмя, шестью и восемью цилиндрами (1). Количество цилиндров (`cyl`) сохраняется как категориальный фактор (`Cylinders`). Это обязательный шаг, потому что `ggplot2` ожидает получить категориальную переменную для группировки (а `cyl` – это непрерывная переменная).

Для каждого уровня переменной `Cylinders` строится своя кривая плотности (2). Цвет (красный, зеленый, синий) и тип линий (сплошная, точечная, пунктирная) тоже зависят от количества цилиндров. Наконец, тот же график создается с заливкой пространства под кривыми (3). Здесь добавляется прозрачность (`alpha=0.4`), потому что области заливки перекрываются, а нам хотелось бы видеть каждую из них.

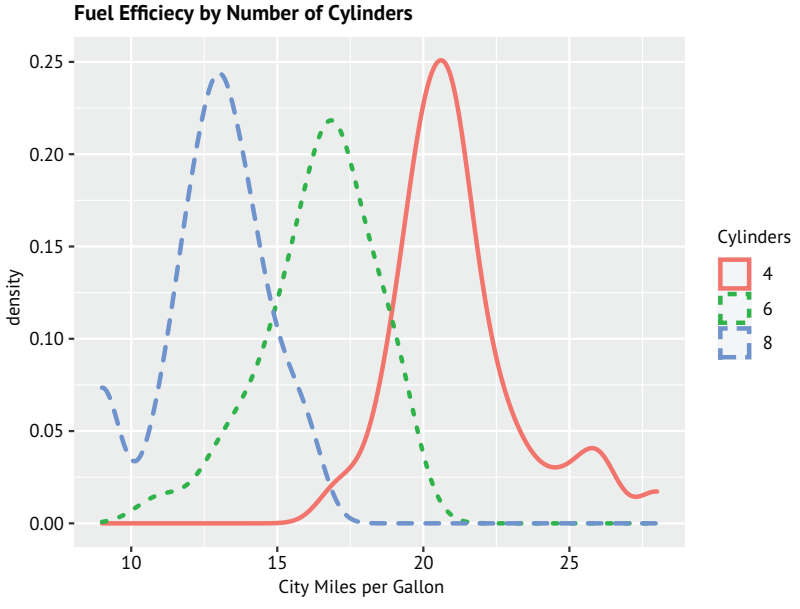


Рис. 6.17. Кривые ядерной плотности расхода топлива в городском цикле в зависимости от количества цилиндров

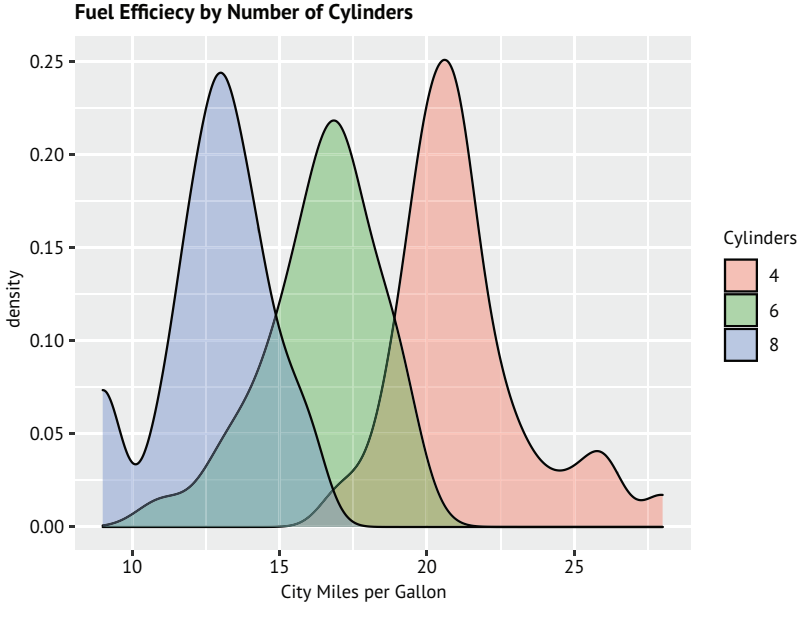


Рис. 6.18. Кривые ядерной плотности расхода топлива в городском цикле в зависимости от количества цилиндров с заливкой

Вывод в черно-белом виде

По умолчанию `ggplot2` выбирает цвета, трудно различимые при черно-белой печати. Это можно видеть на примере рис. 6.18 в печатной версии данной книги. Когда диаграммы должны печататься в черно-белом виде, можно добавить в код вызовы функций `scale_fill_grey()` и `scale_color_grey()`. Они создадут цветовую схему, хорошо различимую при черно-белой печати. Также можно задать цвета с помощью функции `bpy.colors()` из пакета `sp`. Она выберет сине-розово-желтую цветовую схему, которая тоже дает хорошо различимые графики как при цветной, так и при черно-белой печати. Но в последнем случае вы должны любить синий, розовый и желтый цвета!

Диаграммы с перекрывающимися графиками ядерной плотности помогают эффективно сравнивать группы наблюдений по заданной переменной. Используя этот подход, вы можете видеть как формы распределений, так и долю перекрытий между группами. (Мораль этой истории: в моей следующей машине будет четыре цилиндра или она вообще будет электрической.)

Коробчатые диаграммы (диаграммы размахов) – еще один замечательный (и гораздо более широко используемый) подход к визуализации распределения значений и различий между группами наблюдений. Мы обсудим их далее.

6.6. Коробчатые диаграммы

Коробчатые диаграммы (диаграммы размахов) иллюстрируют распределение значений непрерывной переменной, отображая пять параметров: минимум, нижний квартиль (25-й процентиль), медиану (50-й процентиль), верхний квартиль (75-й процентиль) и максимум. На этой диаграмме также могут быть отображены вероятные выбросы (значения, выходящие за диапазон в ±1.5 межквартильного размаха, разности верхнего и нижнего квартилей). Например, следующий код создаст диаграмму, изображенную на рис. 6.19:

```
ggplot(mtcars, aes(x="", y=mpg)) +
  geom_boxplot() +
  labs(y = "Miles Per Gallon", x="", title="Box Plot")
```

Я добавил подписи к элементам диаграммы вручную. По умолчанию каждый «ус» простирается до минимального или максимального значения, которое не выходит за пределы 1.5 межквартильного размаха. Выходящие за эти пределы значения отмечаются точками.

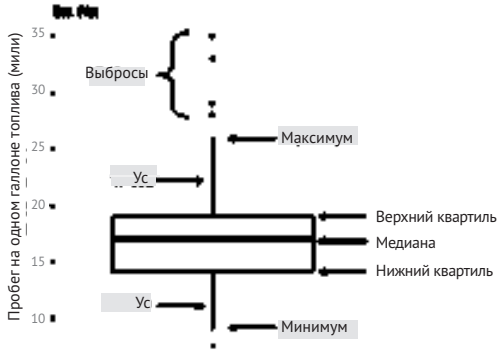


Рис. 6.19. Коробчатая диаграмма («с усами») с добавленными вручную подписями

Например, в нашей выборке с данными об автомобилях медиана расхода топлива равна 17 милям на галлон, 50 % значений попадают в диапазон между 14 и 19, наименьшее значение равно 9, а наибольшее – 35. Как это я смог получить значения из диаграммы с такой точностью? Функция `boxplot.stats(mtcars$mpg)` выводит значения статистик, которые использовались для построения диаграммы (иначе говоря, я схитрил). Здесь наблюдается четыре выброса (выше максимума, самый меньший выброс равен 26). В нормальном распределении эти значения можно ожидать менее чем в 1 % случаев.

6.6.1. Использование коробчатых диаграмм для сравнения групп

Коробчатые диаграммы с успехом можно использовать для сравнения распределения количественной переменной по уровням категориальной переменной. Давайте снова сравним расход бензина в городском цикле для четырех-, шести- и восьмицилиндровых автомобилей, но на этот раз используем данные за 1999 и за 2008 годы. Поскольку пятицилиндровых машин немного, исключим их из анализа. Также преобразуем переменные `year` и `cyl` в категориальные (группирующие) факторы:

```
library(ggplot2)
cars <- mpg[mpg$cyl != 5, ]
cars$Cylinders <- factor(cars$cyl)
cars$Year <- factor(cars$year)
```

Следующий код:

```
ggplot(cars, aes(x=Cylinders, y=cty)) +
  geom_boxplot() +
  labs(x="Number of Cylinders",
       y="Miles Per Gallon",
       title="Car Mileage Data")
```

выведет диаграмму, изображенную на рис. 6.20. Обратите внимание, насколько хорошо видно разделение на группы по расходу топлива бензина, при этом эффективность использования топлива падает с увеличением количества цилиндров. В группе автомобилей с четырьмя цилиндрами также есть четыре аутсайдера (это автомобили с необычно низким расходом, способные преодолеть большее расстояние на одном галлоне топлива).

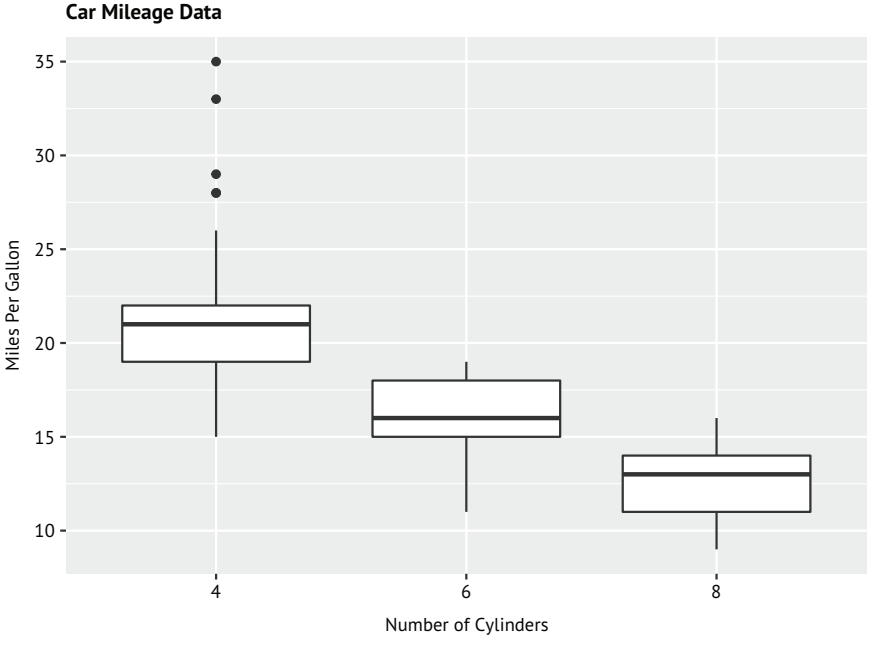


Рис. 6.20. Коробчатая диаграмма, иллюстрирующая расход топлива автомобилями с разным числом цилиндров

Коробчатые диаграммы очень многообразны. Если добавить параметр `notch=TRUE`, то получатся коробки *с насечками*. Если «насечки» двух коробок не перекрываются, то высока вероятность, что медианы соответствующих совокупностей различаются (Chambers et al., 1983, стр. 62). Следующий код создает коробки с насечками для нашего примера с расходом топлива:

```
ggplot(cars, aes(x=Cylinders, y=cty)) +
  geom_boxplot(notch=TRUE,
               fill="steelblue",
               varwidth=TRUE) +
  labs(x="Number of Cylinders",
       y="Miles Per Gallon",
       title="Car Mileage Data")
```

Параметр `fill` определяет цвет заливки коробок. В стандартной коробчатой диаграмме ширина коробок не имеет никакого опре-

деленного смысла, но если добавить параметр `varwidth=TRUE`, то ширина коробок будет пропорциональна квадратному корню из количества наблюдений в каждой группе.

На рис. 6.21 можно видеть, что медианные значения расхода топлива у четырех-, шести- и восьмицилиндровых машин различаются. Расход топлива заметно увеличивается (уменьшается расстояние, которое преодолевает автомобиль на одном галлоне топлива) с увеличением числа цилиндров. Кроме того, автомобилей с восемью цилиндрами меньше, чем с четырьмя или шестью цилиндрами (хотя разница невелика).

Наконец, можно создать коробчатые диаграммы размахов для нескольких группирующих факторов. Следующий код строит диаграммы расхода топлива в зависимости от количества цилиндров по годам (рис. 6.21). Для настройки цветов заливки здесь добавлен вызов функции `scale_fill_manual()`:

```
ggplot(cars, aes(x=Cylinders, y=cty, fill=Year)) +
  geom_boxplot() +
  labs(x="Number of Cylinders",
       y="Miles Per Gallon",
       title="City Mileage by # Cylinders and Year") +
  scale_fill_manual(values=c("gold", "green"))
```

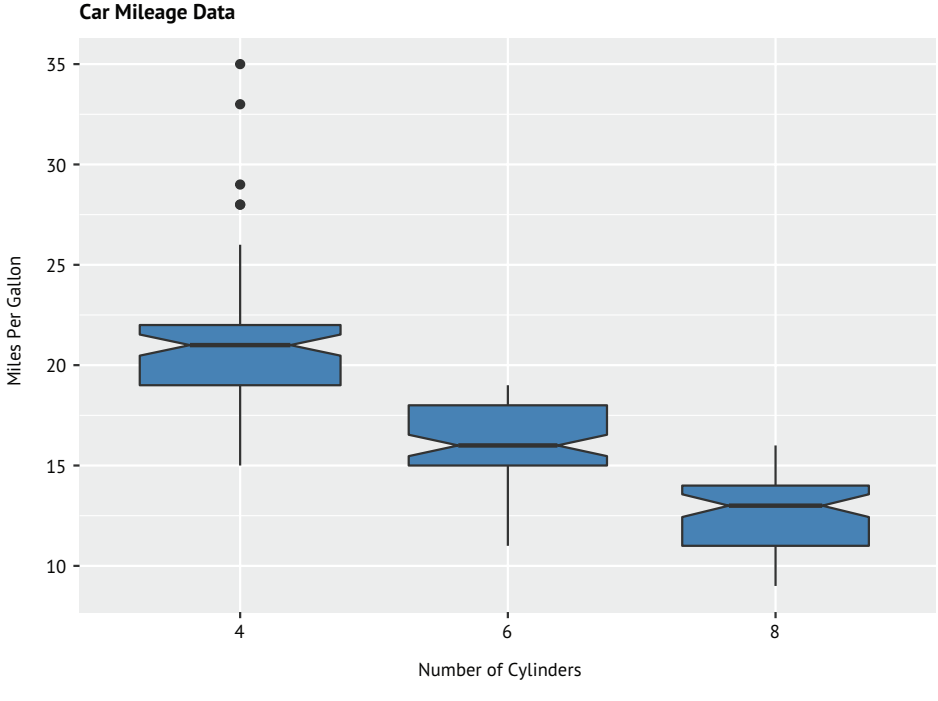


Рис. 6.21. Коробки с насечками, отражающие расход топлива автомобилями с разным числом цилиндров

Как показано на рис. 6.22, средний расход увеличивается (уменьшается пробег на одном галлоне топлива) с увеличением количества цилиндров. Кроме того, в каждой группе расход уменьшился в период с 1999 по 2008 год.

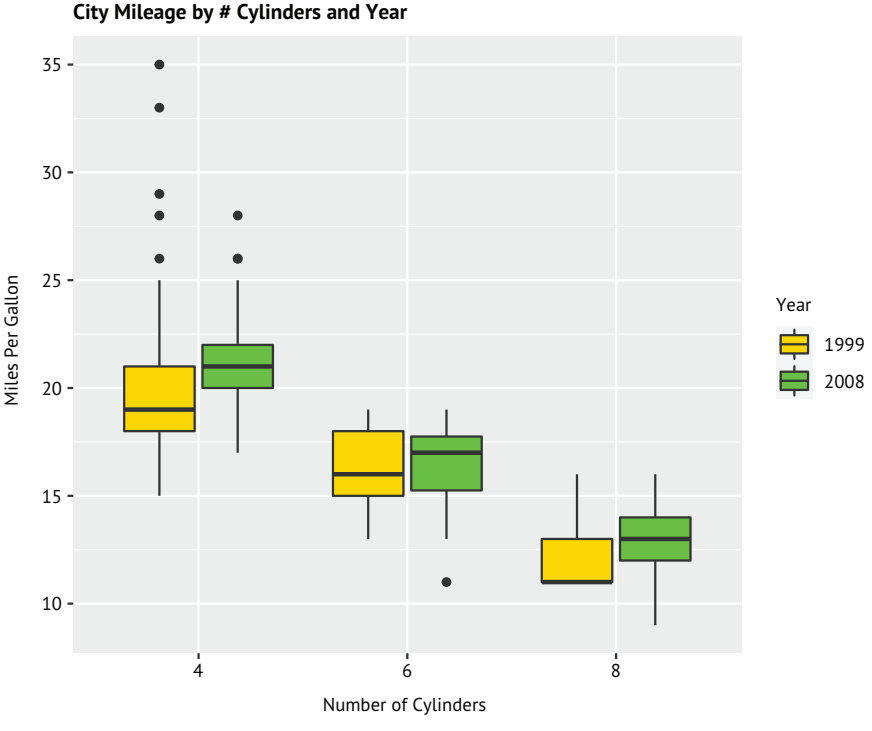


Рис. 6.22. Коробчатая диаграмма, иллюстрирующая расход топлива автомобилями с разным числом цилиндров по годам

6.6.2. *Скрипичные диаграммы*

Прежде чем закончить обсуждение коробчатых диаграмм, рассмотрим их разновидность, названную *скрипичной диаграммой* (violin plot). Это сочетание коробчатой диаграммы и диаграммы ядерной оценки функции плотности. Такую диаграмму можно создать при помощи функции `geom_violin()`. В листинге 6.10 приводится код, добавляющий скрипичную диаграмму к коробчатой, как показано на рис. 6.23.

Листинг 6.10. Скрипичные диаграммы

```
library(ggplot2)
cars <- mpg[mpg$cyl != 5, ]
cars$Cylinders <- factor(cars$cyl)

ggplot(cars, aes(x=Cylinders, y=cty)) +
  geom_boxplot(width=0.2,
               fill="green") +
```

```
geom_violin(fill="gold",
            alpha=0.3) +
labs(x="Number of Cylinders",
     y="City Miles Per Gallon",
     title="Violin Plots of Miles Per Gallon")
```

Ширина коробчатых диаграмм установлена равной `0.2`, чтобы уместить их внутрь скрипичных диаграмм. Скрипичные диаграммы на рис. 6.23 отображаются полупрозрачными (`alpha=0.3`), что позволяет видеть коробчатые диаграммы, находящиеся под ними.

Рис. 6.23. Скрипичные диаграммы, отражающие расход топлива у автомобилей с разным числом цилиндров

По своей сути скрипичные диаграммы – это симметричные графики ядерной оценки функции плотности, наложенные на коробчатые диаграммы. Линии в середине соответствуют медианам, черная рамка отражает межквартильный размах, а тонкие черные линии – усы. Точки соответствуют выбросам. Внешний контур фигуры – это графики ядерной оценки функции плотности. Здесь видно, что распределение расхода бензина восьмицилиндровыми автомобилями имеет бимодальный характер, который невозможно подметить, используя одни только коробчатые диаграммы. Скрипичные диаграммы еще не вошли в моду. Опять же, это может быть обусловлено отсутствием доступного программного обеспечения. Время покажет.

И наконец, закончим эту главу знакомством с точечными диаграммами. На них, в отличие от всех других диаграмм, с которыми мы познакомились к настоящему моменту, отображается каждое отдельное значение переменной.

6.7. Точечные диаграммы

Точечные диаграммы позволяют изобразить множество значений на простой горизонтальной шкале. Эта диаграмма создается при помощи функции `dotchart()`, имеющей следующий синтаксис:

```
ggplot(data, aes(x=contvar, y=catvar)) + geom_point()
```

где `data` – это таблица данных, `contvar` – непрерывная переменная и `catvar` – категориальная переменная. Рассмотрим пример анализа расхода топлива в загородном цикле для автомобилей 2008 года в наборе данных `mpg`. Расход топлива усреднен по моделям автомобилей:

```
library(ggplot2)
library(dplyr)
plotdata <- mpg %>%
  filter(year == "2008") %>%
  group_by(model) %>%
  summarize(meanHwy=mean(hwy))

> plotdata

# Таблица данных: 38×2
   model               meanHwy
   <chr>                 <dbl>
 1 4runner 4wd           18.5
 2 a4                    29.3
 3 a4 quattro            26.2
 4 a6 quattro            24
 5 altima                29
 6 c1500 suburban 2wd    18
 7 camry                 30
 8 camry solara          29.7
 9 caravan 2wd           22.2
10 civic                 33.8
# ... и еще 28 строк


ggplot(plotdata, aes(x=meanHwy, y=model)) +
  geom_point() +
  labs(x="Miles Per Gallon",
       y="",
       title="Gas Mileage for Car Models")
```

Получившаяся диаграмма показана на рис. 6.24.

Диаграмма позволяет увидеть расход топлива для каждой марки автомобилей на одной и той же горизонтальной оси. Обычно точечные диаграммы наиболее эффективны, когда значения отсортированы, а каждой группе соответствуют свой символ и цвет. Следующий код сортирует автомобили в порядке от наибольшего к наименьшему расходу топлива (от наименьшего к наибольшему пробегу на одном галлоне).

```
ggplot(plotdata, aes(x=meanHwy, y=reorder(model, meanHwy))) +
  geom_point() +
  labs(x="Miles Per Gallon",
       y="",
       title="Gas Mileage for Car Models")
```

Результат показан на рис. 6.25. Чтобы отсортировать наблюдения в обратном порядке, используйте функцию `reorder(model, -meanHwy)`.

Рассматривая точечную диаграмму, полученную в этом примере, можно сделать важные выводы, благодаря тому что каждая точка подписана, значение каждой точки по своей сути значимо, а точки расположены так, что их можно сравнивать. Но с увеличением количества точек данных полезность точечной диаграммы снижается.

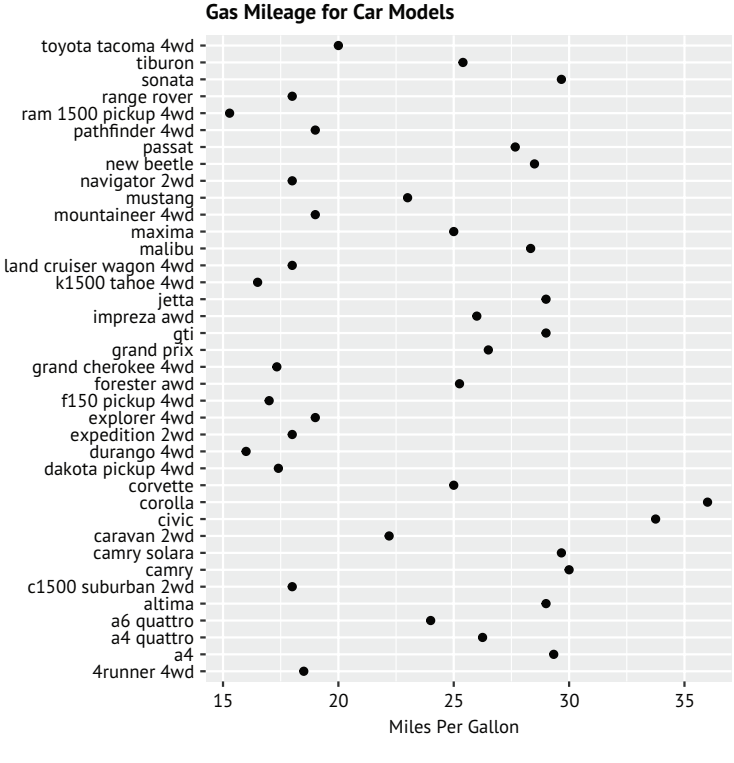


Рис. 6.24. Точечная диаграмма расхода топлива автомобилями разных марок

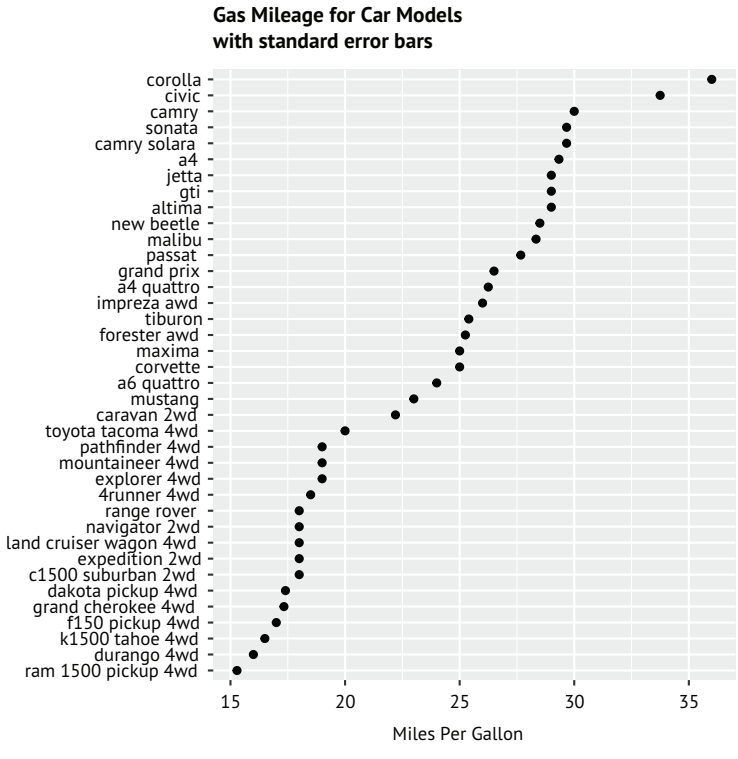


Рис. 6.25. Точечная диаграмма расхода топлива автомобилями разных марок с сортировкой

Итоги

- Столбиковые диаграммы (и, в меньшей степени, круговые диаграммы и «плоские деревья») можно использовать для получения информации о распределении категориальной переменной.
- Столбиковые диаграммы составные, с группировкой и спинограммы могут помочь понять, как различаются группы по категориальной переменной.
- Гистограммы, коробчатые, скрипичные и точечные диаграммы могут помочь визуализировать распределение непрерывных переменных.
- Перекрывающиеся графики ядерной оценки функции плотности и параллельные коробчатые диаграммы могут помочь визуализировать различия между группами в непрерывной переменной.

7

Основные методы статистической обработки данных

В этой главе:

- описательные статистики;
- таблицы частот и сопряженности;
- корреляция и ковариация;
- критерий Стьюдента;
- непараметрические методы.

В предыдущих главах вы узнали, как импортировать данные в R и как использовать разнообразные функции для упорядочения и преобразования данных в нужный формат. Затем мы рассмотрели базовые методы визуализации данных.

Обычно следующий шаг после упорядочения данных и их визуального исследования – описание распределения значений каждой переменной при помощи числовых показателей. Затем, как правило, исследуют взаимосвязи между парами переменных. Цель этих процедур – ответить на следующие вопросы:

- Каков расход топлива у современных автомобилей? А именно как распределены значения расхода топлива (среднее, стандартное отклонение, медиана, размах и т. д.) в имеющихся данных по разным маркам машин?
- Отличается ли действие нового лекарства (нет улучшения, некоторое улучшение, заметное улучшение) по сравнению с плацебо? Зависит ли результат испытаний от пола пациентов?
- Как взаимосвязаны доход и средняя продолжительность жизни? Можно ли утверждать, что коэффициент корреляции значимо отличается от нуля?
- Правда ли, что вероятность сесть в тюрьму за преступление неодинакова в разных штатах США? Значимы ли различия между штатами?

В этой главе мы рассмотрим функции R, позволяющие вычислять описательные и предсказательные статистики. Для начала познакомимся с методами оценки центральной тенденции и разброса значений количественных переменных. Затем узнаем, как создавать таблицы частот и сопряженности (и как проверить соответствие критерию хи-квадрат) для категориальных переменных. Потом исследуем разные формы коэффициентов корреляции, применимые к непрерывным и порядковым переменным. Наконец, обратимся к исследованию различий между группами при помощи параметрических (критерий Стьюдента) и непараметрических (критерии Манна–Уитни и Краскела–Уоллиса) методов. Основное наше внимание будет сосредоточено на числовых показателях, тем не менее постоянно будут упоминаться графические методы, которые можно использовать для визуализации этих показателей.

С обсуждаемыми в этой главе статистическими методами люди обычно знакомятся на первых курсах вузов. Если эти методы незнакомы вам, я рекомендую два замечательных пособия: MacCall (2000) и Kirk (2008)[1]. По каждой теме существует также множество источников в интернете (таких как Википедия).

7.1. Описательные статистики

В этом разделе мы обсудим числовые характеристики центральной тенденции, разброса и типа распределения значений непрерывных переменных на примере нескольких переменных из набора данных `mtcars` с характеристиками разных марок автомобилей, с которыми вы познакомились еще в первой главе. Мы сосредоточимся на расходе топлива (в милях на галлон – miles per gallon, `mpg`), мощности (лошадиных сил – horsepower, `hp`) и весе (weight, `wt`):

[1] На русском языке основные статистические методы кратко описаны в пособии П. А. Волковой и А. Б. Шипунова «Статистическая обработка данных в учебно-исследовательских работах» (Форум, 2012). – *Прим. перев.*

```
> myvars <- c("mpg", "hp", "wt")
> head(mtcars[myvars])
                   mpg  hp   wt
Mazda RX4         21.0 110 2.62
Mazda RX4 Wag     21.0 110 2.88
Datsun 710        22.8  93 2.32
Hornet 4 Drive    21.4 110 3.21
Hornet Sportabout 18.7 175 3.44
Valiant           18.1 105 3.46
```

Для начала рассмотрим описательные статистики для всех 32 марок автомобилей в целом. Затем вычислим описательные статистики отдельно для каждого типа трансмиссии (`am`) и вида двигателя по расположению цилиндров (`vs`). Переменная, описывающая тип трансмиссии, имеет два возможных значения: `0` (автоматическая коробка) и `1` (механическая коробка). Переменная, описывающая расположение цилиндров, тоже имеет два значения: `0` (V-образное) и `1` (рядное).

7.1.1. Калейдоскоп методов

Возможности R в отношении вычисления описательных статистик потрясают воображение. Начнем с функций, которые включены в базовую версию. Затем познакомимся с расширенными возможностями, которые становятся доступными после установки дополнительных пакетов.

В базовой версии имеется функция `summary()`, вычисляющая описательные статистики. Пример ее использования приводится в листинге 7.1.

Листинг 7.1. Вычисление описательных статистик при помощи функции summary()

```
> myvars <- c("mpg", "hp", "wt")
> summary(mtcars[myvars])
      mpg              hp              wt
 Min.   :10.4   Min.   : 52.0   Min.   :1.51
 1st Qu.:15.4   1st Qu.: 96.5   1st Qu.:2.58
 Median :19.2   Median :123.0   Median :3.33
 Mean   :20.1   Mean   :146.7   Mean   :3.22
 3rd Qu.:22.8   3rd Qu.:180.0   3rd Qu.:3.61
 Max.   :33.9   Max.   :335.0   Max.   :5.42
```

Функция `summary()` вычисляет минимум, максимум, квартили и среднее для числовых переменных и частоты значений для факторов и логических векторов. Можно также использовать функции `apply()` или `sapply()`, описанные в главе 5, и вычислять любые описательные статистики с их помощью. Функция `sapply()` имеет следующий синтаксис:

```
sapply(x, FUN, options)
```

где x – это таблица данных, а *FUN* – произвольная функция. Если заданы дополнительные параметры *options*, то они передаются функции *FUN*. Обычно в роли такой функции используются mean, sd, var, min, max, median, length, range и quantile. Функция fivenum() вычисляет пять описательных статистик Тьюки (Tukey) (минимум, нижний квартиль, медиана, верхний квартиль, максимум).

Удивительно, но в базовой версии R нет функций для вычисления таких характеристик, как асимметрия и эксцесс распределения значений, однако вы можете написать их самостоятельно. Пример в листинге 7.2 демонстрирует, как вычислить несколько описательных статистик, включая асимметрию и эксцесс.

Листинг 7.2. Вычисление описательных статистик при помощи функции sapply()

```
> mystats <- function(x, na.omit=FALSE){
                if (na.omit)
                    x <- x[!is.na(x)]
                m <- mean(x)
                n <- length(x)
                s <- sd(x)
                skew <- sum((x-m)^3/s^3)/n
                kurt <- sum((x-m)^4/s^4)/n - 3
                return(c(n=n, mean=m, stdev=s,
                       skew=skew, kurtosis=kurt))
              }

> myvars <- c("mpg", "hp", "wt")
> sapply(mtcars[myvars], mystats)
             mpg      hp       wt
n         32.000   32.000  32.0000
mean      20.091  146.688   3.2172
stdev      6.027   68.563   0.9785
skew       0.611    0.726   0.4231
kurtosis  -0.373   -0.136  -0.0227
```

Согласно полученным результатам, средний пробег в милях на галлон по всем маркам автомобилей составляет 20.1 мили со стандартным отклонением 6.0. Максимум кривой распределения значений смещен вправо (+0.61) и находится немного ниже максимума кривой нормального распределения (-0.37). Это будет хорошо заметно на графике. Отметьте также, что если понадобится удалить строки с пропущенными значениями, то вызов функции следует записать так: sapply(mtcars[vars], mystats, na.omit=TRUE).

7.1.2. Дополнительные возможности

Функции для вычисления описательных статистик имеются также в некоторых дополнительных пакетах, таких как Hmisc, pastecs, psych, skimr и summytools. Из-за ограниченного пространства в кни-

ге я продемонстрирую только первые три, но вы в своей работе с успехом можете использовать все пять. Не забудьте установить эти пакеты перед первым использованием (раздел 1.4), потому что они не входят в состав дистрибутива R.

Функция describe() из пакета Hmisc выводит число переменных и наблюдений, число пропущенных и неповторяющихся значений, среднее, квартили, а также пять наименьших и пять наибольших значений. Пример представлен в листинге 7.3.

Листинг 7.3. Вычисление описательных статистик при помощи функции describe() из пакета Hmisc

```
> library(Hmisc)
> myvars <- c("mpg", "hp", "wt")
> describe(mtcars[myvars])

 3  Variables      32  Observations
---------------------------------------------------------------------------
mpg
n missing  unique  Mean    .05   .10     .25   .50    .75    .90    .95
32       0    25   20.09 12.00  14.34  15.43  19.20  22.80  30.09  31.30

lowest : 10.4 13.3 14.3 14.7 15.0, highest: 26.0 27.3 30.4 32.4 33.9
---------------------------------------------------------------------------
hp
n missing  unique    Mean    .05     .10   .2     .50   .75   .90     .95
32       0     22   146.7  63.65  66.00 96.50 123.00 180.00 243.50 253.55

lowest :  52  62  65  66  91, highest: 215 230 245 264 335
---------------------------------------------------------------------------
wt
n missing  unique    Mean    .05    .10    .25    .50    .75    .90   .95
32       0      29   3.217  1.736  1.956  2.581  3.325  3.610  4.048 5.293

lowest : 1.513 1.615 1.835 1.935 2.140, highest: 3.845 4.070 5.250 5.345 5.424
---------------------------------------------------------------------------
```

В пакете pastecs есть функция stat.desc(), вычисляющая множество описательных статистик. Она имеет следующий синтаксис:

```
stat.desc(x, basic=TRUE, desc=TRUE, norm=FALSE, p=0.95)
```

где x – это таблица данных или временной ряд. Если basic=TRUE (по умолчанию), то вычисляется число действительных, пустых (null) и пропущенных значений, минимум, максимум, размах и сумма. Если desc=TRUE (тоже по умолчанию), то вычисляются медиана, среднее арифметическое, стандартная ошибка среднего, 95%-ный доверительный интервал для среднего, дисперсия, стандартное отклонение и коэффициент вариации. Наконец, если norm=TRUE (не по умолчанию), вычисляются статистики нормального распределения, включая асимметрию и эксцесс (и их достоверность), и кри-

терий Шапиро–Уилка (Shapiro-Wilk) соответствия нормальному распределению. Параметр p используется при вычислении доверительного интервала для среднего арифметического (по умолчанию он получает значение 0.95). Пример приводится в листинге 7.4.

Листинг 7.4. Вычисление описательных статистик при помощи функции stat.desc() из пакета pastecs

```
> library(pastecs)
> myvars <- c("mpg", "hp", "wt")
> stat.desc(mtcars[myvars])
                 mpg       hp      wt
nbr.val        32.00   32.000  32.000
nbr.null        0.00    0.000   0.000
nbr.na          0.00    0.000   0.000
min            10.40   52.000   1.513
max            33.90  335.000   5.424
range          23.50  283.000   3.911
sum           642.90 4694.000 102.952
median         19.20  123.000   3.325
mean           20.09  146.688   3.217
SE.mean         1.07   12.120   0.173
CI.mean.0.95    2.17   24.720   0.353
var            36.32 4700.867   0.957
std.dev         6.03   68.563   0.978
coef.var        0.30    0.467   0.304
```

Как будто этого недостаточно, в пакете psych тоже есть функция с именем describe(), которая выводит на экран число действительных значений, среднее арифметическое, стандартное отклонение, усеченное среднее, минимум, максимум, размах, асимметрию, эксцесс и стандартную ошибку среднего. Пример показан в листинге 7.5.

Листинг 7.5. Вычисление описательных статистик при помощи функции describe() из пакета psych

```
> library(psych)
Attaching package: 'psych'
        The following object(s) are masked from package:Hmisc :
         describe
> myvars <- c("mpg", "hp", "wt")
> describe(mtcars[myvars])
    var  n   mean    sd median trimmed   mad   min    max
mpg   1 32  20.09  6.03  19.20   19.70  5.41 10.40  33.90
hp    2 32 146.69 68.56 123.00  141.19 77.10 52.00 335.00
wt    3 32   3.22  0.98   3.33    3.15  0.77  1.51   5.42
     range skew kurtosis    se
mpg  23.50 0.61    -0.37  1.07
hp  283.00 0.73    -0.14 12.12
wt    3.91 0.42    -0.02  0.17
```

Я же говорил вам, что возможности потрясают воображение!

ПРИМЕЧАНИЕ. Как показывают предыдущие примеры, функция с именем `describe()` определена и в пакете `psych`, и в пакете `Hmisc`. Как R узнает, какую из них выбрать? Очень просто: преимущество имеет пакет, загруженный последним, как это видно в листинге 7.5. В данном случае пакет `psych` загружается после `Hmisc`, и на экран выводится сообщение о том, что функция `describe()` из пакета `Hmisc` «замаскирована» функцией с таким же именем из пакета `psych`. Встретив вызов функции, R начинает поиск с пакета `psych`, обнаруживает ее там и вызывает. Чтобы воспользоваться функцией из пакета `Hmisc`, ее можно вызвать так: `Hmisc::describe(mt)`. Сама функция никуда не делась, просто нужно подсказать R, где ее следует искать.

Теперь, познакомившись со способами вычисления описательных статистик для набора данных в целом, посмотрим, как получить их для отдельных групп.

7.1.3. Вычисление описательных статистик для групп данных

При сравнении групп объектов или наблюдений обычно обращают внимание на значения описательных статистик, характеризующих эти группы, а не на всю выборку в целом. Описательные статистики для групп можно получить с помощью функции `by()`, имеющей следующий синтаксис:

```
by(data, INDICES, FUN)
```

где `data` – это таблица данных или матрица, *`INDICES`* – фактор или список факторов, определяющих группы, и *`FUN`* – произвольная функция, оперирующая всеми столбцами в таблице данных. Пример использования этой функции приводится в листинге 7.6.

Листинг 7.6. Получение описательных статистик для групп при помощи функции by()

```
> dstats <- function(x)sapply(x, mystats)
> myvars <- c("mpg", "hp", "wt")
> by(mtcars[myvars], mtcars$am, dstats)

mtcars$am: 0
             mpg        hp        wt
n         19.000    19.0000   19.000
mean      17.147   160.2632    3.769
stdev      3.834    53.9082    0.777
skew       0.014    -0.0142    0.976
kurtosis  -0.803    -1.2097    0.142
---------------------------------------
mtcars$am: 1
             mpg        hp        wt
n         13.0000    13.000   13.000
```

```
mean       24.3923   126.846    2.411
stdev       6.1665    84.062    0.617
skew        0.0526     1.360    0.210
kurtosis   -1.4554     0.563   -1.174
```

В этом примере `dstats()` применяет функцию `mystats()` из листинга 7.2 к каждому столбцу в таблице данных. Поместив ее в вызов функции `by()`, мы получаем описательные статистики для автомобилей с разными типами трансмиссии (`am`).

Следующий пример (листинг 7.7) демонстрирует получение описательных статистик для двух переменных (`am` и `vs` – тип трансмиссии и расположение цилиндров) и выводит результаты для каждой группы под заданными нами подписями (`mpg`, `hp` и `wt` – пробег в милях на галлон, мощность в лошадиных силах и вес соответственно). Кроме того, перед вычислением статистик он удаляет пропущенные значения.

Листинг 7.7. Получение описательных статистик для групп, определяемых по нескольким переменным

```
> dstats <- function(x)sapply(x, mystats, na.omit=TRUE)
> myvars <- c("mpg", "hp", "wt")
> by(mtcars[myvars],
     list(Transmission=mtcars$am,
          Engine=mtcars$vs),
     FUN=dstats)

Transmission: 0
Engine: 0
                mpg          hp         wt
n        12.0000000  12.0000000 12.0000000
mean     15.0500000 194.1666667  4.1040833
stdev     2.7743959  33.3598379  0.7683069
skew     -0.2843325   0.2785849  0.8542070
kurtosis -0.9635443  -1.4385375 -1.1433587
------------------------------------------------------------
Transmission: 1
Engine: 0
                mpg          hp          wt
n         5.0000000   6.0000000  6.00000000
mean     19.5000000 180.8333333  2.85750000
stdev     4.4294469  98.8158219  0.48672117
skew      0.3135121   0.4842372  0.01270294
kurtosis -1.7595065  -1.7270981 -1.40961807
------------------------------------------------------------
Transmission: 0
Engine: 1
                mpg          hp         wt
n         7.0000000   7.0000000  7.0000000
mean     20.7428571 102.1428571  3.1942857
stdev     2.4710707  20.9318622  0.3477598
```

```
skew      0.1014749  -0.7248459 -1.1532766
kurtosis -1.7480372  -0.7805708 -0.1170979
------------------------------------------------------------
Transmission: 1
Engine: 1
                mpg         hp         wt
n         7.0000000  7.0000000  7.0000000
mean     28.3714286 80.5714286  2.0282857
stdev     4.7577005 24.1444068  0.4400840
skew     -0.3474537  0.2609545  0.4009511
kurtosis -1.7290639 -1.9077611 -1.3677833
```

В предыдущих примерах использовалась функция `mystats()`, но точно так же можно было бы использовать функцию `describtion()` из пакетов `Hmisc` и `psych` или `stat.desc()` из пакета `pastecs`. Фактически функция `by()` обеспечивает лишь общий механизм выполнения *любого* анализа по подгруппам.

7.1.4. Получение описательных статистик в интерактивном режиме с помощью dplyr

До сих пор мы рассматривали методы, генерирующие полный набор описательных статистик для заданной таблицы данных. Однако в ходе интерактивных исследований часто стоит цель – получить ответы на конкретные вопросы. В таких случаях желательно получить ограниченное количество статистик по конкретным группам наблюдений.

Пакет `dplyr`, с которым мы познакомились в разделе 3.11, содержит инструменты для быстрого и гибкого решения этой задачи. Функции `summarize()` и `summarize_all()` можно использовать для вычисления любых статистик, а функцию `group_by()` – для описания групп, по которым следует вычислять статистики.

В качестве примера зададим и ответим на ряд вопросов, используя таблицу данных `Salaries` из пакета `carData`. Этот набор данных содержит информацию о размере заработной платы за девять месяцев в долларах США (`salary`) в 2008–2009 годах для 397 преподавателей университета в США. Данные были собраны в рамках мониторинга разницы в заработной плате между преподавателями мужского и женского полов.

Прежде чем продолжить, установите пакеты `carData` и `dplyr` (`install.packages(c("carData", "dplyr"))`) и затем загрузите их:

```
library(dplyr)

library(carData)
```

Теперь можно приступать к анализу данных.

Какова средняя зарплата и диапазон зарплат для 397 профессоров?

```
> Salaries %>%
    summarize(med = median(salary),
              min = min(salary),
              max = max(salary))
     med   min    max
1 107300 57800 231545
```

Набор данных `Salaries` передается функции `summ()`, которая вычисляет медиану, минимальное и максимальное значения зарплаты и возвращает результат в виде однострочной таблицы данных. Средняя зарплата за девять месяцев составляет 107 300 долларов, и по крайней мере один человек заработал более 230 000 долларов. Очевидно, что я имею полное право просить о повышении.

Каково количество преподавателей, средняя заработная плата и размах в зависимости от пола и должности?

```
> Salaries %>%
    group_by(rank, sex) %>%
    summarize(n = length(salary),
              med = median(salary),
              min = min(salary),
              max = max(salary))

  rank      sex        n     med   min    max
  <fct>     <fct>  <int>   <dbl> <int>  <int>
1 AsstProf  Female    11  77000  63100  97032
2 AsstProf  Male      56  80182  63900  95079
3 AssocProf Female    10  90556. 62884 109650
4 AssocProf Male      54  95626. 70000 126431
5 Prof      Female    18 120258. 90450 161101
6 Prof      Male     248 123996  57800 231545
```

Когда функции `by_group()` передаются категориальные переменные, функция `summary()` генерирует статистики для каждой комбинации их значений. На всех факультетах у женщин медианная заработная плата ниже, чем у мужчин. Кроме того, в этом университете очень много профессоров-мужчин.

Каков средний стаж работы и как давно получена докторская степень для преподавателей в разбивке по полу и должности?

```
> Salaries %>%
    group_by(rank, sex) %>%
    select(yrs.service, yrs.since.phd) %>%
    summarize_all(mean)

  rank      sex    yrs.service yrs.since.phd
  <fct>     <fct>        <dbl>         <dbl>
1 AsstProf  Female        2.55          5.64
2 AsstProf  Male          2.34          5
3 AssocProf Female       11.5          15.5
4 AssocProf Male         12.0          15.4
```

```
5 Prof      Female      17.1          23.7
6 Prof      Male        23.2          28.6
```

Функция `summarize_all()` вычисляет сводные статистики для каждой негрупповой переменной (`yrs.service` и `yrs.since.phd`). Чтобы получить более одной статистики для каждой переменной, их нужно представить в виде списка. Например, `summarize_all(list(mean=mean, std=sd))` вычислит среднее значение и стандартное отклонение для каждой переменной. Мужчины и женщины имеют сопоставимый стаж работы на должности ассистента и доцента. Однако женщины-профессора имеют меньший стаж, чем их коллеги-мужчины.

Одно из преимуществ пакета `dplyr` – результаты возвращаются в виде усовершенствованных таблиц данных (tibbles). Это позволяет применять дополнительные виды анализа к сводным результатам, отображать их в графическом виде и выводить на печать. Он также предоставляет простой механизм агрегирования данных.

Как правило, у аналитиков есть свои предпочтения в отношении описательных статистик и формата их отображения. Вероятно, поэтому существует множество вариантов. Выберите тот, который лучше подходит для вас, или создайте свой!

7.1.5. Визуализация результатов

Представление характеристик распределения данных в числовом виде полезно, но не заменяет графического изображения. Для отображения количественных данных можно использовать гистограммы (раздел 6.4), диаграммы плотности рассеяния (раздел 6.5), коробчатые диаграммы (раздел 6.6) и точечные диаграммы (раздел 6.7). Они помогают заметить вещи, которые легко упустить, если полагаться на небольшой набор описательных статистик.

Функции, которые мы рассматривали до сих пор, позволяют охарактеризовать количественные данные. Функции, описанные в следующем разделе, предназначены для анализа распределения категориальных переменных.

7.2. Таблицы частот и таблицы сопряженности

В этом разделе мы рассмотрим таблицы частот и таблицы сопряженности для категориальных переменных, а также познакомимся с критериями независимости, мерами связи и способами графического представления результатов. Мы будем использовать функции, имеющиеся в базовой версии R, а также функции из пакетов `vcd` и `gmodels`. В примерах, представленных ниже, буквами A, B и C будут обозначаться категориальные переменные.

В примерах в этом разделе использован набор данных `Arthritis` из пакета `vcd`. Эти данные опубликованы в Kock & Edward (1988)

и представляют результаты клинического исследования нового способа лечения ревматоидного артрита двойным слепым методом. Вот первые несколько наблюдений:

```
> library(vcd)
> head(Arthritis)
   ID     Treatment     Sex     Age     Improved
1  57     Treated       Male    27      Some
2  46     Treated       Male    29      None
3  77     Treated       Male    30      None
4  17     Treated       Male    32      Marked
5  36     Treated       Male    46      Marked
6  23     Treated       Male    58      Marked
```

Способ лечения (`Treatment`: плацебо – `Placebo` и лекарство – `Treated`), пол (`Sex`: мужской – `Male` и женский – `Female`) и степень улучшения состояния больных (`Improved`: нет – `None`, некоторое – `Some`, заметное – `Marked`) – это категориальные факторы. В следующем подразделе мы создадим таблицы частот и сопряженности для этих данных.

7.2.1. Создание таблиц частот

R предлагает несколько методов создания таблиц частот и сопряженности. Самые важные функции перечислены в табл. 7.1.

Таблица 7.1. Функции для создания и преобразования таблиц сопряженности

Функция	Описание
`table(`*`var1, var2, ..., varN`*`)`	Создает *N*-мерную таблицу сопряженности для *N* категориальных переменных (факторов)
`xtabs(`*`formula, data`*`)`	Создает *N*-мерную таблицу сопряженности на основе формулы и матрицы или таблицы данных
`prop.table(`*`table, margins`*`)`	Представляет значения таблицы в виде долей от суммы всех значений в строке таблицы, заданной параметром `margins`
`margin.table(`*`tAble, margins`*`)`	Суммирует значения таблицы по строкам или столбцам (определяется параметром `margins`)
`addmargins(`*`table, margins`*`)`	Вычисляет описательные статистики `margins` (суммы по умолчанию) для заданной таблицы
`ftable(`*`table`*`)`	Создает компактную «плоскую» таблицу сопряженности

В следующих подразделах мы используем все эти функции для исследования категориальных переменных. Для начала вычислим простые частоты, потом построим таблицы сопряженности для двух переменных и закончим созданием таблиц сопряженности для нескольких переменных. В качестве первого шага создадим таблицу при помощи функций `table()` и `xtabs()`, а затем будем исследовать ее при помощи других функций.

Таблицы для одной переменной

Частоты значений одной переменной можно рассчитать при помощи функции `table()`. Например:

```
> mytable <- with(Arthritis, table(Improved))
> mytable
Improved
  None   Some  Marked
   42     14     28
```

Эти частоты можно преобразовать в доли от общей суммы при помощи функции `prop.table()`:

```
> prop.table(mytable)
Improved
  None   Some  Marked
 0.500  0.167  0.333
```

Или в проценты, используя инструкцию `prop.table()*100`:

```
> prop.table(mytable)*100
Improved
  None   Some  Marked
  50.0   16.7   33.3
```

Судя по результатам, у половины пациентов после лечения имело место некоторое или заметное улучшение состояния (`16.7 + 33.3`).

Таблицы для двух переменных

В случае двух переменных синтаксис применения функции `table()` имеет следующий вид:

```
mytable <- table(A, B)
```

где `A` – это переменная, определяющая строки таблицы сопряженности, а `B` – столбцы. Как вариант можно воспользоваться функцией `xtabs()`, которая позволяет при создании таблицы сопряженности вводить данные в виде формулы:

```
mytable <- xtabs(~ A + B, data=mydata)
```

где *`mydata`* – это матрицы или таблица данных. В общем случае переменные, для которых строится таблица сопряженности, находятся в правой части формулы (то есть справа от знака `~`) и разделяются знаками `+`. Если переменная находится в левой части формулы, предполагается, что это вектор с частотами значений (удобно, если вы их уже вычислили).

Для набора данных `Arthritis` у нас получится такая таблица:

```
> mytable <- xtabs(~ Treatment + Improved, data=Arthritis)
> mytable
          Improved
Treatment  None  Some  Marked
```

```
  Placebo   29    7      7
  Treated   13    7     21
```

С помощью функций `margin.table()` и `prop.table()` можно рассчитать соответственно частоты и доли от сумм по столбцам или строкам. При суммировании и вычислении долей по строкам получаем:

```
> margin.table(mytable, 1)
Treatment
Placebo Treated
   43      41
> prop.table(mytable, 1)
           Improved
Treatment    None   Some   Marked
  Placebo  0.674  0.163   0.163
  Treated  0.317  0.171   0.512
```

Индекс `1` в вызове `table()` относится к первой переменной. Рассматривая полученную таблицу, можно заметить, что у 51 % пациентов, получавших настоящее лекарство, наступило заметное улучшение. Для сравнения: такой эффект наступил только у 16 % пациентов, получавших плацебо.

При суммировании и вычислении долей по столбцам получаем:

```
> margin.table(mytable, 2)
Improved
   None    Some  Marked
    42      14      28
> prop.table(mytable, 2)
            Improved
Treatment    None   Some   Marked
  Placebo  0.690  0.500   0.250
  Treated  0.310  0.500   0.750
```

В этом случае индекс `2` в вызове `table()` относится к первой переменной.

При суммировании и вычислении долей по всем ячейкам таблицы получаем:

```
> prop.table(mytable)
            Improved
Treatment    None    Some   Marked
  Placebo  0.3452  0.0833  0.0833
  Treated  0.1548  0.0833  0.2500
```

Сумма всех долей равна 1.

Для вычисления сумм по столбцам или строкам можно использовать функцию `addmargins()`:

```
> addmargins(mytable)
                Improved
Treatment    None    Some   Marked    Sum
```

```
  Placebo    29        7        7        43
  Treated    13        7       21        41
  Sum        42       14       28        84
> addmargins(prop.table(mytable))
                Improved
Treatment    None    Some    Marked     Sum
  Placebo  0.3452  0.0833  0.0833  0.5119
  Treated  0.1548  0.0833  0.2500  0.4881
  Sum      0.5000  0.1667  0.3333  1.0000
```

По умолчанию функция `addmargins()` вычисляет суммы и по столбцам, и по строкам таблицы. Также можно вычислить суммы только по строкам:

```
> addmargins(prop.table(mytable, 1), 2)
                Improved
Treatment    None    Some    Marked     Sum
  Placebo   0.674   0.163   0.163    1.000
  Treated   0.317   0.171   0.512    1.000
```

или только по столбцам:

```
> addmargins(prop.table(mytable, 2), 1)
             Improved
Treatment    None    Some    Marked
  Placebo   0.690   0.500   0.250
  Treated   0.310   0.500   0.750
  Sum       1.000   1.000   1.000
```

Судя по полученным результатам, 25 % пациентов, чье состояние заметно улучшилось после лечения, получали плацебо.

> **ПРИМЕЧАНИЕ.** Функция `table()` по умолчанию игнорирует пропущенные значения (`NA`). Чтобы добавить их в таблицу сопряженности как одно из возможных значений, используйте параметр `useNA="ifany"`.

Третий способ создания таблиц сопряженности для двух переменных – использовать функцию `CrossTable()` из пакета `gmodels`. Эта функция создает такую же таблицу, как функции `PROC FREQ` в SAS или `CROSSTABS` в SPSS. В листинге 7.8 показан пример ее использования.

Листинг 7.8. Построение таблицы сопряженности для двух переменных при помощи функции CrossTable()

```
> library(gmodels)
> CrossTable(Arthritis$Treatment, Arthritis$Improved)

   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
```

```
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  84
                    | Arthritis$Improved
Arthritis$Treatment |      None |      Some |    Marked | Row Total |
--------------------|-----------|-----------|-----------|-----------|
            Placebo |        29 |         7 |         7 |        43 |
                    |     2.616 |     0.004 |     3.752 |           |
                    |     0.674 |     0.163 |     0.163 |     0.512 |
                    |     0.690 |     0.500 |     0.250 |           |
                    |     0.345 |     0.083 |     0.083 |           |
--------------------|-----------|-----------|-----------|-----------|
            Treated |        13 |         7 |        21 |        41 |
                    |     2.744 |     0.004 |     3.935 |           |
                    |     0.317 |     0.171 |     0.512 |     0.488 |
                    |     0.310 |     0.500 |     0.750 |           |
                    |     0.155 |     0.083 |     0.250 |           |
--------------------|-----------|-----------|-----------|-----------|
       Column Total |        42 |        14 |        28 |        84 |
                    |     0.500 |     0.167 |     0.333 |           |
--------------------|-----------|-----------|-----------|-----------|
```

Функция `CrossTable()` принимает дополнительные параметры, управляющие вычислением процентов (по строкам, столбцам и ячейкам); округлением результатов до заданного числа знаков после запятой; вычислением критериев хи-квадрат, Фишера и Мак-Немара; вычислением ожидаемых значений и остатков (по Пирсону, стандартизованные, скорректированные стандартизованные); учетом пропущенных значений; добавлением подписей в виде имен строк и столбцов; форматированием результатов в стиле SAS и SPSS. Более подробную информацию можно получить, вызвав `help(CrossTable)`.

Если в наборе данных имеется больше двух категориальных переменных, то есть возможность строить многомерные таблицы сопряженности. Давайте рассмотрим их.

Многомерные таблицы

Функции `table()` и `xtabs()` можно использовать для создания многомерных таблиц сопряженности для трех и более категориальных переменных. Функции `margin.table()`, `prop.table()` и `addmargins()` тоже можно применять к многомерным таблицам. Кроме того, функция `ftable()` позволяет выводить многомерные таблицы в компактном и удобном виде. Примеры использования этих функций показаны в листинге 7.9.

Листинг 7.9. Трехмерная таблица сопряженности

```
> mytable <- xtabs(~ Treatment+Sex+Improved, data=Arthritis)   (1)
> mytable
, , Improved = None

           Sex
Treatment  Female  Male
  Placebo      19    10
  Treated       6     7

, , Improved = Some

           Sex
Treatment  Female  Male
  Placebo       7     0
  Treated       5     2

, , Improved = Marked

           Sex
Treatment  Female  Male
  Placebo       6     1
  Treated      16     5

> ftable(mytable)
                   Sex Female Male
Treatment Improved
Placebo   None             19   10
          Some              7    0
          Marked            6    1
Treated   None              6    7
          Some              5    2
          Marked           16    5

> margin.table(mytable, 1)                                    (2)

Treatment
Placebo Treated
     43      41
> margin.table(mytable, 2)
Sex
Female   Male
    59     25
> margin.table(mytable, 3)
Improved
  None   Some Marked
    42     14     28
> margin.table(mytable, c(1, 3))                              (3)
          Improved
Treatment None Some Marked
  Placebo   29    7      7
```

```
  Treated   13    7     21
 > ftable(prop.table(mytable, c(1, 2)))                    4
                 Improved  None  Some Marked
Treatment Sex
Placebo   Female          0.594 0.219  0.188
          Male            0.909 0.000  0.091
Treated   Female          0.222 0.185  0.593
          Male            0.500 0.143  0.357

> ftable(addmargins(prop.table(mytable, c(1, 2)), 3))
                 Improved  None  Some Marked   Sum
Treatment Sex
Placebo   Female          0.594 0.219  0.188 1.000
          Male            0.909 0.000  0.091 1.000
Treated   Female          0.222 0.185  0.593 1.000
          Male            0.500 0.143  0.357 1.000
```

1 **Доли от общей суммы по всем ячейкам**

2 **Доли от суммы по строкам и столбцам**

3 **Доли от сумм по столбцам** `Treatment` **и** `Improved`

4 **Доли от сумм по столбцам** `Treatment` **и** `Sex`

Инструкция 1 вычисляет частоты для сочетаний всех трех факторов. Эта инструкция также демонстрирует, как можно использовать функцию `ftable()` для представления результатов в более удобном и компактном виде.

Инструкция 2 вычисляет частоты для отдельных факторов `Treatment`, `Sex` и `Improved`. Поскольку исходная таблица была создана при помощи формулы `~ Treatment+Sex+Improved`, индекс `1` соответствует переменной `Treatment`, индекс `2` – переменной `Sex`, а индекс `3` – переменной `Improved`.

Инструкция 3 вычисляет частоты по всем сочетаниям значений `Treatment` и `Improved`, объединяя данные для мужчин и женщин. Инструкция 4 вычисляет долю пациентов с разной степенью улучшения для каждого сочетания типа лечения и пола (`Treatment` и `Sex`). Здесь можно заметить, что заметное улучшение наступило у 36 % мужчин и 59 % женщин, получавших настоящее лекарство. В общем случае частоты вычисляются так, чтобы их сумма для всех значений фактора (в данном случае `Improved`), не указанного в вызове `prop.table()`, была равна единице. Это видно в последнем примере, где частоты суммируются по строкам.

Чтобы представить результаты в процентах, а не в долях от единицы, можно умножить все значения на 100. Например, следующая инструкция:

```
ftable(addmargins(prop.table(mytable, c(1, 2)), 3)) * 100
```

выведет такую таблицу:

```
                    Sex Female  Male   Sum
Treatment Improved
Placebo   None            65.5  34.5 100.0
          Some           100.0   0.0 100.0
          Marked          85.7  14.3 100.0
Treated   None            46.2  53.8 100.0
          Some            71.4  28.6 100.0
          Marked          76.2  23.8 100.0
```

Таблицы сопряженности содержат информацию о частотах всех сочетаний значений факторов, но нередко желательно также знать, связаны ли эти факторы между собой или они независимы. Критерии независимости обсуждаются в следующем разделе.

7.2.2. Критерии независимости

Проверить независимость категориальных данных в R можно несколькими способами. В этом разделе описаны три критерия: хи-квадрат, Фишера и Кохрана–Мантеля–Хензеля.

Хи-квадрат

К двумерной таблице можно применить функцию `chisq.test()`, чтобы проверить значение критерия хи-квадрат для переменных, представляющих строки и столбцы. Пример применения показан в листинге 7.10.

Листинг 7.10. Критерий хи-квадрат

```
> library(vcd)
> mytable <- xtabs(~Treatment+Improved, data=Arthritis)
> chisq.test(mytable)
        Pearson's Chi-squared test
data:  mytable
 X-squared = 13.1, df = 2, p-value = 0.001463                    (1)

> mytable <- xtabs(~Improved+Sex, data=Arthritis)
> chisq.test(mytable)
        Pearson's Chi-squared test
data:  mytable
 X-squared = 4.84, df = 2, p-value = 0.0889                      (2)

Warning message:
In chisq.test(mytable) : Chi-squared approximation may be incorrect
```

(1) **Переменные `Treatment` и `Improved` не являются независимыми.**

(2) **Переменные `Sex` и `Improved` не зависят друг от друга.**

Как показывают результаты вычисления критерия хи-квадрат (1), между типом лечения (`Treatment`) и степенью улучшения (`Improved`) состояния пациентов существует связь (`p < 0.01`). А вот между полом пациентов и степенью улучшения их состояния связи нет

(p > 0.05) 2. Значение статистической ошибки первого рода (p-value) – это вероятность отсутствия зависимости между данными переменными в генеральной совокупности. Поскольку эта вероятность достаточно мала в случае 1, можно отвергнуть гипотезу о независимости результата лечения от его типа. Однако в случае 2 недостаточно мала, поэтому есть основание полагать, что способ лечения и пол пациентов не зависят друг от друга. Предупреждение (warning message) в результатах в листинге 7.10 появляется потому, что в одной из шести ячеек таблицы сопряженности (некоторое улучшение состояния у мужчин) ожидаемое значение меньше 5, что может сделать недействительным критерий хи-квадрат.

Точный критерий Фишера

Точный критерий Фишера (Fisher's exact test) можно вычислить с помощью функции fisher.test(). Этот критерий проверяет нулевую гипотезу о независимости столбцов и строк в таблице сопряженности. Эта функция имеет следующий синтаксис:

```
fisher.test(mytable)
```

где *mytable* – это двухмерная таблица. Вот пример:

```
> mytable <- xtabs(~Treatment+Improved, data=Arthritis)
> fisher.test(mytable)
        Fisher's Exact Test for Count Data
data:  mytable
p-value = 0.001393
alternative hypothesis: two.sided
```

В отличие от многих статистических программ, функцию fisher.test() в R можно применять к любой таблице сопряженности (с двумя и более столбцами и строками), а не только к таблице размерности 2×2.

Критерий Кохрана–Мантеля–Хензеля

Функция mantelhaen.test() позволяет вычислить критерий хи-квадрат Кохрана–Мантеля–Хензеля (Cochran-Mantel-Haenszel), проверяющий нулевую гипотезу о независимости двух номинальных переменных для каждого значения третьей переменной. Приведенный ниже код проверяет гипотезу о независимости переменных Treatment и Improved для каждого значения переменной Sex. При этом предполагается отсутствие трехсторонней взаимозависимости (Treatment × Improved × Sex).

```
> mytable <- xtabs(~Treatment+Improved+Sex, data=Arthritis)
> mantelhaen.test(mytable)
        Cochran-Mantel-Haenszel test
data:  mytable
Cochran-Mantel-Haenszel M^2 = 14.6, df = 2, p-value = 0.0006647
```

Полученный результат позволяет считать, что вне зависимости от пола пациентов между способом лечения и его результатом имеется выраженная связь.

7.2.3. Меры тесноты связи

Статистические критерии значимости, описанные в предыдущем разделе, оценивали достаточность оснований для отклонения нулевой гипотезы о независимости двух переменных. Если нулевую гипотезу можно отвергнуть, то возникает следующий естественный вопрос: насколько сильна обнаруженная взаимосвязь. Функция `assocstats()` из пакета `vcd` вычисляет фи-коэффициент (phi coefficient), коэффициент сопряженности и V-коэффициент Крамера (Cramer's V) для двумерной таблицы. Пример ее использования показан в листинге 7.11.

Листинг 7.11. Оценка тесноты связи для двумерной таблицы

```
> library(vcd)
> mytable <- xtabs(~Treatment+Improved, data=Arthritis)
> assocstats(mytable)
                    X^2 df  P(> X^2)
Likelihood Ratio 13.530  2 0.0011536
Pearson          13.055  2 0.0014626

Phi-Coefficient   : 0.394
Contingency Coeff.: 0.367
Cramer's V        : 0.394
```

В общем случае чем больше значения коэффициентов, тем теснее связь. В пакете `vcd` также имеется функция `kappa()`, которая вычисляет каппу Коэна (Cohen's kappa) и взвешенную каппу для матрицы соответствий (например, степень согласованности между двумя суждениями, классифицирующими набор объектов по категориям).

7.2.4. Визуализация результатов

В R реализованы способы визуализации взаимосвязей между категориальными переменными, которые выходят далеко за пределы возможностей большинства других статистических программ. Обычно для визуализации частот значений одной переменной используются столбиковые диаграммы (раздел 6.1). В пакете `vcd` имеются замечательные функции визуализации взаимосвязей между категориальными переменными в многомерных наборах данных с использованием мозаичных диаграмм и диаграмм связей (раздел 11.4). Наконец, функции анализа соответствий в пакете `ca` позволяют визуально исследовать связи между строками и столбцами таблиц сопряженности при помощи различных геометрических образов (Nenadic, Greenacre, 2007).

На этом мы приостанавливаем обсуждение таблиц сопряженности и возобновим его после знакомства с более сложными темами в главах 11 и 19. Теперь давайте познакомимся с разными типами коэффициентов корреляции.

7.3. Корреляция

Коэффициенты корреляции используются для описания тесноты связей между количественными переменными. Знак коэффициента (+ или –) свидетельствует о направлении связи (положительная или отрицательная), а величина коэффициента показывает силу связи (от 0 – нет связи до 1 – абсолютно предсказуемая взаимосвязь).

В этом разделе мы рассмотрим разные коэффициенты корреляции наряду с критериями оценки их значимости. В наших примерах мы используем набор данных `state.x77`, входящий в состав базового дистрибутива R. В нем содержатся сведения о численности, доходе, проценте неграмотного населения, средней продолжительности жизни, уровне преступности и доле людей со средним образованием по 50 штатам США в 1977 году. Там также есть данные о температурном режиме и о площади штатов, но мы опустим их, чтобы сэкономить место. Выполните вызов `help(state.x77)`, чтобы узнать больше об этих данных. В дополнение к базовой версии R нам понадобятся пакеты `psych` и `ggm`.

7.3.1. Типы корреляций

В R можно вычислять разные коэффициенты корреляции, включая коэффициенты Пирсона, Спирмена, Кенделла, частные, полихорические и многорядные. Рассмотрим их по порядку.

Коэффициенты Пирсона, Спирмена и Кенделла

Коэффициент линейной корреляции Пирсона (Pearson product moment correlation) отражает тесноту линейной связи между двумя количественными переменными. Коэффициент ранговой корреляции Спирмана (Spearman's Rank Order correlation) отражает тесноту связи между двумя ранжированными переменными. Тау Кенделла (Kendall's Tau) – еще один непараметрический показатель ранговой корреляции.

Функция `cor()` вычисляет все три коэффициента, а функция `cov()` вычисляет ковариации. У этих функций есть много дополнительных параметров, но в общем случае синтаксис вычисления корреляций выглядит так:

```
cor(x, use= , method= )
```

Доступные параметры перечислены в табл. 7.2.

Таблица 7.2. Дополнительные параметры функций cor() и cov()

Параметр	Описание
x	Матрица или таблица данных
use	Определяет необходимость обработки пропущенных данных. Может принимать следующие значения: all.obs (предполагается, что в наборе данных нет пропущенных значений и их появление вызовет сообщение об ошибке), everything (коэффициенты корреляции для строк с отсутствующими значениями будут пропускаться и обозначаться как missing), complete.obs (учитывать только строки без пропущенных значений) и pairwise.complete.obs (учитывать все полные наблюдения для каждой пары переменных в отдельности)
method	Определяет тип коэффициента корреляции. Возможные значения: pearson, spearman и kendall

По умолчанию параметры принимают следующие значения: use="everything" и method="pearson". В листинге 7.12 приводится пример использования функции cor().

Листинг 7.12. Ковариации и корреляции

```
> states<- state.x77[,1:6]
> cov(states)
           Population Income Illiteracy Life Exp  Murder  HS Grad
Population   19931684 571230    292.868 -407.842 5663.52 -3551.51
Income         571230 377573   -163.702  280.663 -521.89  3076.77
Illiteracy        293   -164      0.372   -0.482    1.58    -3.24
Life Exp         -408    281     -0.482    1.802   -3.87     6.31
Murder           5664   -522      1.582   -3.869   13.63   -14.55
HS Grad         -3552   3077     -3.235    6.313  -14.55    65.24

> cor(states)
           Population Income Illiteracy Life Exp Murder HS Grad
Population     1.0000  0.208      0.108   -0.068  0.344 -0.0985
Income         0.2082  1.000     -0.437    0.340 -0.230  0.6199
Illiteracy     0.1076 -0.437      1.000   -0.588  0.703 -0.6572
Life Exp      -0.0681  0.340     -0.588    1.000 -0.781  0.5822
Murder         0.3436 -0.230      0.703   -0.781  1.000 -0.4880
HS Grad       -0.0985  0.620     -0.657    0.582 -0.488  1.0000
> cor(states, method="spearman")
           Population Income Illiteracy Life Exp Murder HS Grad
Population      1.000  0.125      0.313   -0.104  0.346  -0.383
Income          0.125  1.000     -0.315    0.324 -0.217   0.510
Illiteracy      0.313 -0.315      1.000   -0.555  0.672  -0.655
Life Exp       -0.104  0.324     -0.555    1.000 -0.780   0.524
Murder          0.346 -0.217      0.672   -0.780  1.000  -0.437
HS Grad        -0.383  0.510     -0.655    0.524 -0.437   1.000
```

Первый вызов выводит таблицу дисперсий и ковариаций. Второй – таблицу коэффициентов корреляции Пирсона, а третий – коэффициентов ранговой корреляции Спирмена. Судя по полученным результатам, например, между средним доходом и долей лю-

дей со средним образованием существует сильная положительная корреляция, а между средней продолжительностью жизни и долей неграмотного населения – сильная отрицательная корреляция.

Обратите внимание на то, что по умолчанию получается квадратная таблица (выводятся все комбинации всех переменных со всеми). Но при желании можно вывести прямоугольную таблицу, как показано в следующем примере:

```
> x <- states[,c("Population", "Income", "Illiteracy", "HS Grad")]
> y <- states[,c("Life Exp", "Murder")]
> cor(x,y)
            Life Exp Murder
Population    -0.068  0.344
Income         0.340 -0.230
Illiteracy    -0.588  0.703
HS Grad        0.582 -0.488
```

Этот способ применения функции `cor()` может пригодиться в случаях, когда требуется определить наличие связи между двумя наборами переменных. Учтите, что эти результаты не позволяют узнать, отличаются ли достоверно коэффициенты корреляции от нуля (иными словами, можно ли на основании имеющейся выборки уверенно утверждать, что коэффициент корреляции для генеральной совокупности отличается от нуля). Для этого нужно использовать критерии, описанные в разделе 7.3.2.

Частные корреляции

Частная корреляция – это корреляция между двумя количественными переменными, зависящими, в свою очередь, от одной или более других количественных переменных. Для вычисления коэффициентов частной корреляции можно использовать функцию `pcor()` из пакета `ggm`. Этот пакет не входит в состав дистрибутива R, и его следует устанавливать отдельно. Функция `pcor()` имеет следующий синтаксис:

```
pcor(u, S)
```

где *u* – числовой вектор, где первые два числа определяют номера переменных, для которых нужно вычислить коэффициент корреляции, а остальные числа – номера «влияющих» переменных (воздействие которых должно быть отделено). *S* – это ковариационная матрица для всех этих переменных. Проиллюстрируем применение этой функции на примере:

```
> library(ggm)
> colnames(states)
[1] "Population" "Income" "Illiteracy" "Life Exp" "Murder" "HS Grad"
> pcor(c(1,5,2,3,6), cov(states))
[1] 0.346
```

В данном случае `0.346` – это коэффициент корреляции между численностью населения и уровнем преступности после исключения

влияния величины дохода и долей неграмотного населения и людей со средним образованием (переменные 2, 3 и 6 соответственно). Частные корреляции обычно используются в социологии.

Другие виды корреляций

Функция `hetcor()` из пакета `polycor` позволяет вычислить комбинированную корреляционную матрицу, содержащую коэффициенты корреляции Пирсона для числовых переменных, многорядные корреляции между числовыми и порядковыми переменными, полихорические корреляции между порядковыми переменными и тетрахорические корреляции между двумя дихотомическими переменными. Многорядные, полихорические и тетрахорические корреляции могут вычисляться для порядковых и дихотомических переменных, подчиняющихся нормальному распределению. Дополнительную информацию об этих видах корреляций можно найти в справке пакета.

7.3.2. Проверка статистической значимости корреляций

Как проверить статистическую значимость вычисленных коэффициентов корреляции? Стандартная нулевая гипотеза – отсутствие связи (то есть коэффициент корреляции для генеральной совокупности равен нулю). Для проверки значимости отдельных коэффициентов корреляции Пирсона, Спирмена и Кенделла можно использовать функцию `cor.test()`. Она имеет следующий синтаксис:

```
cor.test(x, y, alternative = , method = )
```

где *x* и *y* – это переменные, корреляция между которыми исследуется, параметр `alternative` определяет тип проверки критерия (`"two.side"`, `"less"` или `"greater"`), параметр `method` определяет тип критерия (`"pearson"`, `"kendall"` или `"spearman"`). Параметр `alternative="less"` проверяет гипотезу о том, что в генеральной совокупности коэффициент корреляции меньше нуля, а параметр `alternative="greater"` – что он больше нуля. По умолчанию используется тип проверки `alternative="two.side"` (проверяется гипотеза о том, что коэффициент корреляции в генеральной совокупности не равен нулю). Пример использования `cor.test()` показан в листинге 7.13.

Листинг 7.13. Проверка статистической значимости коэффициента корреляции

```
> cor.test(states[,3], states[,5])

        Pearson's product-moment correlation

  data:  states[, 3] and states[, 5]
t = 6.85, df = 48, p-value = 1.258e-08
alternative hypothesis: true correlation is not equal to 0
```

```
95 percent confidence interval:
 0.528 0.821
sample estimates:
  cor
0.703
```

Здесь проверяется нулевая гипотеза о том, что коэффициент корреляции Пирсона между средней продолжительностью жизни и уровнем преступности равен нулю. Если в генеральной совокупности этот коэффициент равен нулю, то его значение для случайной выборки будет превышать 0,703 реже, чем в одном случае из 10 млн (это и означает `p-value = 1.258e-08`). Учитывая, насколько мала вероятность, можно отвергнуть нулевую гипотезу и принять альтернативную – о том, что значение этого коэффициента для генеральной совокупности *не равно* нулю.

К сожалению, при помощи функции `cor.test()` одновременно можно проверить значимость только одного коэффициента корреляции. Зато в пакете `psych` есть функция `corr.test()`, которая позволяет сделать больше. С ее помощью можно вычислить коэффициенты корреляции Пирсона, Спирмена и Кенделла между несколькими переменными и оценить их значимость, как показано в листинге 7.14.

Листинг 7.14. Создание матрицы коэффициентов корреляции и проверка их значимости при помощи функции corr.test()

```
> library(psych)
> corr.test(states, use="complete")

Call:corr.test(x = states, use = "complete")
Correlation matrix
           Population Income Illiteracy Life Exp Murder HS Grad
Population       1.00   0.21       0.11    -0.07   0.34   -0.10
Income           0.21   1.00      -0.44     0.34  -0.23    0.62
Illiteracy       0.11  -0.44       1.00    -0.59   0.70   -0.66
Life Exp        -0.07   0.34      -0.59     1.00  -0.78    0.58
Murder           0.34  -0.23       0.70    -0.78   1.00   -0.49
HS Grad         -0.10   0.62      -0.66     0.58  -0.49    1.00

Sample Size
[1] 50

Probability value
           Population Income Illiteracy Life Exp Murder HS Grad
Population       0.00   0.15       0.46     0.64   0.01     0.5
Income           0.15   0.00       0.00     0.02   0.11     0.0
Illiteracy       0.46   0.00       0.00     0.00   0.00     0.0
Life Exp         0.64   0.02       0.00     0.00   0.00     0.0
Murder           0.01   0.11       0.00     0.00   0.00     0.0
HS Grad          0.50   0.00       0.00     0.00   0.00     0.0
```

Параметр use= может принимать значения "pairwise" и "complete" (для попарного или построчного удаления пропущенных значений соответственно). Параметр method= может принимать значения "pearson" (по умолчанию), "spearman" и "kendall". Из приведенного примера видно, что коэффициент корреляции между неграмотностью и ожидаемой продолжительностью жизни (-0.59) значимо отличается от нуля (p = 0.00), откуда можно сделать вывод, что с ростом неграмотности снижается продолжительность жизни. Однако коэффициент корреляции между численностью населения и долей людей со средним образованием (-0.10) не отличается значимо от нуля (p = 0.5).

Другие критерии статистической значимости

В разделе 7.4.1 мы обсуждали частные корреляции. Отсутствие зависимости между двумя переменными при исключении влияния других переменных можно проверить при помощи функции pcor.test() из пакета psych, при условии что значения всех этих переменных распределены нормально. Эта функция имеет следующий синтаксис:

```
pcor.test(r, q, n)
```

где *r* – частная корреляция, вычисленная при помощи функции pcor(), *q* – число переменных, влияние которых исключается, *n* – объем выборки.

Прежде чем закончить обсуждение этой темы, я должен упомянуть функцию r.test() из пакета psych, которая позволяет вычислить несколько критериев статистической значимости. Эту функцию можно использовать для проверки значимости:

- коэффициента корреляции;
- различий между двумя независимыми корреляциями;
- различий между двумя зависимыми корреляциями, имеющими одну общую переменную;
- различий между двумя зависимыми корреляциями разных пар переменных.

Подробности ищите в справке help(r.test).

7.3.3. Визуализация корреляций

Корреляции между парами переменных можно визуализировать при помощи диаграмм рассеяния и составленных из них матриц. Коррелограммы – это непревзойденный и действенный метод сравнения большого числа коэффициентов корреляции в легко интерпретируемой форме. Оба этих подхода мы рассмотрим в главе 11.

7.4. Критерий Стьюдента

Очень часто исследователям приходится сравнивать две группы объектов. И отвечать на такие вопросы: верно ли, что больные, получающие новое лекарство, показывают лучшую динамику выздоровления, чем пациенты, которых лечат старым способом? Верно ли, что одна технология производства характеризуется меньшим процентом брака, чем другая? Какой из методов преподавания эффективнее по отношению цена/качество? Если результат выражен в виде категориальной переменной, можно использовать методы, описанные в разделе 7.3. Здесь мы сосредоточимся на сравнении групп по значениям непрерывной переменной, распределенным нормально.

Для иллюстрации используем набор данных `UScrime`, входящий в пакет `MASS`. Эти данные содержат информацию о влиянии карательного законодательства на уровень преступности в 47 штатах США в 1960 году. Мы исследуем переменные `Prob` (вероятность угодить в тюрьму), `U1` (уровень безработицы для городских жителей мужского пола в возрасте от 14 до 24 лет) и `U2` (этот же показатель для мужчин в возрасте 35–39 лет). Категориальная переменная `So` (признак принадлежности штата к группе южных штатов) будет использована в качестве группирующей. Данные были масштабированы авторами исследования. (Приступая к работе над этим разделом, я хотел назвать его «Преступление и наказание на Далеком Юге[1]», но благоразумие взяло верх).

7.4.1. Критерий Стьюдента для независимых выборок

Верно ли, что вероятность оказаться в тюрьме после совершения преступления выше в южных штатах? Чтобы ответить на этот вопрос, нужно сравнить вероятность лишения свободы в южных штатах и в остальных. Для проверки гипотезы о равенстве двух средних значений можно использовать критерий Стьюдента для независимых выборок. В данном случае подразумевается, что эти две группы не зависят друг от друга и данные происходят из нормальных распределений. Функция `t.test()` поддерживает два синтаксиса:

```
t.test(y ~ x, data)
```

где *y* – числовая переменная, а *x* – дихотомическая, или:

```
t.test(y1, y2)
```

где *y1* и *y2* – это числовые векторы (анализируемые значения для каждой из групп). Необязательный аргумент *data* – это матрица

[1] Далекий Юг (Deep South) – традиционное название южных штатов, связанное с особым укладом жизни и традициями населения. – *Прим. перев.*

или таблица данных, в которой содержатся данные. В отличие от большинства статистических программ, в R по умолчанию не подразумевается равенство дисперсий и используется трансформация степеней свободы Уэлча (Welch degrees of freedom modification). Указать на равенство дисперсий можно, добавив параметр `var.equal=TRUE`. По умолчанию используется двухсторонняя альтернативная гипотеза (о том, что средние значения различаются, но не важно, в какую сторону). Также можно использовать параметр `alternative="less"` или `alternative="greater"`, чтобы проверить значимость различий в определенном направлении.

Следующий код сравнивает вероятность попасть в тюрьму в южных (группа 1) и остальных (группа 0) штатах при помощи двухстороннего критерия, не предполагая равенства дисперсий:

```
> library(MASS)
> t.test(Prob ~ So, data=UScrime)

        Welch Two Sample t-test

data:  Prob by So
t = -3.8954, df = 24.925, p-value = 0.0006506
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.03852569 -0.01187439
sample estimates:
mean in group 0 mean in group 1
     0.03851265      0.06371269
```

Согласно этому критерию (`p < 0.001`), можно отвергнуть гипотезу о равенстве шансов попасть в тюрьму в южных и остальных штатах.

> **ПРИМЕЧАНИЕ.** Поскольку исследуемая переменная выражена в долях единицы, перед вычислением критерия Стьюдента можно попробовать привести ее к нормальному распределению. В данном случае все разумные преобразования этой переменной (`Y/1-Y`, `log(Y/1-Y)`, `arcsin(Y)` и `arcsin(sqrt(Y))`) приведут к одному и тому же результату. Преобразования переменных подробно обсуждаются в главе 8.

7.4.2. Критерий Стьюдента для зависимых выборок

В качестве второго примера можно узнать, верно ли, что уровень безработицы у юношей (14–24 года) выше, чем у мужчин (35–39 лет). В данном случае эти две группы не независимы. Вы ведь не станете ожидать, что уровень безработицы у юношей и у мужчин в Алабаме – независимые величины? Когда данные для двух групп связаны между собой, следует вычислять критерий в предположении зависимости переменных. Это же относится и к результатам повторных из-

мерений или к измерениям одного и того же объекта, проведенным до и после какого-либо воздействия.

Подразумевается, что разность значений для двух групп имеет нормальное распределение. В данном случае синтаксис вызова функции `t.test()` имеет следующий вид:

```
t.test(y1, y2, paired=TRUE)
```

где *y1* и *y2* – это числовые векторы, представляющие две зависимые группы. Результаты вычисления критерия выглядят следующим образом:

```
> library(MASS)
> sapply(UScrime[c("U1","U2")], function(x)(c(mean=mean(x),sd=sd(x))))
       U1    U2
mean 95.5 33.98
sd   18.0  8.45

> with(UScrime, t.test(U1, U2, paired=TRUE))

        Paired t-test

data:  U1 and U2
t = 32.4066, df = 46, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 57.67003 65.30870
sample estimates:
mean of the differences
               61.48936
```

Разность средних (`61.5`) достаточно велика, чтобы отклонить гипотезу о равенстве уровня безработицы для юношей и мужчин (у юношей она выше). В самом деле, вероятность получить такое значение разности средних для выборок, в то время как они будут равны в генеральных совокупностях, меньше `0.00000000000000022` (т. е. `2.2e-16`).

7.4.3. Когда имеется больше двух групп

Что делать, если понадобится одновременно сравнить больше двух групп? Если предположить, что эти группы независимы и подчиняются закону нормального распределения, то можно использовать дисперсионный анализ (ANOVA). Это сложный подход, который можно применять для анализа разных типов экспериментов и квазиэкспериментов, поэтому он заслуживает отдельной главы. Вы можете спокойно прервать чтение данного раздела в любом месте и перейти к главе 9.

7.5. Непараметрические критерии межгрупповых различий

Если ваши данные не удовлетворяют условиям применения таких параметрических методов, как критерий Стьюдента или ANOVA, можно обратиться к непараметрическим методам. Описанные в этом разделе методы можно использовать, например, если исследуемые переменные порядковые или их распределение сильно отличается от нормального.

7.5.1. Сравнение двух групп

Если две группы независимы, можно использовать критерий Вилкоксона ранговых сумм, более известный как критерий Манна–Уитни (Mann-Whitney U test), чтобы узнать, принадлежат ли наблюдения одному и тому же распределению вероятностей (иными словами, действительно ли вероятность получить более высокие значения выше в одной выборке, чем в другой). Эта функция имеет следующий синтаксис:

```
wilcox.test(y ~ x, data)
```

где *y* – числовая переменная, а *x* – дихотомическая, или такой:

```
wilcox.test(y1, y2)
```

где *y1* и *y2* – исследуемые переменные, представляющие группы. Необязательный аргумент *data* определяет матрицу или таблицу данных, содержащую исследуемые переменные. По умолчанию вычисляется двухсторонний критерий, однако можно добавить параметр `exact`, чтобы вычислить точный критерий (exact test), и параметр `alternative="less"` или `alternative="greater"`, чтобы проверить соответствующую одностороннюю гипотезу.

Если применить критерий Манна–Уитни[1] для ответа на вопрос о вероятности попадания в тюрьму, обсуждаемый в предыдущем разделе, получится следующий результат:

```
> with(UScrime, by(Prob, So, median))


So: 0
[1] 0.0382
--------------------
So: 1
[1] 0.0556

> wilcox.test(Prob ~ So, data=UScrime)

        Wilcoxon rank sum test
```

[1] По умолчанию для независимых переменных. – *Прим. перев.*

```
data:  Prob by So
W = 81, p-value = 8.488e-05
alternative hypothesis: true location shift is not equal to 0
```

И снова есть все основания отвергнуть гипотезу о равенстве вероятностей оказаться в тюрьме в южных и в других штатах (`p < 0.01`).

Критерий Вилкоксона[1] – это непараметрическая альтернатива критерию Стьюдента для зависимых выборок. Его следует применять в случаях, когда имеет место зависимость между группами, распределение которых отличается от нормального. Синтаксис применения функции такой же, как и в предыдущем примере, нужно лишь добавить параметр `paired=TRUE`. Давайте применим этот критерий, чтобы ответить на вопрос о безработице из предыдущего раздела.

```
> sapply(UScrime[c("U1","U2")], median)
U1 U2
92 34

> with(UScrime, wilcox.test(U1, U2, paired=TRUE))

        Wilcoxon signed rank test with continuity correction

data:  U1 and U2
V = 1128, p-value = 2.464e-09
alternative hypothesis: true location shift is not equal to 0
```

И снова мы приходим к тому же выводу, что и при использовании критерия Стьюдента для зависимых переменных.

В данном случае параметрический критерий Стьюдента и его непараметрические аналоги дали одинаковые результаты. Когда условия применения критерия Стьюдента выполняются, параметрические критерии имеют бóльшую мощность (бóльшую вероятность обнаружить существующие различия) по сравнению с непараметрическими. Непараметрические критерии разумно применять, когда условия применения параметрических критериев существенно нарушаются (например, для порядковых переменных).

7.5.2. Сравнение более двух групп

Когда нужно сравнить больше двух групп, приходится использовать другие методы. Вернемся к набору данных `state.x77`, представленному в разделе 7.4. Он содержит данные о численности населения, доходе, уровне неграмотности, средней продолжительности жизни, уровне преступности и доле людей со средним образованием в разных штатах Америки. Представьте, что вам понадобилось сравнить уровень неграмотности в четырех регионах страны (северо-восток – Northeast, юг – South, север – North Central и запад – West). Это называется *однофакторный дизайн* эксперимента, и для

[1] Для зависимых выборок. – *Прим. перев.*

ответа на поставленный вопрос можно использовать как параметрические, так и непараметрические методы.

Если данные не удовлетворяют требованиям ANOVA для оценки межгрупповых различий средних значений, то можно использовать непараметрические методы. Если группы независимы, лучше всего подойдет критерий Краскела–Уоллиса (Kruskal-Wallis test). Если группы зависимы (например, это результаты повторных измерений или данные эксперимента с блочным дизайном), то следует использовать критерий Фридмана (Friedman test).

Синтаксис применения критерия Краскела–Уоллиса имеет следующий вид:

```
kruskal.test(y ~ A, data)
```

где *y* – это числовая переменная, *A* – группирующая переменная, принимающая два значения и более (в случае с двумя значениями этот критерий идентичен критерию Вилкоксона). Синтаксис применения критерия Фридмана имеет следующий вид:

```
friedman.test(y ~ A | B, data)
```

где *y* – числовая переменная, *A* – группирующая переменная, *B* – блочная переменная, идентифицирующая парные наблюдения. В обоих случаях `data` – это необязательный аргумент, матрица или таблица данных, содержащая исследуемые переменные.

Давайте используем критерий Краскела–Уоллиса для ответа на вопрос об уровне неграмотности. Сначала добавим в набор данных информацию о региональной принадлежности штатов. Эта информация имеется в наборе данных `state.region`, входящем в дистрибутив R.

```
states <- data.frame(state.region, state.x77)
```

Теперь можно вычислить критерий:

```
> kruskal.test(Illiteracy ~ state.region, data=states)

        Kruskal-Wallis rank sum test
data:  states$Illiteracy by states$state.region
Kruskal-Wallis chi-squared = 22.7, df = 3, p-value = 4.726e-05
```

Значимость критерия свидетельствует о том, что уровень неграмотности неодинаков во всех четырех регионах (`p < 0.001`).

Мы отвергли нулевую гипотезу об отсутствии различий, но из полученных результатов не ясно, *какие* регионы значимо отличаются от других. Чтобы ответить на этот вопрос, можно сравнить попарно все группы, используя критерий Вилкоксона. Более изящное решение состоит в одновременном множественном сравнении всех пар регионов с контролем значений статистической ошибки первого рода (вероятность посчитать значимыми незначимые различия). Этот подход реализован в функции `wmc()`. Она сравнивает

группы попарно, используя критерий Вилкоксона, и корректирует значения вероятности, используя функцию `p.adj()`.

Честно говоря, здесь я немного вышел за пределы *основных* методов, заявленных в названии этой главы, но поскольку рассмотрение данного подхода здесь вполне уместно, я надеюсь, что вы меня простите. Для начала загрузите текстовый файл с исходным кодом функции `wmc()`, доступный по адресу: https://rkabacoff.com/RiA/wmc.R. Листинг 7.15 демонстрирует применение этой функции для сравнения уровней неграмотности в четырех регионах США.

Листинг 7.15. Непараметрическое сравнение нескольких групп

```
> source("https://rkabacoff.com/RiA/wmc.R")                    (1)
> states <- data.frame(state.region, state.x77)
> wmc(Illiteracy ~ state.region, data=states, method="holm")

Descriptive Statistics                                         (2)

        West North Central Northeast South
n      13.00         12.00       9.0 16.00
median  0.60          0.70       1.1  1.75
mad     0.15          0.15       0.3  0.59

Multiple Comparisons (Wilcoxon Rank Sum Tests)                 (3)
Probability Adjustment = holm

        Group.1       Group.2  W       p
1          West North Central 88 8.7e-01
2          West     Northeast 46 8.7e-01
3          West         South 39 1.8e-02   *
4 North Central     Northeast 20 5.4e-02   .
5 North Central         South  2 8.1e-05 ***
6     Northeast         South 18 1.2e-02   *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(1) **Загрузка функции**

(2) **Вычисление базовых статистик**

(3) **Попарные сравнения**

Функция `source()` загружает и запускает сценарий R, определяющий функцию `wmc()` (1). Эта функция имеет следующий синтаксис: `wmc(`*`y ~ A, data, method`*`)`, где *y* – числовая переменная, *A* – группирующая переменная, *data* – таблица данных, содержащая эти переменные, а `method` – метод ограничения ошибок первого рода. Код в листинге 7.15 использует метод корректировки, разработанный Холмом (Holm, 1979), который обеспечивает строгий контроль над коэффициентом ошибок по семействам (вероятность совершения одной или нескольких ошибок первого рода в наборе сравнений). Описание других доступных методов ищите в справке `help(p.adjust)`.

Функция `wmc()` сначала определяет размеры выборки, медианы и абсолютные отклонения медиан для каждой группы (2). На западе самый низкий уровень неграмотности, а на юге – самый высокий. Затем функция генерирует шесть статистических сравнений (запад и север, запад и северо-восток, запад и юг, север и северо-восток, север и юг и северо-восток и юг) (3). Из двусторонних значений `p` можно заключить, что юг значительно отличается от трех других регионов и что остальные три региона не отличаются друг от друга на уровне `p < 0.05`.

7.6. Визуализация групповых различий

В разделах 7.4 и 7.5 мы рассмотрели статистические методы сравнения групп. Визуальный анализ межгрупповых различий – это тоже очень важный этап комплексного подхода к анализу данных. Он позволяет получить информацию о величине различий, выявить особенности распределения значений (асимметрию, бимодальность, выбросы), которые влияют на результаты сравнения, и оценить, насколько данные удовлетворяют статистическим критериям. В R реализовано множество графических методов для сравнения групп, включая коробчатые диаграммы (простые, с насечками и скрипичные), которые описаны в разделе 6.6; наложенные друг на друга диаграммы ядерной плотности, рассмотренные в разделе 6.5, и графические методы оценки соответствия данных статистическим критериям, которые обсуждаются в главе 9.

Итоги

- Описательные статистики позволяют численно описать распределение количественной переменной. В R есть множество пакетов, вычисляющих описательные статистики для таблиц данных. Выбор пакета зависит прежде всего от личных предпочтений.
- Таблицы частот и сопряженности обобщают распределение категориальных переменных.
- Для сравнения двух групп по количественным переменным можно использовать критерии Стьюдента и Манна–Уитни.
- Для оценки тесноты связи между двумя категориальными переменными можно использовать критерий хи-квадрат, а для оценки тесноты связи между двумя количественными переменными – коэффициент корреляции.
- Числовые сводки и результаты проверки статистических критериев должны сопровождаться визуализацией данных, иначе можно упустить из виду важные особенности данных.

Часть III

Методы средней сложности

Вторая часть была посвящена базовым методам обработки данных, а в этой третьей части описываются методы средней сложности. В главе 8 мы перейдем от описания взаимоотношений между двумя переменными к моделированию взаимосвязей между числовыми зависимыми переменными и наборами непрерывных и/или категориальных независимых переменных. Моделирование данных – обычно сложный многоэтапный интерактивный процесс.

В главе 8 содержится пошаговое описание методов построения линейных моделей, их оценки и интерпретации.

Глава 9 посвящена обработке результатов основных типов экспериментальных и квазиэкспериментальных исследований при помощи дисперсионного анализа и его разновидностей. В данном случае нас будет интересовать, как разные комбинации воздействующих факторов влияют на числовую зависимую переменную. В этой главе представлены функции, при помощи которых в R осуществляется дисперсионный анализ, ковариационный анализ, дисперсионный анализ для повторных измерений, а также много-

мерный и многофакторный дисперсионный анализ. Кроме того, мы обсудим методы оценки соответствия результатов анализа реальной ситуации и способы визуализации результатов.

При разработке экспериментальных и квазиэкспериментальных исследований важно определить размер выборки, необходимый для обнаружения интересующих нас закономерностей (провести анализ мощности). А иначе зачем же проводить исследование? Детальному обсуждению анализа мощности посвящена глава 10. Она начинается с обсуждения общих вопросов проверки гипотез, но основное внимание уделено использованию функций R для определения объема выборки, необходимого для обнаружения эффекта заданной силы с заданной степенью уверенности. Это поможет вам планировать исследования, обеспечивающие высокую вероятность получения полезных результатов.

Глава 11 – это логическое продолжение главы 6. В ней обсуждается создание диаграмм для визуализации взаимосвязей между двумя и более переменными. К таким диаграммам относятся разные типы двух- и трехмерных диаграмм рассеяния, матрицы диаграмм рассеяния, линейные графики и пузырьковые диаграммы. Также будут описаны полезные, но менее известные коррелограммы и мозаичные диаграммы.

Линейные модели, описанные в главах 8 и 9, подразумевают не только числовой характер зависимой переменной, но подчиненность нормальному распределению. Однако иногда нет никаких оснований предполагать нормальность распределения зависимой переменной. Поэтому в главе 12 будут описаны аналитические методы, хорошо работающие в случаях, когда данные имеют смешанное или неизвестное распределение, когда объем выборки невелик, когда часто встречаются выбросы или когда разработка подходящего критерия на основании теоретического распределения затруднена с математической точки зрения. Эти методы основаны на использовании повторных выборок и бутстреп-анализа – подходов, требующих интенсивных компьютерных вычислений и широко представленных в R. Описанные в этой главе методы позволят вам разрабатывать критерии для проверки гипотез в отношении данных, которые нельзя проанализировать с использованием традиционных параметрических методов.

К концу третьей части вы научитесь решать большинство аналитических задач, встречающихся на практике. И сможете создавать некоторые эффектные диаграммы!

8 Регрессия

В этой главе:

- создание и интерпретация линейных моделей;
- оценка адекватности допущений, сделанных при построении модели;
- выбор между альтернативными моделями.

Во многих отношениях регрессионный анализ лежит в основе статистической обработки данных. Под этим широким термином скрывается набор методов, используемых для предсказания переменной-отклика (также называемой зависимой, результирующей или условной переменной) по значениям одной или более предсказывающих переменных (также называемых независимыми, или объясняющими). В общем случае регрессионный анализ можно использовать для обнаружения независимых переменных, которые имеют отношение к зависимой, для описания типа взаимосвязи и для составления уравнения, позволяющего предсказать значения зависимой переменной по значениям независимых.

Например, тренер по фитнесу может использовать регрессионный анализ, чтобы разработать уравнение, которое позволит предсказать ожидаемое число израсходованных калорий при занятиях на беговой дорожке. Зависимая переменная – это число сожженных калорий (вычисленное по объему потребленного кислорода), а к независимым переменным могут быть отнесены длительность

тренировки (в минутах), доля времени, когда пульс держался на нужной частоте, средняя скорость (миль в час), возраст (в годах), пол и индекс массы тела.

С точки зрения теоретика, регрессионный анализ поможет найти ответ, например, на такие вопросы:

- Как связаны продолжительность упражнений и число сожженных калорий? Она линейная или нелинейная? Например, верно ли, что упражнение оказывает меньшее влияние на число потраченных калорий, если характеризующие его величины превышают порог?
- Какова роль затраченных усилий (доля времени с нужной частотой пульса, средняя скорость ходьбы)?
- Одинаковы ли обнаруженные взаимосвязи для молодых и пожилых, мужчин и женщин, толстых и худых?

С точки зрения практика, регрессионный анализ поможет найти ответ, например, на такие вопросы:

- Сколько калорий может израсходовать 30-летний мужчина с индексом массы тела, равным 28.7, если он будет идти 45 минут со скоростью 4 мили в час, удерживая пульс на нужной частоте в течение 80 % времени?
- Какое минимальное число переменных нужно использовать, чтобы точно предсказать число калорий, которое человек израсходует во время ходьбы?
- Насколько точными будут такие прогнозы?

Поскольку регрессионный анализ играет важную роль в современной статистической обработке данных, в этой главе мы рассмотрим его достаточно детально. Сначала мы узнаем, как создавать и интерпретировать регрессионные модели. Затем рассмотрим разные способы обнаружения возможных недостатков этих моделей и узнаем, что делать с ними. После этого разберем вопрос о выборе переменных. Как выбрать независимые переменные из доступных для включения в окончательную модель? Потом мы обратимся к задаче обобщения. Насколько хорошо будет работать модель в реальном мире? И наконец, затронем вопрос относительной важности переменных. Какие независимые переменные, включенные в модель, самые важные, какие находятся на втором месте по важности, а какие наименее важные?

Как видите, глава рассматривает широкий спектр проблем. Эффективный регрессионный анализ – это интерактивный и целостный процесс, состоящий из множества этапов, для которого требуется больше, чем простое умение. Я решил не разбивать обсуждаемый материал на несколько глав и представить все в одной главе, чтобы вы почувствовали все своеобразие регрессионного анализа.

В результате получилась самая длинная и сложная глава во всей книге. Внимательно прочитайте ее до самого конца, и вы овладеете всеми инструментами, необходимыми для решения различных исследовательских задач. Обещаю!

8.1. Многоликая регрессия

Термин *регрессия* может вызвать путаницу, поскольку существует множество ее разновидностей (табл. 8.1). Кроме того, в R реализован большой арсенал мощных методов подбора регрессионных моделей, и обилие имеющихся возможностей тоже может вводить в заблуждение. Например, в 2005 году Вито Риччи (Vito Ricci) составил список из 205 функций в R, которые используются для регрессионного анализа (http://mng.bz/NJhu).

Таблица 8.1. Разновидности регрессионного анализа

Вид регрессии	Где используется
Простая линейная	Предсказание значений количественной зависимой переменной по значениям одной количественной независимой переменной
Полиномиальная	Предсказание значений количественной зависимой переменной по значениям количественной независимой переменной, когда взаимосвязь моделируется как полином *n*-й степени
Множественная линейная	Предсказание значений количественной зависимой переменной по значениям двух и более количественных независимых переменных
Многоуровневая	Предсказание значений зависимой переменной по значениям, имеющим иерархическую структуру (например, по данным об учениках с группировкой по классам и школам). Этот вид регрессии также называют иерархической, вложенной или смешанной
Многомерная	Предсказание значений более чем одной зависимой переменной по значениям одной и более независимых переменных
Логистическая	Предсказание значений категориальной зависимой переменной по значениям одной и более независимых переменных
Пуассона	Предсказание значений зависимой счетной переменной по значениям одной или более независимых переменных
Пропорциональных рисков Кокса	Предсказание времени до наступления события (смерти, аварии, рецидива) по значениям одной или более независимых переменных
Временных рядов	Моделирование временных рядов с коррелированными ошибками
Нелинейная	Предсказание значений количественной зависимой переменной по значениям одной и более независимых переменных с использованием нелинейной модели
Непараметрическая	Предсказание значений количественной зависимой переменной по значениям одной и более независимых переменных с использованием полученной из данных и не заданной заранее модели
Устойчивая	Предсказание значений количественной зависимой переменной по значениям одной и более независимых переменных с использованием метода, устойчивого к выбросам

В этой главе мы сосредоточимся на регрессионных методах, основанных на методе наименьших квадратов (МНК), включая простую линейную, полиномиальную и множественную линейную регрессию. Другие виды регрессии (включая логистическую и Пуассона) будут рассмотрены в главе 13.

8.1.1. Когда используется МНК-регрессия

В случае МНК-регрессии значения количественной зависимой переменной предсказываются на основании взвешенной суммы значений независимых переменных, где веса переменных оцениваются, исходя из данных. Давайте взглянем на конкретный пример, взятый из публикации Fwa (2006) с упрощениями.

Инженеру нужно выявить наиболее существенные факторы, влияющие на износ мостов (такие как срок службы, интенсивность движения, тип конструкции, использованные материалы и методы, качество постройки и погодные условия), и вывести математическую формулу, описывающую их влияние. Инженер собирает все необходимые сведения в репрезентативную выборку мостов и моделирует данные при помощи МНК-регрессии.

Процесс решения задачи в высшей степени интерактивный. Инженер подбирает разные модели, проверяет их соответствие статистическим допущениям, на которых основаны эти модели, исследует все неожиданные или отклоняющиеся от нормы факты и, наконец, выбирает «лучшую» модель из многих возможных. В случае успеха полученная модель поможет инженеру:

- сосредоточиться на важных переменных, определив, какие из изученных признаков полезны для предсказания износа мостов и какие из них наиболее важны;
- отыскать мосты, находящиеся, скорее всего, в плохом состоянии, применив полученное уравнение для оценки степени износа мостов в новых ситуациях (когда значения независимых переменных известны, а степень изношенности моста – нет);
- использовать преимущества интуитивной прозорливости и выявить необычные мосты. Если инженер обнаружит, что некоторые мосты изнашиваются быстрее или медленнее, чем ожидалось, то исследование таких «выбросов» может привести к важным открытиям, которые помогут понять механизмы износа мостов.

Мосты могут быть вам неинтересны. Я занимаюсь клинической психологией и статистикой, а о строительстве не знаю почти ничего. Однако основные принципы анализа применимы к удивительно широкому спектру задач в психологии, биологии и социологии. МНК-регрессия может помочь найти ответы на каждый из следующих вопросов:

- Как связаны засоленность поверхностного стока и площадь мощеных дорог (Montgomery, 2007)?
- Какие аспекты жизненного опыта игроков обусловливают чрезмерное увлечение массовыми ролевыми сетевыми играми (Hsu, Wen, & Wu, 2009)?
- Какие характеристики образовательной среды наиболее сильно влияют на успеваемость студентов?
- Как связаны артериальное давление и количество потребляемой соли с возрастом?
- Какой вклад в увеличение площади мегаполисов вносят стадионы и наличие профессиональных спортивных команд (Baade & Dye, 1990)?
- Какие факторы влияют на цену пива в разных штатах (Culbertson & Bradford, 1991)? (Этот вопрос определенно привлек ваше внимание!)

Решающее значение имеют наша способность сформулировать интересный вопрос и умение выбрать зависимую переменную для исследования и собрать нужные данные.

8.1.2. Что нужно знать

В оставшейся части этой главы я буду описывать функции R, используемые для подбора регрессионной модели МНК, оценки ее соответствия данным, проверки предположений, на которых построена модель, и выбора наилучшей модели из имеющихся. Предполагается, что у вас уже есть общее представление об МНК-регрессии. Однако я постарался свести число математических формул к минимуму и в большей степени сосредоточиться не на теории, а на практике. Существует ряд замечательных пособий, которые посвящены вошедшим в эту главу статистическим методам. Мои любимые – это «Applied Regression Analysis and Generalized Linear Models» Джона Фокса (John Fox) – в том, что касается теории, – и «An R and S-Plus Companion to Applied Regression» (2002) – в том, что касается практики. Эти два пособия послужили основным источником информации для данной главы. Хороший нетехнический обзор представлен Лихтом (Licht, 1995).

8.2. МНК-регрессия

На протяжении большей части этой главы мы будем предсказывать значения зависимой переменной по набору независимых (это также называют *регрессией* зависимой переменной по независимой – отсюда и название метода) с использованием МНК (метода наименьших квадратов). МНК-регрессия позволяет подбирать модели вида:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{1i} + \ldots + \hat{\beta}_k X_{ki} \quad i = 1 \ldots n,$$

где

- $\hat{Y}_i$ – предсказанное значение зависимой переменной для i-го наблюдения (а именно оценка среднего значения распределения Y по набору независимых переменных);
- X_{ki} – значение k-й независимой переменной в i-м наблюдении;
- $\hat{\beta}_0$ – свободный член уравнения (предсказанное значение Y при нулевом значении всех независимых переменных);
- $\hat{\beta}_k$ – регрессионный коэффициент для k-й независимой переменной (угол наклона прямой, описывающей изменение Y при изменении X_k на одну единицу).

Наша цель – подобрать параметры модели (свободный член и регрессионные коэффициенты), минимизирующие различия между реальными и предсказанными значениями зависимой переменной. В частности, параметры модели должны минимизировать сумму квадратов остатков:

$$\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2 = \sum_{i=1}^{n}\left(Y_i - \left(\widehat{\beta}_0 + \widehat{\beta}_1 X_{1i} + \ldots + \widehat{\beta}_k X_{ki}\right)\right)^2 = \sum_{i=1}^{n}\varepsilon_i^2.$$

Для правильной интерпретации коэффициентов МНК-модели данные должны удовлетворять следующим требованиям:

- *нормальность* – значения зависимой переменной должны подчиняться нормальному закону распределения при фиксированных значениях независимых переменных;
- *независимость* – значения Y_i не должны зависеть друг от друга;
- *линейность* – зависимая переменная линейно связана с независимыми;
- *гомоскедастичность* – дисперсия зависимой переменной постоянна при разных значениях независимых переменных. (Я мог бы назвать это свойство «однородность дисперсии», но употребление термина «гомоскедастичность» позволяет мне чувствовать себя умнее.)

Если эти требования нарушаются, то критерии значимости и доверительные интервалы могут быть вычислены неточно. Также подразумевается, что независимые переменные определяются и измеряются без ошибок, однако это требование обычно игнорируется на практике.

8.2.1. Подгонка регрессионных моделей при помощи lm()

Для подгонки регрессионных моделей в R обычно используется функция `lm()`. Она имеет следующий синтаксис:

```
myfit <- lm(formula, data)
```

где *formula* описывает вид подбираемой модели, а *data* – это таблица данных, используемая для создания модели. Полученный объект (в данном случае myfit) – это список, содержащий обширную информацию о модели. Формула *formula* обычно записывается в таком виде:

```
Y ~ X1 + X2 + ... + Xk
```

где ~ отделяет зависимую переменную слева от независимых переменных (разделенных знаками +) справа. Также в формуле можно использовать другие символы (табл. 8.2).

Таблица 8.2. Символы, часто используемые в формулах R

Символ	Описание
~	Отделяет зависимые переменные (слева) от независимых (справа). Например, предсказание значений y по значениям x, z и w можно представить так: `y ~ x + z + w`
+	Разделяет независимые переменные
:	Обозначает взаимосвязь (коллинеарность) между независимыми переменными. Предсказание значений y по значениям x и z с учетом взаимосвязи между x и z можно описать как `y ~ x + z + x:z`
*	Кратко обозначает все возможные взаимосвязи. Код `y ~ x * z * w` будет развернут в `y ~ x + z + w + x:z + x:w + z:w + x:z:w`
^	Обозначает взаимосвязь до определенного порядка. Код `y ~ (x + z + w)^2` будет развернут в `y ~ x + z + w + x:z + x:w + z:w`
.	Символ-заполнитель для всех переменных в таблице данных, кроме зависимой. Например, если таблица данных содержит переменные x, y, z и w, то код `y ~ .` будет означать `y ~ x + z + w`
-	Знак минус удаляет переменную из уравнения. Например, `y ~ (x + z + w)^2 - x:w` будет развернут в `y ~ x + z + w + x:z + z:w`
-1	Подавляет свободный член уравнения. Например, формула `y ~ x -1` создаст регрессионную модель, график которой будет проходить через начало координат
I()	Элемент в скобках интерпретируется как арифметическое выражение. Например, `y ~ x + (z + w)^2` означает `y ~ x + z + w + z:w`. Для сравнения: `y ~ x + I((z + w)^2)` означает `y ~ x + h`, где h – это новая переменная, полученная возведением в квадрат суммы z и w
function	В формулах можно использовать математические функции. Например, `log(y) ~ x + z + w` будет предсказывать значения `log(y)` по значениям x, z и w

В дополнение к lm() имеется еще несколько функций (табл. 8.3), которые могут пригодиться для выполнения простого или множественного регрессионного анализа. Каждая из них применяется к объекту, созданному вызовом функции lm(), чтобы получить дополнительную информацию о полученной модели.

Таблица 8.3. Другие функции, которые могут пригодиться при создании линейных моделей

Функция	Описание
`summary()`	Выводит подробную информацию о модели
`coefficients()`	Выводит параметры модели (свободный член и регрессионные коэффициенты)
`confint()`	Вычисляет доверительные интервалы для параметров модели (по умолчанию 95 %)
`fitted()`	Выводит значения, предсказанные моделью
`residuals()`	Выводит остатки для модели
`anova()`	Создает таблицу ANOVA (дисперсионного анализа) для модели или сравнивающую две либо более моделей
`vcov()`	Выводит ковариационную матрицу для параметров модели
`AIC()`	Вычисляет информационный критерий Акаике (Akaike Information Criterion)
`plot()`	Создает диагностические диаграммы для оценки адекватности модели
`predict()`	Использует модель для предсказания зависимой переменной по новому набору данных

Когда в регрессионной модели есть одна зависимая и одна независимая переменная, то она называется *простой линейной регрессией*. Когда есть одна зависимая переменная, но модель включает ее степени (например, X, X^2, X^3), то такая модель называется *полиномиальной регрессией*. Когда модель включает больше одной независимой переменной, то она называется *множественной регрессией*. Начнем с примера простой линейной регрессии, затем перейдем к примерам полиномиальной и множественной линейной регрессии и закончим примером множественной регрессии с учетом взаимосвязей между независимыми переменными.

8.2.2. Простая линейная регрессия

Рассмотрим пример использования функций, перечисленных в табл. 8.3, на простом примере. Набор данных `women`, входящий в состав дистрибутива R, содержит данные о росте и весе 15 женщин в возрасте от 30 до 39 лет. Наша цель – научиться предсказывать вес по величине роста, что поможет выявлять женщин с чрезмерным или недостаточным весом. Реализация регрессионного анализа приводится в листинге 8.1, а получившаяся диаграмма показана на рис. 8.1.

Листинг 8.1. Простая линейная регрессия

```
> fit <- lm(weight ~ height, data=women)
> summary(fit)

Call:
lm(formula=weight ~ height, data=women)

Residuals:
   Min     1Q Median     3Q    Max
-1.733 -1.133 -0.383  0.742  3.117

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -87.5167     5.9369   -14.7  1.7e-09 ***
height        3.4500     0.0911    37.9  1.1e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.53 on 13 degrees of freedom
Multiple R-squared: 0.991,      Adjusted R-squared: 0.99
F-statistic: 1.43e+03 on 1 and 13 DF,  p-value: 1.09e-14

> women$weight

 [1] 115 117 120 123 126 129 132 135 139 142 146 150 154 159 164

> fitted(fit)

     1      2      3      4      5      6      7      8      9
112.58 116.03 119.48 122.93 126.38 129.83 133.28 136.73 140.18
    10     11     12     13     14     15
143.63 147.08 150.53 153.98 157.43 160.88

> residuals(fit)

    1     2     3     4     5     6     7     8     9    10    11
 2.42  0.97  0.52  0.07 -0.38 -0.83 -1.28 -1.73 -1.18 -1.63 -1.08
   12    13    14    15
-0.53  0.02  1.57  3.12

> plot(women$height,women$weight,
       xlab="Height (in inches)",
       ylab="Weight (in pounds)")
> abline(fit)
```

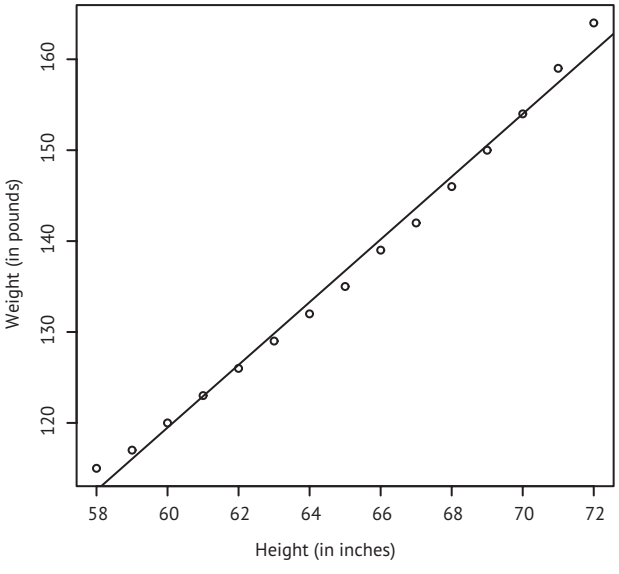


Рис. 8.1. Диаграмма рассеяния с регрессионной прямой, предсказывающей вес по росту

Из полученных результатов следует, что уравнение для предсказания веса по росту имеет следующий вид:

weight = –87.52 + 3.45 × *height*.

Поскольку нулевой рост невозможен, мы не будем пытаться интерпретировать свободный член. В данном случае это просто поправочная константа. Из столбца `Pr(>|t|)` видно, что коэффициент регрессии (`3.45`) статистически значимо отличается от нуля (`p < 0.001`) и означает, что на каждый дюйм[1] роста ожидается увеличение веса на 3.45 фунта[2]. Множественный коэффициент детерминации (`0.991`) означает, что модель объясняет 99.1 % дисперсии значений веса. Этот коэффициент равен квадрату коэффициента корреляции между реальными и предсказанными значениями[3]. Стандартную ошибку остатков (`1.53` фунта) можно интерпретировать как усредненную ошибку предсказания веса по росту с использованием данной модели. F-критерий позволяет проверить, предсказывают ли вместе взятые независимые переменные значения зависимой переменной выше уровня вероятности. Поскольку в простой регрессии имеется только одна независимая переменная, в этом примере F-критерий эквивалентен критерию Стьюдента для коэффициента регрессии при переменной `height`.

[1] 2,54 см. – *Прим. перев.*

[2] Почти 1,6 кг. – *Прим. перев.*

[3] Такая интерпретация множественного коэффициента детерминации справедлива только для простой линейной регрессии. – *Прим. перев.*

В иллюстративных целях я вывел реальные и предсказанные значения, а также остатки (разность между предсказанными и реальными значениями). Очевидно, что наибольшие остатки характерны для самых низких и высоких женщин, что также можно видеть на диаграмме (рис. 8.1).

На диаграмме заметно, что предсказание можно улучшить, используя линию с одним изгибом. Например, модель

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2$$

может лучше соответствовать данным. Полиномиальная регрессия позволяет предсказывать зависимую переменную по независимой, если эта связь имеет вид полинома *n*-й степени.

8.2.3. Полиномиальная регрессия

Из диаграммы на рис. 8.1 следует, что точность предсказания можно улучшить, если использовать уравнение квадратичной регрессии (то есть включить в него член X^2). Получить модель в виде квадратичного уравнения можно при помощи такого выражения:

```
fit2 <- lm(weight ~ height + I(height^2), data=women)
```

Тут нужно объяснить новое обозначение: `I(height^2)`. `height^2` добавляет в уравнение рост, возведенный в квадрат. Функция `I()` интерпретирует содержимое скобок как выражение на языке R. Это необходимо, потому что оператор `^` имеет в формулах особое значение (табл. 8.2), которое нужно подавить в данном случае.

В листинге 8.2 показан результат подгонки квадратичного уравнения.

Листинг 8.2. Полиномиальная регрессия

```
> fit2 <- lm(weight ~ height + I(height^2), data=women)
> summary(fit2)

Call:
lm(formula=weight ~ height + I(height^2), data=women)

Residuals:
    Min      1Q  Median      3Q     Max
-0.5094 -0.2961 -0.0094  0.2862  0.5971

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 261.87818   25.19677   10.39  2.4e-07 ***
height       -7.34832    0.77769   -9.45  6.6e-07 ***
I(height^2)   0.08306    0.00598   13.89  9.3e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.384 on 12 degrees of freedom
Multiple R-squared: 0.999,      Adjusted R-squared: 0.999
F-statistic: 1.14e+04 on 2 and 12 DF,  p-value: <2e-16

> plot(women$height,women$weight,
       xlab="Height (in inches)",
       ylab="Weight (in lbs)")
> lines(women$height,fitted(fit2))
```

Теперь регрессионное уравнение приобретает вид

$$weight = 261.88 - 7.35 \times height + 0.083 \times height^2$$

и оба регрессионных коэффициента оказываются значимыми на уровне `p < 0.0001`. Доля объясненной дисперсии в данном случае повысилась до 99,9 %. Значимость квадратного члена (`t = 13.89, p < .001`) указывает на то, что его включение в модель улучшило ее адекватность. Если посмотреть на диаграмму для второй модели (рис. 8.2), то можно заметить, что кривая лучше описывает реальные данные, чем прямая линия.

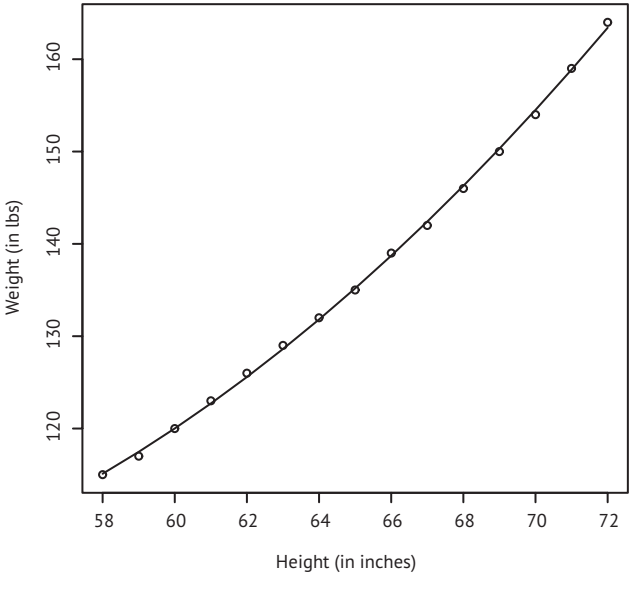


Рис. 8.2. Квадратичная регрессия точнее предсказывает значения веса по значениям роста

Линейные и нелинейные модели

Обратите внимание, что это полиномиальное уравнение по-прежнему попадает в разряд линейной регрессии. Эта модель линейная, поскольку в уравнение входит взвешенная сумма независимых переменных (в данном случае рост и квадрат роста). Даже такая модель:

$$\hat{Y}_i = \hat{\beta}_0 \times \log(X_1) + \hat{\beta}_2 \times \sin X_2$$

– считается линейной (исходя из ее параметров) и описывается формулой

$$Y \sim \log(X1) + \sin(X2).$$

А вот пример настоящей нелинейной модели:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 e^{x/\hat{\beta}_2}.$$

Нелинейные модели этого вида можно получить при помощи функции `nls()`.

В общем случае полиномиальная функция n-й степени соответствует кривой с $n - 1$ изгибами. Для подгонки кубического полинома нужно использовать выражение:

```
fit3 <- lm(weight ~ height + I(height^2) +I(height^3), data=women)
```

Возможно применение полиномов и более высоких степеней, но я редко сталкивался с необходимостью использовать члены в степени выше третьей.

8.2.4. Множественная линейная регрессия

Если существует больше одной независимой переменной, то простая линейная регрессия превращается во множественную линейную регрессию, а ход вычислений становится более сложным. Технически полиномиальная регрессия – это частный случай множественной регрессии. В квадратичной регрессии есть две независимые переменные (X и X^2), а в кубической – три независимые переменные (X, X^2, X^3). Рассмотрим более общий пример.

Для этого примера мы используем набор данных `state.x77`, входящий в состав дистрибутива R, и исследуем связь между уровнем преступности и другими характеристиками для каждого штата, включая численность населения, уровень неграмотности, средний доход и морозность (среднее число дней с отрицательной температурой).

Поскольку функция `lm()` требует передать ей таблицу данных (а набор данных `state.x77` хранится в виде матрицы), выполним необходимое преобразование, как показано ниже:

```
states <- as.data.frame(state.x77[,c("Murder", "Population",
                                     "Illiteracy", "Income", "Frost")])
```

Этот код создаст таблицу данных с именем `states`, содержащую интересующие нас переменные. Эта таблица будет использоваться на протяжении всей главы.

Важный первый шаг во множественной регрессии – исследование связей между парами переменных. Двухмерные корреляции вычисляются при помощи функции `cor()`, а диаграммы рассеяния

создаются при помощи функции `scatterplotMatrix()`[1] из пакета `car` (листинг 8.3 и рис. 8.4).

Листинг 8.3. Исследование связей между парами переменных

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                          "Illiteracy", "Income", "Frost")])

> cor(states)
           Murder Population Illiteracy Income Frost
Murder       1.00       0.34       0.70  -0.23 -0.54
Population   0.34       1.00       0.11   0.21 -0.33
Illiteracy   0.70       0.11       1.00  -0.44 -0.67
Income      -0.23       0.21      -0.44   1.00  0.23
Frost       -0.54      -0.33      -0.67   0.23  1.00

> library(car)
> scatterplotMatrix(states, smooth=FALSE, main="Scatter Plot Matrix")
```

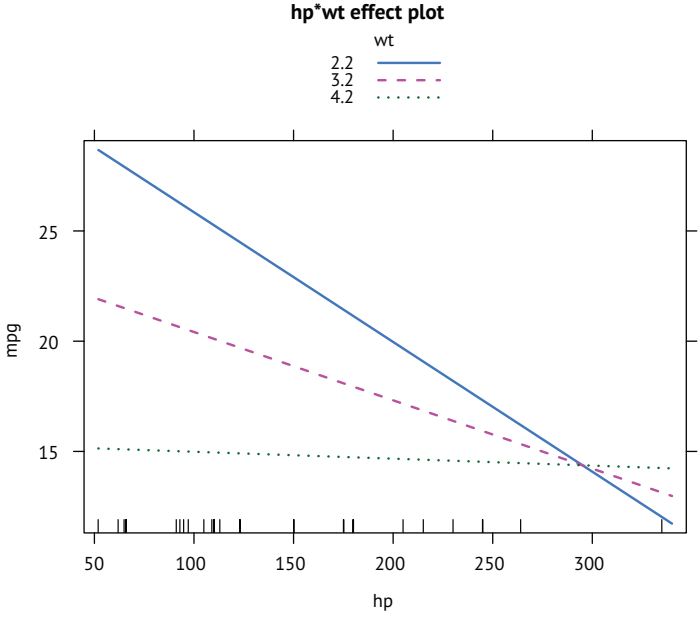


Рис. 8.4. Матрица диаграмм рассеяния значений зависимой и независимых переменных в данных о штатах, включающая регрессионные прямые и сглаживающие линии, а также распределения переменных (диаграмма ядерной оценки функции плотности и график-щетка)

По умолчанию функция `scatterplotMatrix()` выводит графики рассеяния переменных относительно друг друга по диагоналям

[1] В первых версиях пакета `car` эта функция называлась `scatterplot.matrix()`. – *Прим. перев.*

и накладывает на эти графики сглаживающие линии и линии линейной регрессии. Главная диагональ содержит графики плотности и графики-щетки для каждой переменной. Вывод сглаживающих линий можно подавить, передав параметр `smooth=FALSE`.

Как показано на рис. 8.4, уровень преступности (Murder) может иметь бимодальный характер, и каждая из независимых переменных в некоторой степени искажена. Уровень преступности растет с увеличением населения и неграмотности и снижается с ростом уровня доходов и морозности. В то же время в более холодных штатах уровень неграмотности ниже, меньше численность населения и выше доходы.

Теперь подберем модель множественной регрессии при помощи функции `lm()` (листинг 8.4).

Листинг 8.4. Множественная линейная регрессия

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                          "Illiteracy", "Income", "Frost")])

> fit <- lm(Murder ~ Population + Illiteracy + Income + Frost,
            data=states)
> summary(fit)

Call:
lm(formula=Murder ~ Population + Illiteracy + Income + Frost,
    data=states)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7960 -1.6495 -0.0811  1.4815  7.6210

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.23e+00   3.87e+00    0.32    0.751
Population  2.24e-04   9.05e-05    2.47    0.017 *
Illiteracy  4.14e+00   8.74e-01    4.74  2.2e-05 ***
Income      6.44e-05   6.84e-04    0.09    0.925
Frost       5.81e-04   1.01e-02    0.06    0.954
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.v 0.1 'v' 1

Residual standard error: 2.5 on 45 degrees of freedom
Multiple R-squared: 0.567,      Adjusted R-squared: 0.528
F-statistic: 14.7 on 4 and 45 DF,  p-value: 9.13e-08
```

Если существует больше одной независимой переменной, то регрессионные коэффициенты показывают, на сколько увеличится значение зависимой переменной при изменении данной независимой переменной на единицу, при условии что все остальные независимые переменные останутся неизменными. Например, ре-

грессионный коэффициент при переменной `Illiteracy` (уровень неграмотности) равен `4.14`. Это свидетельствует о том, что увеличение неграмотности на 1 % влечет увеличение уровня преступности на 4,14 % при постоянных значениях численности населения, дохода и морозности. Этот коэффициент статистически значимо отличается от нуля при `p < 0.0001`. С другой стороны, регрессионный коэффициент при переменной `Frost` (морозность) статистически не отличается от нуля. Это говорит о том, что морозность и уровень преступности не имеют линейной зависимости при неизменных значениях всех прочих переменных. Взятые вместе, независимые переменные могут объяснить 57 % дисперсии уровня преступности по штатам.

До этого момента мы предполагали, что независимые переменные не взаимосвязаны между собой. В следующем разделе мы рассмотрим пример, когда это не так.

8.2.5. Множественная линейная регрессия с учетом взаимосвязей

Иногда можно получить довольно интересные результаты, добавив учет взаимосвязей между независимыми переменными. Рассмотрим данные об автомобилях из таблицы данных `mtcars`. Допустим, что нас интересует влияние веса автомобиля и мощности двигателя на расход топлива. Можно попробовать подобрать регрессионную модель, включающую обе независимые переменные, а также учитывающую взаимосвязь между ними, как показано в листинге 8.5.

Листинг 8.5. Множественная линейная регрессия со значимым эффектом взаимосвязи

```
> fit <- lm(mpg ~ hp + wt + hp:wt, data=mtcars)
> summary(fit)

Call:
lm(formula=mpg ~ hp + wt + hp:wt, data=mtcars)

Residuals:
   Min     1Q Median     3Q    Max
-3.063 -1.649 -0.736  1.421  4.551

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 49.80842    3.60516   13.82  5.0e-14 ***
hp          -0.12010    0.02470   -4.86  4.0e-05 ***
wt          -8.21662    1.26971   -6.47  5.2e-07 ***
hp:wt        0.02785    0.00742    3.75  0.00081 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 2.1 on 28 degrees of freedom
Multiple R-squared: 0.885,      Adjusted R-squared: 0.872
F-statistic: 71.7 on 3 and 28 DF,  p-value: 2.98e-13
```

Судя по столбцу `Pr(>|t|)`, мощность двигателя и вес автомобиля машины имеют значимую взаимосвязь. Что это означает? Значимая взаимосвязь между двумя независимыми переменными свидетельствует о том, что влияние одной независимой переменной на зависимую отчасти обусловлено значениями другой независимой переменной. В данном случае характер зависимости между расходом топлива и мощностью двигателя не одинаков для автомобилей разного веса.

Получившаяся модель предсказания величины пробега на одном галлоне топлива имеет вид: `mpg = 49.81 – 0.12 × hp – 8.22 × wt + 0.03 × hp × wt`. Для интерпретации взаимовлияния можно попробовать подставлять разные значения веса (`wt`), чтобы упростить уравнение. Например, можно попробовать взять среднее значение `wt` (`3.2`) и значения на одно стандартное отклонение меньше и больше среднего (`2.2` и `4.2` соответственно). Для `wt=2.2` уравнение упрощается до: `mpg = 49.81 – 0.12 × hp – 8.22 × (2.2) + 0.03 × hp × (2.2) = 31.41 – 0.06 × hp`. Для `wt=3.2` – до: `mpg = 23.37 – 0.03 × hp`. Наконец, для `wt=4.2` – до: `mpg = 15.33 – 0.003 × hp`. Как видите, с увеличением веса (`2.2`, `3.2`, `4.2`) ожидаемое изменение `mpg` на единицу изменения `hp` уменьшается (`0.06`, `0.03`, `0.003`).

Взаимовлияния можно визуализировать при помощи функции `effect()` из пакета `effect`. Она имеет следующий синтаксис:

```
plot(effect(term, mod,, xlevels), multiline=TRUE)
```

где *`term`* – это член модели, который нужно отобразить на диаграмме, *`mod`* – модель, полученная функцией `lm()`, а *`xlevels`* – список переменных, значения которых будут фиксированы, и самих этих значений. Параметр `multiline=TRUE` позволяет наложить на диаграмму линии, а параметр `lines` определяет тип каждой линии (1 = сплошная, 2 = пунктирная, 3 = точечная и т. д.). Для предыдущей модели вывод диаграммы выглядит так:

```
library(effects)
plot(effect("hp:wt", fit,, list(wt=c(2.2,3.2,4.2))),
     lines=c(1,2,3), multiline=TRUE)
```

а сама диаграмма показана на рис. 8.4.

Как можно заметить на диаграмме, с увеличением веса автомобиля взаимовлияние между мощностью мотора и расходом топлива постепенно ослабевает. Для `wt=4.2` линия почти горизонтальная, что говорит о незначительном изменении значений `mpg` при увеличении `hp`.

К сожалению, подгонка модели – это только первый шаг анализа. Прежде чем делать какие-либо выводы, после этого шага нужно оценить, соответствуют ли ваши данные статистическим ограничениям примененного метода. Это тема следующего раздела.

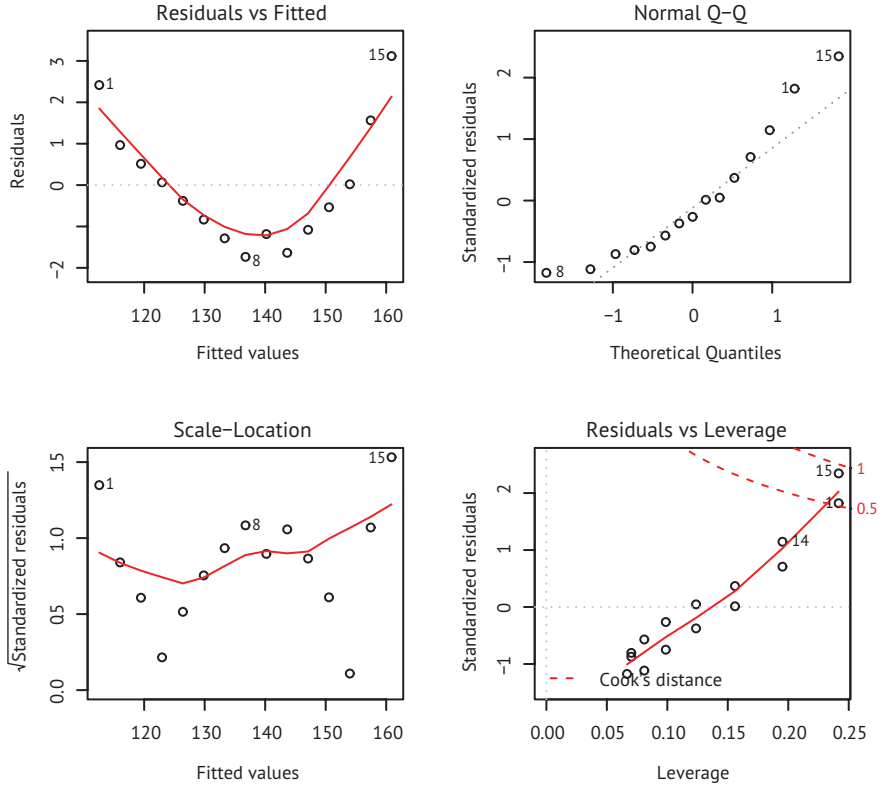


Рис. 8.5. Диаграмма взаимовлияния для hp*wt. Показана взаимозависимость между mpg и hp для трех значений wt

8.3. *Диагностика регрессионных моделей*

В предыдущем разделе мы использовали функцию `lm()` для подгонки МНК-модели и функцию `summary()` для получения параметров этой модели и вычисления характеризующих ее статистик.

К сожалению, ничто из этого не говорит о степени адекватности модели. Степень уверенности в параметрах регрессии зависит от того, насколько модель удовлетворяет ограничениям МНК. Функция `summary()`, приведенная в листинге 8.4, описывает модель, но она ничего не сообщает о том, насколько эта модель соответствует лежащим в ее основе статистическим ограничениям.

Почему это важно? Ошибки в данных или неверно определенные взаимосвязи между зависимыми и независимой переменными могут привести к выбору в значительной степени неточной модели. С одной стороны, вы можете решить, что между зависимой и независимой переменными нет связи, в то время как на самом деле она существует. С другой стороны, можно прийти к заключению, что зависимая и независимая переменные связаны, а на са-

мом деле это не так! Вы также можете получить модель, которая при применении к реальным данным будет давать предсказания со значительными ошибками.

Посмотрим на результат применения функции confint() к набору данных states для решения задачи по множественной регрессии из раздела 8.2.4.

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                          "Illiteracy", "Income", "Frost")])
> fit <- lm(Murder ~ Population + Illiteracy + Income + Frost, data=states)
> confint(fit)
                2.5 %   97.5 %
(Intercept) -6.55e+00 9.021318
Population   4.14e-05 0.000406
Illiteracy   2.38e+00 5.903874
Income      -1.31e-03 0.001441
Frost       -1.97e-02 0.020830
```

Полученный результат свидетельствует о том, что можно быть на 95 % уверенными в том, что интервал [2.38, 5.90] содержит реальное значение, на которое изменяется уровень преступности, при изменении уровня неграмотности на 1 %. Кроме того, поскольку доверительный интервал для морозности (Frost) содержит ноль, можно заключить, что изменения этого параметра не связаны с уровнем преступности при постоянных значениях прочих переменных. Однако уверенность в этих результатах сильна настолько, насколько выполняются статистические допущения, лежащие в основе модели.

Набор методов под названием *диагностика регрессионных моделей* предоставляет необходимые инструменты для оценки адекватности регрессионной модели и может помочь обнаружить и устранить проблемы. Начнем со стандартного подхода с использованием функций, входящих в базовую версию R. Затем мы рассмотрим более новые, усовершенствованные методы, реализованные в пакете car.

8.3.1. Стандартный подход

В базовой версии R реализованы многочисленные методы проверки статистических допущений. Наиболее распространенный подход – применение функции plot() к объекту, представляющему результат действия функции lm(), в результате чего будут выведены четыре диаграммы, по которым можно оценить адекватность модели. Вот как этот подход можно применить к примеру простой линейной регрессии:

```
fit <- lm(weight ~ height, data=women)
par(mfrow=c(2,2))
plot(fit)
par(mfrow=c(1,1))
```

Этот код выведет диаграммы, представленные на рис. 8.6. Выражение `par(mfrow=c(2,2))` используется для размещения четырех диаграмм, создаваемых функцией `plot()`, на одной большой диаграмме 2×2. Второй вызов функции `par()` вернет одну диаграмму.

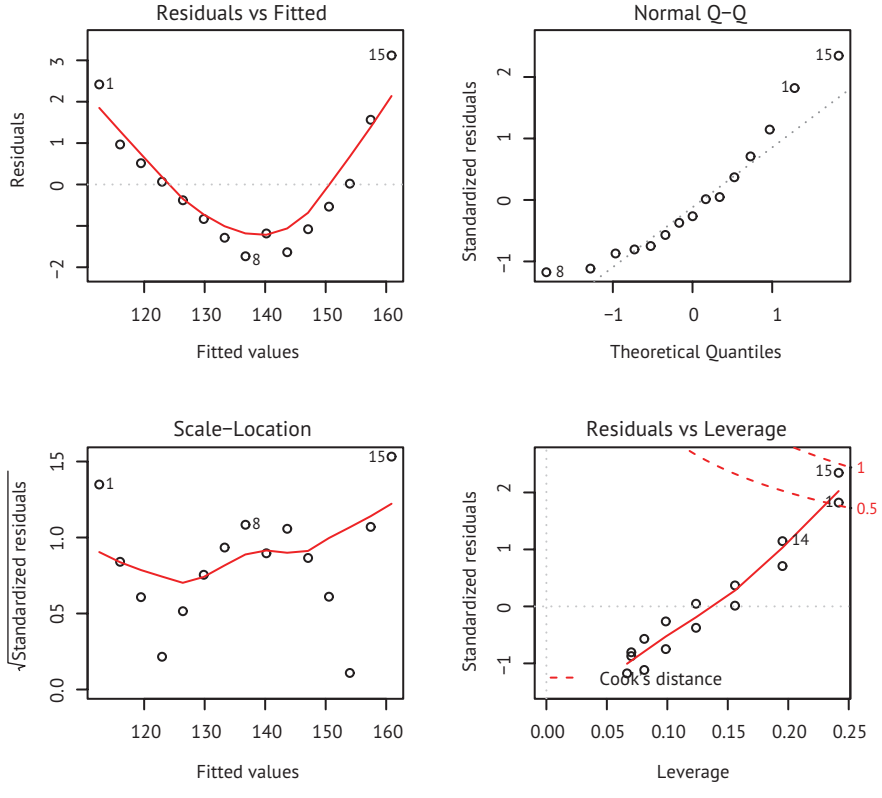


Рис. 8.5. Диагностические диаграммы для оценки адекватности регрессии веса по росту

Чтобы понять эти диаграммы, нужно вспомнить допущения МНК-регрессии:

- *нормальность*. Если значения зависимой переменной нормально распределены при постоянных значениях независимых переменных, тогда и остатки (residuals) должны быть нормально распределены со средним значением 0. Графическая проверка данных на нормальность (Normal Q-Q – справа сверху) – это график распределения вероятностей стандартизованных остатков, наложенный на значения, которые можно было бы ожидать при нормальном распределении. Если допущение о нормальном распределении выполняется, то точки на этой диаграмме должны лежать на прямой с углом наклона в 45°.

Поскольку здесь этого не наблюдается, то можно заключить, что данное допущение не выполняется;

- *независимость*. Глядя на эти диаграммы, нельзя сказать, насколько значения прогнозируемой переменной независимы. Для этого нужно понимать, как были собраны данные. У нас нет никаких оснований полагать, что вес одной женщины зависит от веса другой женщины. Если обнаружится, что данные собраны для членов одной семьи, то вам придется изменить предположение о независимости;
- *линейность*. Если зависимая переменная линейно связана с независимой, то связь между остатками и предсказанными (то есть подогнанными) значениями отсутствует. Другими словами, модель должна отражать всю закономерную изменчивость в данных, учитывая все, кроме белого шума. На диаграмме зависимости остатков от предсказанных значений (вверху слева) ясно видна нелинейная зависимость, что позволяет задуматься о добавлении квадратного члена в уравнение регрессии;
- *гомоскедастичность*. Если допущение о постоянной изменчивости выполняется, то точки на нижней левой диаграмме должны располагаться в форме полосы вокруг горизонтальной линии. Похоже, что это допущение выполняется.

Наконец, диаграмма зависимости остатков от «показателя напряженности» (англ. leverage; слева внизу) содержит информацию о наблюдениях, на которые, возможно, следует обратить особое внимание. Диаграмма выявляет выбросы, точки высокой напряженности и влиятельные наблюдения. Если говорить более конкретно, то:

- выброс – это значение, которое плохо предсказывается подобранной моделью (то есть имеет большой положительный или отрицательный остаток);
- наблюдение с большим значением напряженности описывается необычной комбинацией независимых переменных. Таким образом, это выброс в пространстве независимых переменных. Значения зависимой переменной не используются при вычислении напряженности;
- влиятельное наблюдение – это значение, которое вносит непропорциональный вклад в расчет параметров модели. Влиятельные наблюдения выявляются при помощи статистики, называемой расстоянием Кука (Cook's distance, Cook's D).

Честно говоря, диаграмма зависимости остатков от напряженности (справа внизу) кажется мне сложной для восприятия и малополезной. В следующих разделах вы увидите, как можно представить эту информацию более удачно.

Эти стандартные диагностические диаграммы полезны, но в настоящее время в R имеются более совершенные инструменты, и я рекомендую использовать их вместо `plot(fit)`.

8.3.2. Усовершенствованный подход

В пакете `car` реализован ряд функций, значительно расширяющих возможности подгонки и оценки регрессионных моделей (табл. 8.4).

Таблица 8.4. Функции для диагностики регрессионных моделей (пакет car)

Функция	Описание
`qqPlot()`	Диаграмма сравнения квантилей
`durbinWatsonTest()`	Критерий Дарбина–Уотсона (Durbin-Watson) на автокорреляцию в остатках
`crPlots()`	Диаграмма компонент и остатков
`ncvTest()`	Проверка неоднородности дисперсии остатков
`spreadLevelPlot()`	Диаграмма для обнаружения неоднородности дисперсии остатков (spread-level plot)
`outlierTest()`	Критерий Бонферрони для проверки на выбросы
`avPlots()`	Диаграммы добавленных переменных
`influencePlot()`	Диаграмма влияния наблюдений на регрессию
`vif()`	Фактор инфляции дисперсии

Рассмотрим все эти методы по очереди, применив их к нашему примеру множественной регрессии.

Нормальность

Функция `qqPlot()` – это более точный метод проверки предположения о нормальности, по сравнению с функцией `plot()`. Она изображает связь между остатками Стьюдента (также называемыми *исключенными остатками Стьюдента*, или *остатками, вычисленными методом последовательного исключения значений*, – jackknifed residuals) и квантилями распределения Стьюдента с $n - p - 1$ степенями свободы, где n – это объем выборки, а p – число параметров регрессии (включая свободный член). Вот программный код, генерирующий эту диаграмму:

```
library(car)
states <- as.data.frame(state.x77[,c("Murder", "Population",
                        "Illiteracy", "Income", "Frost")])
fit <- lm(Murder ~ Population + Illiteracy + Income + Frost, data=states)
qqPlot(fit, labels=row.names(states), id=list(method="identify"),
       simulate=TRUE, main="Q-Q Plot")
```

Функция `qqPlot` создает график распределения вероятностей, изображенный на рис. 8.6. Параметр `id=list(method="identify")` делает диаграмму интерактивной – вы можете щелкать мышью на точках графика, чтобы увидеть их значения, заданные параметром `labels`. Нажатие клавиши Esc или щелчок на кнопке Finish (Стоп) в правом верхнем углу диаграммы отключает интерактивный режим. В данном случае я выбрал точку, соответствующую штату Невада. При вызове с параметром `simulate=TRUE` на графике также изображаются 95%-ные доверительные границы, полученные с использованием параметрических бутстреп-методов (описаны в главе 12).

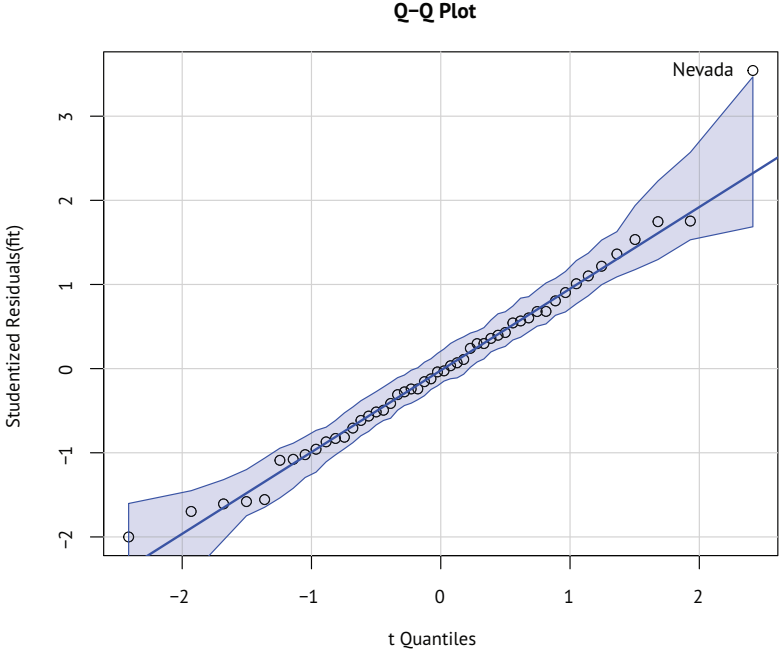


Рис. 8.6. Графическая проверка нормальности распределения (Q-Q-диаграмма)

За исключением штата Невада, все остальные точки располагаются близко к линии и находятся внутри доверительных границ, что свидетельствует о нормальности распределения. Но нам определенно нужно взглянуть на Неваду. Этот штат характеризуется большим положительным остатком (по сравнению с предсказанным значением), показывающим, что модель недооценивает уровень преступности в этом штате. А именно:

```
> states["Nevada",]

       Murder Population Illiteracy Income Frost
Nevada   11.5        590        0.5   5149   188
```

```
> fitted(fit)["Nevada"]

  Nevada
3.878958

> residuals(fit)["Nevada"]

  Nevada
7.621042

> rstudent(fit)["Nevada"]

  Nevada
3.542929
```

Как видите, уровень преступности составляет 11,5 %, но модель предсказывает уровень 3,9 %. Возникает вопрос: «Почему в Неваде уровень преступности выше, чем это предсказано по численности населения, уровню дохода, неграмотности и морозности?» Кто-нибудь (кто не смотрел фильм «Казино») хочет высказать предположение?

Независимость остатков

Как отмечалось выше, лучший способ понять, не связаны ли между собой значения зависимой переменной (и, таким образом, остатки), – выяснить, как происходил сбор данных. Например, временны́м рядам часто свойственна автокорреляция – наблюдения, сделанные в короткие отрезки времени, могут сильнее коррелировать друг с другом, чем разнесенные во времени. В пакете `car` есть функция для проверки критерия Дарбина–Уотсона на наличие автокорреляции в остатках. Этот критерий можно применить к нашей задаче множественной регрессии, как показано ниже:

```
> durbinWatsonTest(fit)
 lag Autocorrelation D-W Statistic p-value
   1          -0.201          2.32   0.282
 Alternative hypothesis: rho != 0
```

Высокое значение вероятности статистической ошибки первого рода (`p = 0.282`) свидетельствует об отсутствии автокорреляции и, следовательно, о независимости остатков. Значение интервала (`lag = 1`) говорит о том, что каждое значение в наборе данных сравнивается с соседним, следующим за ним значением. Однако, будучи предназначенным для временных рядов, этот критерий не так хорошо подходит для других типов данных. Учтите, что функция `durbinWatsonTest()` для вычисления `p` использует бутстреп-метод (глава 12). Если не добавить параметр `simulate=FALSE`, то результаты будут немного различаться от раза к разу.

Линейность

Наличие нелинейной связи между зависимой и независимыми переменными можно проверить при помощи диаграмм компонент и остатков (также известных под названием *диаграммы частных остатков*). Диаграммы создаются при помощи функции `crPlots()` из пакета `car`. Они помогают найти любые систематические отклонения от заданной линейной модели.

Для создания диаграммы компонент и остатков для переменной k нужно отобразить точки зависимости

$$\varepsilon_i + \hat{\beta}_k \times X_{ik} \text{ от } X_{ik},$$

где остатки основаны на полной модели (содержащей все независимые переменные), а $i = 1 \ldots n$. Прямая линия на каждой диаграмме соответствует зависимости $\hat{\beta}_k \times X_{ik}$ от X_{ik}. Непараметрическое сглаживание обсуждается в главе 11. Эти диаграммы создаются при помощи следующих вызовов функций:

```
> library(car)
> crPlots(fit)
```

Полученные диаграммы показаны на рис. 8.7. Наблюдаемая нелинейность свидетельствует о том, что функциональная форма независимой переменной была неверно смоделирована в уравнении. В этом случае может потребоваться добавить нелинейные составляющие, такие как полиномиальные члены, преобразовать одну или более переменных (например, использовать `log(X)` вместо `X`) или же отказаться от линейной регрессии в пользу какой-нибудь другой ее разновидности. Преобразования обсуждаются ниже в этой главе.

Диаграммы компонент и остатков подтверждают, что требование линейности соблюдено. Вид линейной модели подходит для данного набора данных.

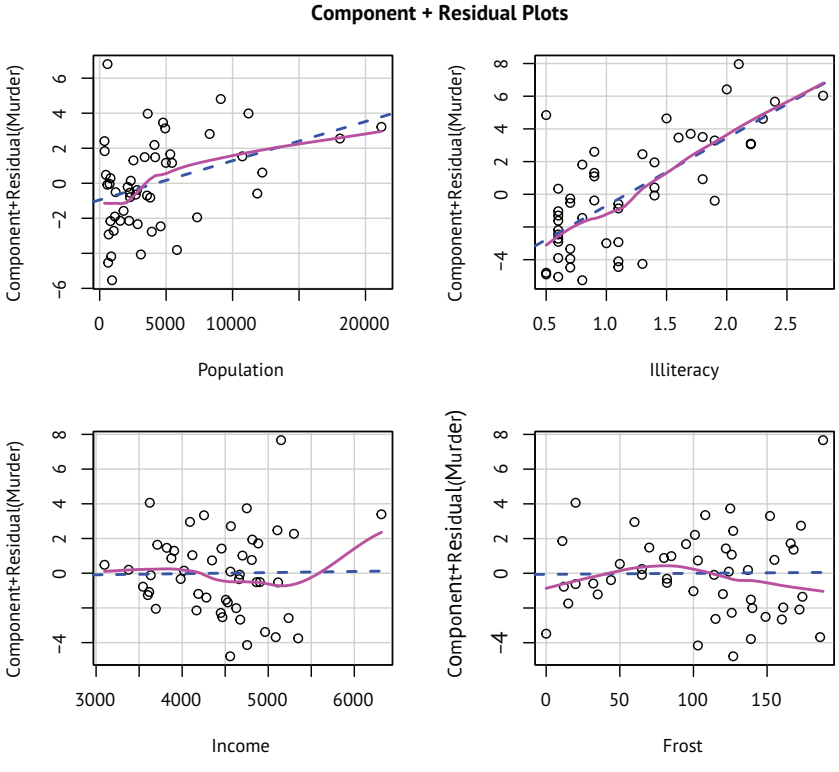


Рис. 8.7. Диаграммы компонент и остатков для регрессии уровня преступности по характеристикам штатов

Гомоскедастичность

В пакете `car` также имеются две полезные функции для обнаружения неоднородности дисперсии остатков. Функция `ncvTest()` позволяет проверить гипотезу о постоянстве дисперсии остатков как альтернативу тому, что дисперсия остатков изменяется в зависимости от подобранных значений. Статистически значимый результат[1] свидетельствует о гетероскедастичности (неоднородности дисперсии остатков).

Функция `spreadLevelPlot()` создает диаграмму рассеяния для абсолютных значений стандартизованных остатков и подобранных значений с наложенной регрессионной прямой. Применение обеих функций показано в листинге 8.6.

Листинг 8.6. Проверка на гомоскедастичность

```
> library(car)
> ncvTest(fit)
```

[1] $p < 0.05$. – *Прим. перев.*

```
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare=1.7    Df=1     p=0.19

> spreadLevelPlot(fit)

Suggested power transformation:  1.2
```

Критерий незначим (`p = 0.19`), что свидетельствует о выполнении условия однородности дисперсии. То же самое можно видеть на диаграмме (рис. 8.8). Точки беспорядочно располагаются в виде горизонтальной полосы вдоль горизонтальной регрессионной прямой. Если бы требование гомоскедастичности не выполнялось, мы увидели бы не горизонтальную прямую. Предлагаемое степенно́е преобразование (`Suggested power transformation`) в листинге 8.6 – это степень $p(Y^p)$, возведение в которую устраняет неоднородность дисперсии остатков. Например, если бы на диаграмме был показан нелинейный тренд и предлагаемое степенное преобразование было `0.5`, то использование $\sqrt{Y}$ вместо `Y` в уравнении регрессии могло бы дать модель, удовлетворяющую требованию гомоскедастичности. Если предлагаемая степень была бы равна `0`, то нужно было использовать логарифмическое преобразование. В данном случае признаки гетероскедастичности отсутствуют, и предлагаемая степень близка к 1 (никакого преобразования не требуется).

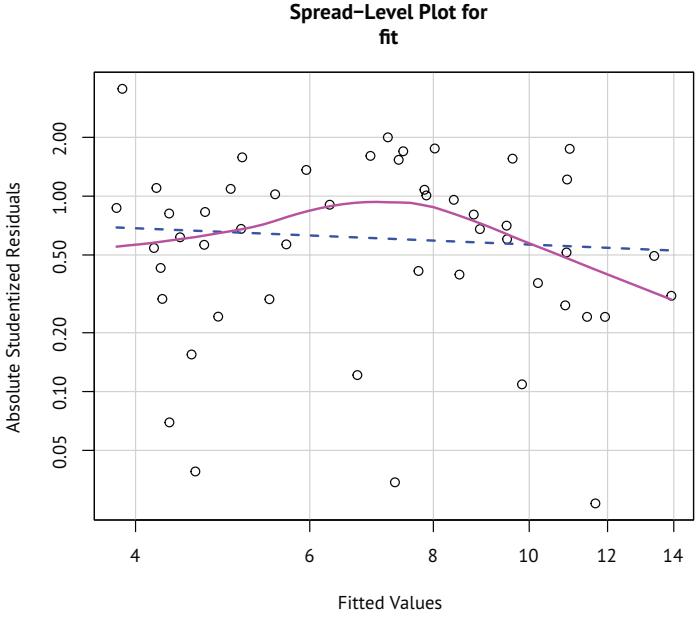


Рис. 8.8. Диаграмма для проверки однородности дисперсии остатков

8.3.3. Мультиколлинеарность

Прежде чем покинуть этот раздел, посвященный диагностике регрессионных моделей, рассмотрим еще одну проблему, не имеющую прямого отношения к статистическим допущениям, но помогающую интерпретировать результаты множественной регрессии. Представьте, что мы проводим исследование силы хвата. Набор данных включает независимые переменные, определяющие дату рождения и возраст. Вы вычисляете регрессионную модель силы хвата в зависимости от даты рождения и возраста и обнаруживаете значимый общий F-критерий при `p < 0.001`. Но, исследовав отдельные коэффициенты регрессии при дате рождения DOB и возрасте, обнаруживаете, что они оба незначимы (то есть нет никаких доказательств, что любой из них связан с силой хвата). Как такое получилось?

Проблема в том, что дата рождения и возраст идеально коррелируют в пределах ошибки округления. Коэффициент регрессии измеряет влияние одной независимой переменной на зависимую переменную, сохраняя неизменными все остальные независимые переменные. Это равносильно рассмотрению взаимосвязи между силой хвата и возрастом при сохранении возраста постоянным. Эта проблема называется *мультиколлинеарностью* (взаимозависимостью независимых переменных). Она приводит к большим доверительным интервалам для параметров модели и затрудняет интерпретацию отдельных коэффициентов.

Мультиколлинеарность можно обнаружить с помощью статистики, называемой коэффициентом инфляции дисперсии (Variance In ation Factor, VIF). Для любой независимой переменной квадратный корень из VIF сообщает степень расширения доверительного интервала для параметра регрессии при этой переменной по сравнению с моделью с некоррелированными независимыми переменными (отсюда и название). Значения VIF вычисляются с помощью функции `vif()` из пакета `car`. Как правило, значение VIF > 10 указывает на проблему мультиколлинеарности. В листинге 8.7 приводится соответствующий код. Как показывают результаты, мультиколлинеарность не является проблемой для этих переменных-предикторов.

Листинг 8.7. Вычисление коэффициента мультиколлинеарности

```
> library(car)
> vif(fit)

Population Illiteracy     Income      Frost
       1.2        2.2        1.3        2.1

> vif(fit) > 10 # проблема?
Population Illiteracy     Income      Frost
     FALSE      FALSE      FALSE      FALSE
```

8.4. Необычные наблюдения

Всеобъемлющий регрессионный анализ включает и анализ необычных наблюдений, а именно выбросов, наблюдений с высокой напряженностью и влиятельных наблюдений. Это те данные, которые требуют дальнейшего изучения: либо потому, что они каким-то образом отличаются от прочих, либо потому, что они значительно влияют на общие результаты. Рассмотрим их по очереди.

8.4.1. Выбросы

Выбросы – это наблюдения, которые плохо предсказываются моделью. Они характеризуются большими положительными или отрицательными остатками ($Y_i - \hat{Y}_i$). Положительные остатки свидетельствуют о том, что модель недооценивает зависимую переменную, а отрицательные остатки – признак переоценки.

Вы уже познакомились с одним способом обнаружения выбросов. Точки на рис. 8.6, находившиеся за пределами доверительной области, мы интерпретировали как выбросы. Приближенный метод оценки таков: стандартизованные остатки больше 2 или меньше –2 заслуживают внимания.

В пакете `car` имеется также статистический критерий для проверки на выбросы. Функция `outlierTest()` вычисляет значение вероятности статистической ошибки первого рода с поправкой Бонферрони для наибольшего остатка Стьюдента:

```
> library(car)
> outlierTest(fit)
       rstudent unadjusted p-value Bonferroni p
Nevada      3.5            0.00095        0.048
```

Здесь можно видеть, что штат Невада классифицирован как выброс (`p = 0.048`). Обратите внимание, что эта функция проверяет значимость одного наибольшего (положительного или отрицательного) остатка как выброса. Если проверка не дала значимого результата, то в данных нет выбросов. А если результат значимый, то нужно удалить выброс[1] и повторить проверку, чтобы убедиться в отсутствии других выбросов.

8.4.2. Точки высокой напряженности

Точки высокой напряженности – это выбросы относительно других независимых переменных. Иными словами, они характеризуются необычным сочетанием значений независимых переменных. Значение зависимой переменной не используется при вычислении напряженности.

Наблюдения с высокой напряженностью идентифицируются при помощи *показателя влияния* (hat statisctic). Для заданного набо-

[1] Имеется в виду наибольший по модулю выброс. – *Прим. перев.*

ра данных среднее значение этой статистики вычисляется как p/n, где p – число параметров в модели (включая свободный член), а n – размер выборки. Грубо говоря, наблюдения, для которых значение этой статистики превышает среднее в два или три раза, должны анализироваться отдельно. Следующий код строит диаграмму значений показателя влияния:

```
hat.plot <- function(fit) {
              p <- length(coefficients(fit))
              n <- length(fitted(fit))
              plot(hatvalues(fit), main="Index Plot of Hat Values")
              abline(h=c(2,3)*p/n, col="red", lty=2)
              identify(1:n, hatvalues(fit), names(hatvalues(fit)))
            }
hat.plot(fit)
```

Получившаяся диаграмма показана на рис. 8.9.

Горизонтальные линии соответствуют значениям, вдвое и втрое превышающим среднее. Функция ввода координат дает возможность выводить подписи точек в интерактивном режиме. Щелкая мышью на интересующих вас точках, можно выводить их подписи, пока не будет нажата клавиша Esc или кнопка Finish (Стоп) в правом верхнем углу диаграммы.

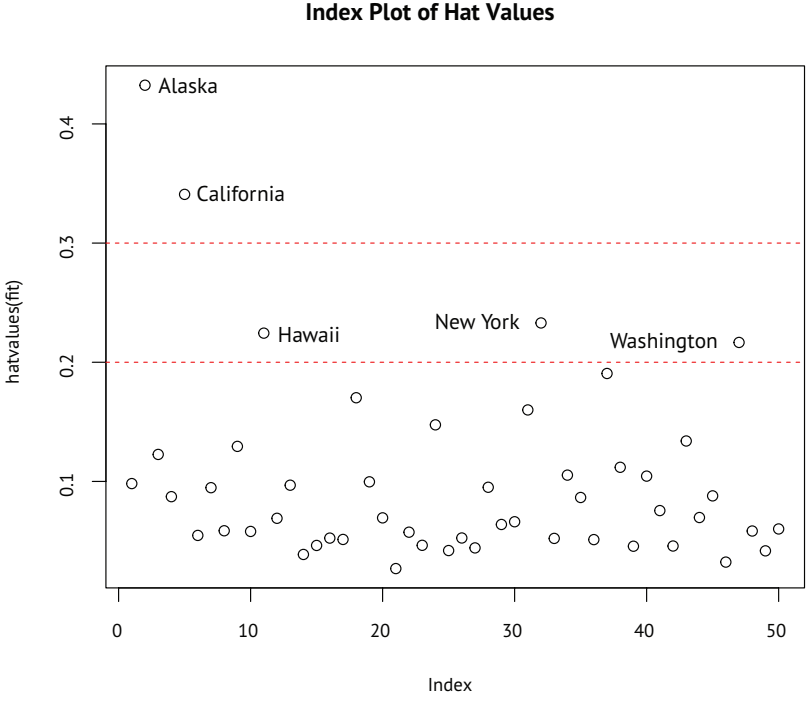


Рис. 8.9. Выявление наблюдений с высокой напряженностью: диаграмма значений показателя влияния для каждого наблюдения (их порядковые номера отложены по оси абсцисс)

Здесь видно, что Аляска и Калифорния – особенно необычные наблюдения, судя по значениям независимых переменных. Для Аляски характерны более высокие доходы и более низкие численность населения и температуры. В Калифорнии, наряду с более высокими доходами, отмечены высокие численность населения и температура. Эти два штата выделяются на фоне остальных 48.

Наблюдения с высокой напряженностью могут быть, а могут и не быть влиятельными. Это зависит от того, являются ли они в то же время выбросами.

8.4.3. Влиятельные наблюдения

Влиятельные наблюдения – это наблюдения, оказывающие непропорционально большое влияние на значения параметров модели. Представьте, что вы обнаружили значительные изменения модели при удалении единственного наблюдения. Это должно стать поводом для проверки данных на наличие влиятельных наблюдений.

Существуют два метода обнаружения влиятельных наблюдений: расстояние Кука (или D-статистика) и диаграммы добавленных переменных. Грубо говоря, значения расстояния Кука, превышающие $4/(n-k-1)$, где n – объем выборки, а k – число независимых переменных, свидетельствуют о влиятельных наблюдениях. Построить диаграмму расстояний Кука (рис. 8.10) можно так:

```
cutoff <- 4/(nrow(states)-length(fit$coefficients)-2)
plot(fit, which=4, cook.levels=cutoff)
abline(h=cutoff, lty=2, col="red")
```

На диаграмме видно, что Аляска, Гавайи и Невада – это влиятельные наблюдения. Удаление этих штатов заметно влияет на значения свободного члена и углового коэффициента регрессионной модели. Отметим, что, несмотря на полезность применения чувствительных методов для поиска влиятельных значений, мне кажется более разумным использовать пороговое значение расстояния Кука, равное 1, а не $4/(n-k-1)$. Если использовать критерий $D = 1$, то в данном наборе данных влиятельные значения не будут выявлены.

Диаграммы расстояния Кука помогают обнаружить влиятельные наблюдения, но не позволяют понять, как эти наблюдения влияют на модель. В этой ситуации приходят на выручку диаграммы добавленных переменных. Для одной зависимой и k независимых переменных описанным ниже способом создается k диаграмм добавленных переменных.

Для каждой независимой переменной X_k отображаются остатки от регрессии зависимой переменной по остальным $k-1$ независимым переменным. Такие диаграммы добавленных переменных можно построить при помощи функции `avPlots()` из пакета `car`:

```
library(car)
avPlots(fit, ask=FALSE, d=list(method="identify"))
```

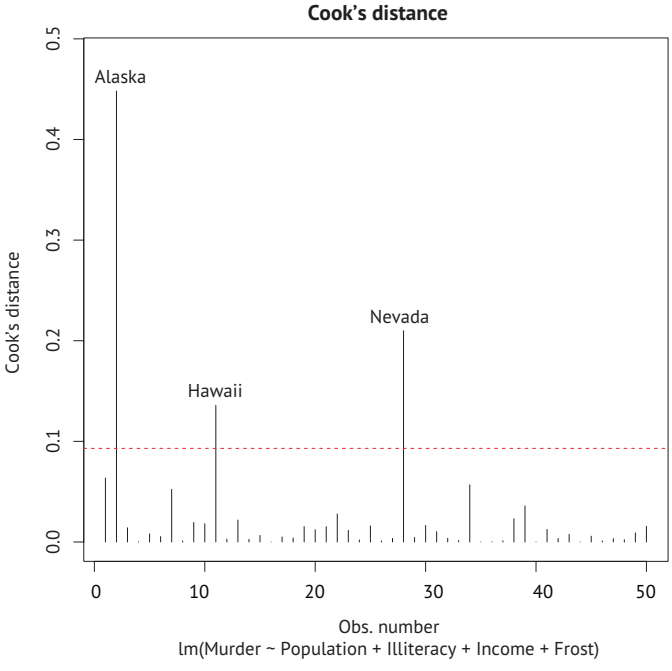


Рис. 8.10. Диаграмма расстояний Кука для обнаружения влиятельных наблюдений

Полученные диаграммы показаны на рис. 8.11. Эти диаграммы создаются одновременно, и пользователи могут щелкать на точках, чтобы вывести соответствующие им подписи. Нажмите Esc или щелкните на кнопке Finish (Стоп) вверху справа на диаграмме, чтобы перейти к следующей диаграмме. Здесь я вывел название Alaska (Аляска) на нижней левой диаграмме.

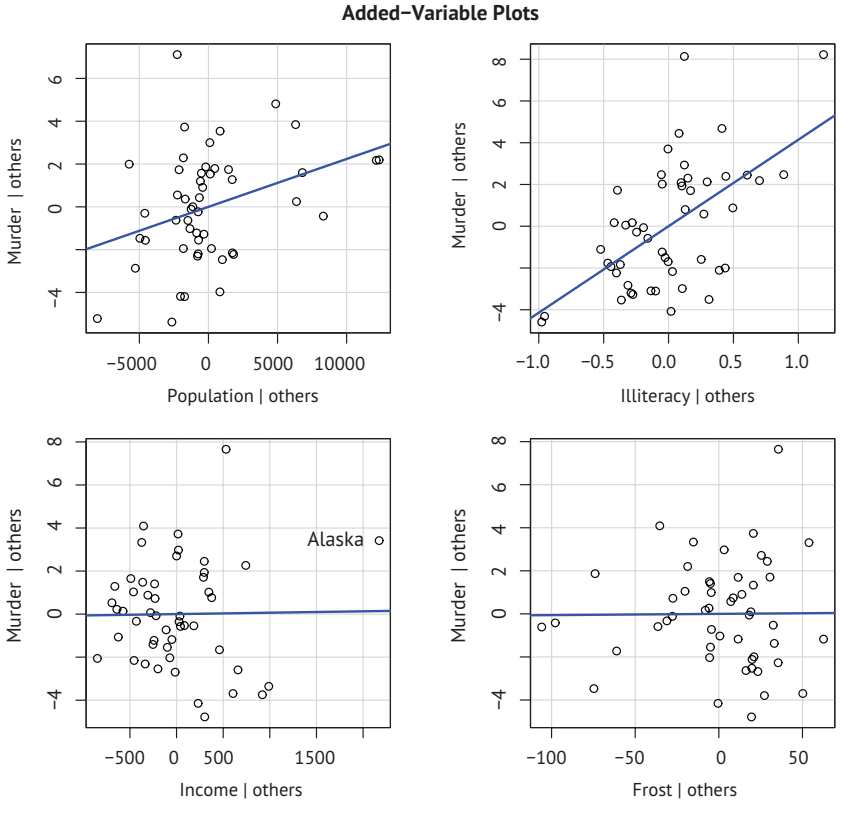


Рис. 8.11. Диаграммы добавленных переменных для исследования вклада влиятельных наблюдений

Прямая на каждой диаграмме – это регрессионный коэффициент при данной независимой переменной. Вклад влиятельных наблюдений можно оценить, если представить, как изменится линия, если удалить точку, соответствующую данному наблюдению. Для примера посмотрите на диаграмму, отражающую зависимость между преступностью и доходами (Murder | Others и Income | Others) снизу слева на рис. 8.11. Как видите, удаление точки, соответствующей штату Аляска, сместило бы линию в область отрицательных значений. На самом деле удаление Аляски приводит к изменению коэффициента регрессии при переменной дохода с положительного (`.00006`) на отрицательный (`-.00085`).

Можно свести информацию о выбросах, точках с высокой напряженностью и влиятельных наблюдениях на одну чрезвычайно информативную диаграмму при помощи функции `influencePlot()` из пакета `car`:

```
library(car)
influencePlot(fit, id="noteworthy", main="Influence Plot",
              sub="Circle size is proportional to Cook's distance")
```

На полученной диаграмме (рис. 8.12) видно, что Невада и Род-Айленд – это выбросы, Калифорния и Гавайи характеризуются высокой напряженностью, а Невада и Аляска – влиятельные наблюдения.

Замена `id="noteworthy"` на `id=list(method="identify")` позволяет интерактивно идентифицировать точки щелчками мыши (для выхода из интерактивного режима нужно нажать клавишу Esc или щелкнуть на кнопке Finish (Стоп)).

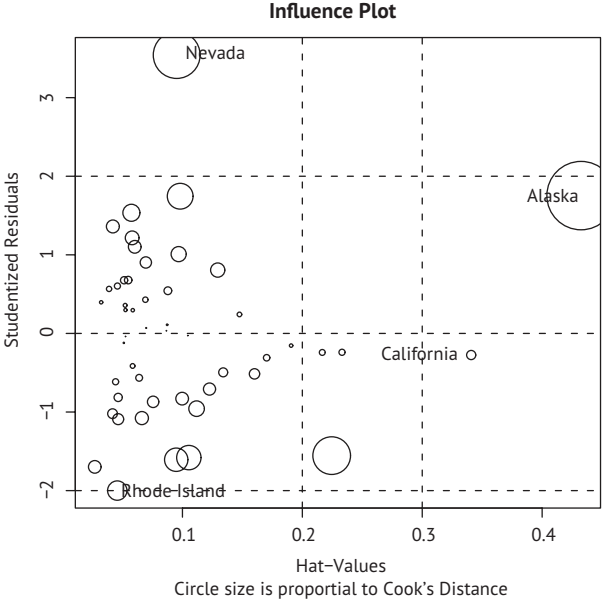


Рис. 8.12. Диаграмма влияния наблюдений на регрессию (influence plot). Штаты выше +2 или ниже –2 по вертикальной оси – выбросы. Штаты выше 0.2 или 0.3 характеризуются высокой напряженностью (необычной комбинацией значений независимых переменных). Размеры кругов пропорциональны степени влияния наблюдения. Наблюдения, обозначенные большими кругами, могут оказывать значительное влияние на параметры модели

8.5. Способы корректировки

Прочитав последние 16 страниц, посвященных диагностике регрессионной модели, вы могли задаться вопросом «Что делать, обнаружив проблему?». Существует четыре метода работы с отклонениями от допущений, лежащих в основе регрессионных моделей:

- удаление наблюдений;
- преобразование переменных;
- добавление или удаление переменных;
- использование регрессии другого вида.

Рассмотрим их по очереди.

8.5.1. Удаление наблюдений

Удаление выбросов часто может улучшить соответствие набора данных требованию нормальности. Влиятельные наблюдения также часто удаляют, потому что они слишком сильно влияют на результаты. После удаления наибольшего выброса или влиятельного наблюдения модель подбирается заново. Если после этого все равно остаются выбросы или влиятельные наблюдения, процесс повторяется, пока не будет достигнуто желаемое соответствие модели данным.

И снова я призываю быть осторожными, принимая решение об удалении наблюдений. Иногда наличие выбросов обусловлено ошибками, допущенными при сборе данных, несоблюдением протокола исследования или нарушением тестируемым инструкции. В таких случаях удаление наблюдений, служащих камнем преткновения, является абсолютно обоснованным.

В других случаях необычные наблюдения могут оказаться самыми интересными объектами в данных. Понимание причин, почему данное наблюдение отличается от остальных, может значительно приблизить вас к разгадке изучаемой проблемы и решению других вопросов, о которых вы могли не задумываться. Некоторые наиболее серьезные достижения могут быть обусловлены интуитивной прозорливостью, когда выявляется что-то, не соответствующее ожиданиям (извините за патетику).

8.5.2. Преобразование переменных

Когда модели не отвечают требованию нормальности, линейности или гомоскедастичности, трансформация одной или более переменных может улучшить или исправить ситуацию. Преобразования обычно заключаются в замене переменной Y на переменную Y^{λ}. Часто используемые значения λ и их интерпретация приведены в табл. 8.5. Если Y – пропорция, то нередко используется логит-преобразование `[log_e(Y/1-Y)]`. Если график распределения Y сильно асимметричен, то обычно применяется лог-трансформация.

Таблица 8.5. Часто используемые трансформации

λ	–2	–1	–0.5	0	0.5	1	2
Преобразование	$1/Y^2$	$1/Y$	$1/\sqrt{Y}$	log(Y)	$\sqrt{Y}$	Отсутствует	Y^2

Если модель не соответствует требованиям нормальности, обычно пытаются преобразовать зависимую переменную. При помощи функции `powerTransform()` из пакета `car` можно оценить по методу максимального правдоподобия величину λ, нормализующую переменную X^{λ}. В листинге 8.8 показан пример применения этого метода к набору данных `states`.

Листинг 8.8. Преобразование Бокса–Кокса (Box-Cox) к нормальному виду

```
> library(car)
> summary(powerTransform(states$Murder))
bcPower Transformation to Normality

              Est.Power Std.Err. Wald Lower Bound Wald Upper Bound
states$Murder       0.6     0.26            0.088               1.1

Likelihood ratio tests about transformation parameters
                     LRT df  pval
LR test, lambda=(0) 5.7  1 0.017
LR test, lambda=(1) 2.1  1 0.145
```

Из полученного результата следует, что переменную `Murder` можно нормализовать, заменив ее на $\texttt{Murder}^{0.6}$. Поскольку `0.6` близко к `0.5`, можно попробовать приблизить переменную к нормальному распределению извлечением квадратного корня. Однако в данном случае гипотеза о $\lambda = 1$ не может быть отвергнута (`p = 0.145`), поэтому необходимость такого преобразования переменной неочевидна. Это согласуется с результатами анализа диагностической диаграммы, представленной на рис. 8.9.

Интерпретация лог-трансформации

Лог-трансформация (замена фактических значений их логарифмами) часто используется для уменьшения асимметричности распределения. Например, переменная `income` (доход) нередко имеет правую асимметрию, когда доход большей части людей сосредоточен в нижней части шкалы, но есть несколько человек, имеющих очень высокие доходы. Как правильно интерпретировать коэффициенты регрессии при переменной, подвергнутой лог-трансформации?

Обычно коэффициент регрессии при X интерпретируется как ожидаемое изменение Y при изменении X на единицу. Рассмотрим модель $Y = 3 + 0{,}6X$. Мы могли бы предсказать увеличение Y на 0,6 при увеличении X на одну единицу. Точно так же изменение X на 10 единиц будет связано с изменением Y на 0,6(10), или на 6 пунктов. Однако если модель является логарифмической $\log_e(Y) = 3 + 0{,}6X$, то изменение X на одну единицу умножает ожидаемое значение Y на $e^{0,6} = 1{,}06$. Таким образом, увеличение X на одну единицу будет предсказывать увеличение Y на 6 %. Увеличение X на 10 единиц умножит ожидаемое значение Y на $e^{0,6(10)} = 1{,}82$. Таким образом, увеличение X на 10 единиц будет предсказывать увеличение Y на 82 %.

Узнать больше об интерпретации логарифмических преобразований в линейной регрессии можно в отличном руководстве Кеннета Бенуа (Kenneth Benoit), доступном по адресу: https://kenbenoit.net/assets/courses/ME104/logmodels2.pdf.

В том случае, когда предположение о линейности не выполняется, обычно помогает преобразование независимых переменных. Для оценки степени, в которую нужно возвести независимые переменные для большего соответствия модели требованию линейности, можно использовать функцию `boxTidwell()` из пакета `car`, реализующую метод наибольшего правдоподобия. Вот пример преобразований Бокса–Тидвелла (Box-Tidwell) для модели, предсказывающей уровень преступности по численности населения и уровню неграмотности:

```
> library(car)
> boxTidwell(Murder~Population+Illiteracy,data=states)

            MLE of lambda Score Statistic (z) Pr(>|z|)
Population        0.86939             -0.3228   0.7468
Illiteracy        1.35812              0.6194   0.5357
```

Из результата следует, что для получения большей линейности стоит попробовать преобразования $\texttt{Population}^{0.87}$ и $\texttt{Illiteracy}^{1.36}$. Однако результаты теста для переменных `Population` (`p = .75`) и `Illiteracy` (`p = .54`) свидетельствуют, что ни одну из них не нужно преобразовывать. Опять же, эти результаты соответствуют диаграмме компонент и остатков (рис. 8.7).

Наконец, преобразование зависимой переменной может помочь в случае гетероскедастичности (непостоянной дисперсии остатков). В листинге 8.8 было показано, что функция `spreadLevelPlot()` из пакета `car` позволяет понять, в какую степень нужно возвести зависимую переменную, чтобы увеличить гомоскедастичность. И снова в примере с набором данных `states` требование постоянства дисперсии ошибок выполняется, и никакие преобразования не требуются.

Предостережение, касающееся преобразований

Среди статистиков ходит старая шутка: если вы не можете доказать A, докажите B и сделайте вид, что это было A (статистикам она кажется довольно смешной). В нашем случае важно, что если вы преобразовали переменные, интерпретация модели должна быть основана на преобразованных переменных, а не на исходных. Если преобразования были осмысленными, такими как логарифм дохода или обратные значения расстояния, то трактовать полученный результат просто. А как интерпретировать взаимосвязь между частотой суицидальных настроений и кубическим корнем из депрессии? Избегайте бессмысленных преобразований.

8.5.3. Добавление или удаление переменных

Изменение числа переменных, входящих в модель, будет влиять на степень ее соответствия данным. Иногда добавление важной пере-

менной может исправить многие из проблем, которые мы обсуждали. Удаление причиняющих беспокойство переменных может привести к аналогичному эффекту.

Удаление переменных – это особенно важное средство в борьбе с мультиколлинеарностью. Если единственная задача – прогнозы, тогда мультиколлинеарность – не проблема. Однако если нужно интерпретировать отдельные независимые переменные, тогда нужно как-то исправить мультиколлинеарность. Наиболее распространенный подход – удалить одну из переменных, из-за которых наблюдается данный феномен (то есть одну из переменных с VIF > 10). Альтернативный подход – использовать гребневую регрессию, разновидность множественной регрессии, разработанную для применения в случаях мультиколлинеарности.

8.5.4. Применение другого подхода

Как только что было показано, один из подходов, применяемых в случае мультиколлинеарности, – это использование другого типа модели (в данном случае гребневой регрессии). При наличии выбросов и/или влиятельных наблюдений можно использовать устойчивую регрессионную модель, а не МНК-регрессию. Если не выполняется предположение о нормальности, то можно подобрать нелинейную регрессионную модель. В случае отклонения от независимости ошибок можно применить модели, которые учитывают структуру остатков, – такие как модели временных рядов или многоуровневые регрессионные модели. Наконец, если не выполняются предположения, лежащие в основе МНК-регрессии, можно обратиться к обобщенным линейным моделям.

Мы рассмотрим некоторые из этих альтернативных подходов в главе 13. Сложно решить, когда нужно стараться подобрать МНК-регрессионную модель, а когда использовать другой подход. Обычно такое решение основано на знании специфики исследуемого явления и выбора подхода, который даст наилучший результат.

Раз мы заговорили о наилучших результатах, обратимся к задаче выбора независимых переменных, включаемых в регрессионную модель.

8.6. Выбор «лучшей» регрессионной модели

Составляя уравнение регрессии, вы в неявном виде сталкиваетесь с выбором из большого числа возможных моделей. Следует ли включить все исследуемые переменные или удалить те, которые не вносят значительного вклада в предсказание значений зависимой переменной? Нужно ли добавлять полиномиальные члены и/или учитывать эффекты взаимовлияния, чтобы улучшить соответствие модели данным? Выбор окончательной регрессионной

модели всегда подразумевает компромисс между точностью предсказания (моделью, которая соответствует данным настолько хорошо, насколько это возможно) и экономностью (простая и легко воспроизводимая модель). При прочих равных условиях из двух моделей с одинаковой предсказательной силой предпочтение обычно отдается более простой. В этом разделе описаны методы выбора между конкурирующими моделями. Слово «лучшей» взято в кавычки, потому что не существует единственного критерия, который можно использовать для выбора. Окончательное решение основывается на мнении исследователя (считайте это гарантией вашей востребованности).

8.6.1. Сравнение моделей

Две вложенные модели можно сравнить по степени соответствия данным при помощи функции `anova()`, входящей в базовую версию R. Вложенная модель (nested model) – это модель, все члены которой входят в другую модель. Мы обнаружили, что в нашей модели множественной регрессии для набора данных `states` коэффициенты при переменных `Income` и `Frost` были незначимыми. Можно проверить, будет ли модель без этих двух переменных предсказывать значения зависимой переменной так же хорошо, как и модель, в которую они включены (листинг 8.9).

Листинг 8.9. Сравнение вложенных моделей при помощи функции anova()

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                          "Illiteracy", "Income", "Frost")])
> fit1 <- lm(Murder ~ Population + Illiteracy + Income + Frost,
          data=states)
> fit2 <- lm(Murder ~ Population + Illiteracy, data=states)
> anova(fit2, fit1)

Analysis of Variance Table

Model 1: Murder ~ Population + Illiteracy
Model 2: Murder ~ Population + Illiteracy + Income + Frost
  Res.Df     RSS  Df     Sum of Sq      F Pr(>F)
1     47 289.246
2     45 289.167  2     0.079 0.0061     0.994
```

В данном случае модель 1 вложена в модель 2. Функция `anova()` одновременно проверяет, занижает или завышает модель предсказанные значения без переменных `Income` и `Frost` по сравнению с полным набором переменных. Поскольку результат проверки незначим (`p = .994`), можно заключить, что эти две переменные не увеличивают предсказательную силу модели, а значит, мы правильно решили исключить их.

Информационный критерий Акаике (Akaike Information Criterion, AIC) – еще один способ сравнения моделей. При расчете этого критерия учитывается статистическое соответствие модели данным и число необходимых для достижения этого соответствия параметров. Предпочтение нужно отдавать моделям с меньшими значениями AIC, указывающими на хорошее соответствие данным при использовании меньшего числа параметров. Этот критерий вычисляется при помощи функции `AIC()` (листинг 8.10).

Листинг 8.10. Сравнение моделей при помощи AIC()

```
> fit1 <- lm(Murder ~ Population + Illiteracy + Income + Frost,
           data=states)
> fit2 <- lm(Murder ~ Population + Illiteracy, data=states)
> AIC(fit1,fit2)

     df      AIC
fit1  6 241.6429
fit2  4 237.6565
```

Значения критерия AIC свидетельствуют, что модель без переменных `Income` и `Frost` лучше. Учтите, что если подход с использованием ANOVA требует вложенных моделей, то для применения AIC это необязательно.

Сравнить две модели относительно просто, но что делать, если имеется четыре, десять или сто возможных моделей, которые нужно проанализировать? Это тема следующего раздела.

8.6.2. Выбор переменных

Существуют два распространенных способа формировать окончательный набор независимых переменных из большего числа имеющихся – это пошаговый метод и регрессия по всем подмножествам.

Пошаговая регрессия

При пошаговом выборе переменные добавляются в модель или удаляются из нее по одной, пока не будет достигнуто заданное значение критерия. Например, при методе *пошагового включения* (forward stepwise) переменные по одной добавляются в модель, пока добавление новых переменных не перестанет ее улучшать. При *пошаговом исключении* (backward stepwise) вы начинаете с модели, включающей все независимые переменные, а потом удаляете их по одной, пока модель не начнет ухудшаться. При *комбинированном методе* (stepwise stepwise) совмещены оба упомянутых подхода. Переменные добавляются по одной, однако на каждом шаге происходит переоценка модели, и те переменные, которые не вносят значительного вклада, удаляются. Независимая переменная может быть включена в модель и удалена из нее несколько раз, пока не будет достигнуто окончательное решение.

Результат применения метода пошаговой регрессии зависит от критериев включения или удаления переменных. При помощи функции `step` из базовой версии R можно провести все три типа пошаговой регрессии с использованием точного критерия AIC. Код в листинге 8.11 применяет метод регрессии с пошаговым исключением для решения задачи множественной регрессии.

Листинг 8.11. Регрессия с пошаговым исключением переменных

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                          "Illiteracy", "Income", "Frost")])

> fit <- lm(Murder ~ Population + Illiteracy + Income + Frost,
          data=states)
> step(fit, direction="backward")

Start:  AIC=97.75
Murder ~ Population + Illiteracy + Income + Frost

             Df Sum of Sq    RSS     AIC
- Frost       1     0.021 289.19  95.753
- Income      1     0.057 289.22  95.759
<none>                    289.17  97.749
- Population  1    39.238 328.41 102.111
- Illiteracy  1   144.264 433.43 115.986

Step:  AIC=95.75
Murder ~ Population + Illiteracy + Income

             Df Sum of Sq    RSS     AIC
- Income      1     0.057 289.25  93.763
<none>                    289.19  95.753
- Population  1    43.658 332.85 100.783
- Illiteracy  1   236.196 525.38 123.605

Step:  AIC=93.76
Murder ~ Population + Illiteracy

             Df Sum of Sq    RSS     AIC
<none>                    289.25  93.763
- Population  1    48.517 337.76  99.516
- Illiteracy  1   299.646 588.89 127.311

Call:
lm(formula = Murder ~ Population + Illiteracy, data = states)

Coefficients:
(Intercept)   Population   Illiteracy
  1.6515497    0.0002242    4.0807366
```

Мы начинаем с модели, включающей все четыре независимые переменные. В столбце `AIC` выводится значение одноименного кри-

терия для модели, из которой удалена указанная в соответствующей строке переменная. Значение AIC для строки <none> (никакой) – это значение критерия для модели, из которой не удалено никаких переменных. На первом шаге удалена переменная Frost, что привело к уменьшению AIC с 97.75 до 95.75. На втором шаге удалена переменная Income, при этом значение AIC снизилось до 93.76. Удаление остальных переменных увеличивает значение критерия, поэтому процесс остановлен.

Пошаговая регрессия – спорный подход. С его помощью можно найти хорошую модель, однако нет гарантии, что она будет лучшей, поскольку рассмотрены не все возможные модели. Обойти это ограничение позволяет применение регрессии по всем подмножествам.

Регрессия по всем подмножествам

В подходе регрессии по всем подмножествам исследуются все возможные модели. Вы можете просмотреть все полученные результаты или вывести на экран только заданное число лучших моделей для каждого подмножества (одна независимая переменная, две и т. д.). Например, при значении параметра nbest=2 выводятся на экран две лучшие модели для одной независимой переменной, потом две лучшие модели для двух независимых переменных, затем – для трех, и, наконец, две лучшие модели со всеми независимыми переменными.

Регрессия по всем подмножествам проводится при помощи функции regsubsets() из пакета leaps. В качестве критерия «лучшей» модели можно выбрать коэффициент R-квадрат, скорректированный коэффициент R-квадрат или Cp-статистику Мэллоуса (Mallows Cp statistic).

Как вы уже знаете, коэффициент R-квадрат (коэффициент детерминации) – это доля дисперсии зависимой переменной, объясняемая независимыми переменными. Скорректированный коэффициент R-квадрат учитывает число параметров модели. Дело в том, что коэффициент R-квадрат всегда увеличивается при добавлении независимых переменных. Когда число независимых переменных достаточно велико (по сравнению с объемом выборки), соответствие модели данным может быть переоценено. Скорректированный коэффициент R-квадрат создан с целью дать более устойчивую оценку коэффициента детерминации для генеральной совокупности. Статистика Мэллоуса тоже часто используется в качестве критерия «лучшей» модели. Считается, что для хорошей модели эта статистика должна принимать значения, близкие к числу параметров модели (включая свободный член).

В листинге 8.12 показано применение регрессии по всем подмножествам к набору данных states. Результаты можно отобразить

графически при помощи функции `plot()` из пакета `leaps`, но я обнаружил, что многие испытывают затруднения при интерпретации этой диаграммы. В листинге 8.12 те же результаты представлены в виде таблицы, которую, как мне кажется, проще понять.

Листинг 8.12. Регрессия по всем подмножествам

```
library(leaps)
states <- as.data.frame(state.x77[,c("Murder", "Population",
                        "Illiteracy", "Income", "Frost")])

leaps <-regsubsets(Murder ~ Population + Illiteracy + Income +
                   Frost, data=states, nbest=4)

subsTable <- function(obj, scale){
  x <- summary(leaps)
  m <- cbind(round(x[[scale]],3), x$which[,-1])
  colnames(m)[1] <- scale
  m[order(m[,1]), ]
}

subsTable(leaps, scale="adjr2)

  adjr2 Population Illiteracy Income Frost
1 0.033          0          0      1     0
1 0.100          1          0      0     0
1 0.276          0          0      0     1
2 0.292          1          0      0     1
3 0.309          1          0      1     1
3 0.476          0          1      1     1
2 0.480          0          1      1     0
2 0.481          0          1      0     1
1 0.484          0          1      0     0
4 0.528          1          1      1     1
3 0.539          1          1      1     0
3 0.539          1          1      0     1
2 0.548          1          1      0     0
```

Каждая строка таблицы представляет модель. Первый столбец сообщает количество независимых переменных в модели. Второй – масштаб (в данном случае скорректированный коэффициент R-квадрат), используемый для описания соответствия каждой модели, причем строки отсортированы по этому масштабу. (Примечание: вместо `adjr2` можно использовать другие значения шкалы. Список доступных параметров вы найдете в справке `?regsubsets`.) Единицы и нули в строках сообщают, какие переменные включены (`1`) или исключены (`0`) из модели.

Например, модель, основанная на единственной независимой переменной `Income`, имеет скорректированный R-квадрат, равный `0.033`. Модель с независимыми переменными `Population`, `Illiteracy`

и `Income` имеет скорректированный R-квадрат `0.539`. Напротив, модель, использующая только независимые переменные `Population` и `Illiteracy`, имеет скорректированный R-квадрат `0.548`. Здесь ясно видно, что модель с меньшим количеством независимых переменных на самом деле имеет больший скорректированный коэффициент R-квадрат (чего не может произойти с нескорректированным коэффициентом R-квадрат). Из таблицы также видно, что модель с двумя независимыми переменными (`Population` и `Illiteracy`) является наилучшей.

В большинстве случаев регрессия по всем подмножествам предпочтительнее пошаговой регрессии, потому что анализирует больше моделей. Однако если независимых переменных много, то вычисления могут занять много времени. В целом автоматизированный выбор переменных нужно рассматривать как помощь, а не определяющий фактор при выборе моделей. В конечном счете вы должны опираться на знание исследуемого предмета.

8.7. Продолжение анализа

Закончим обсуждение регрессии знакомством с методами оценки применимости модели для генеральной совокупности и анализа относительной важности независимых переменных.

8.7.1. Перекрестная проверка

В предыдущем разделе мы изучали методы выбора переменных, входящих в уравнение регрессии. Если главная задача – описание, то работа заканчивается выбором регрессионной модели и ее интерпретацией. Однако если цель – предсказание, то есть все основания спросить: «Насколько хорошо полученное уравнение работает в реальном мире?»

По определению, регрессионные методы позволяют рассчитать оптимальные для имеющегося набора данных параметры. При использовании МНК-регрессии параметры модели подбираются так, чтобы минимизировать сумму квадратов ошибок предсказаний (остатков) и, напротив, максимизировать долю объясняемой дисперсии зависимой переменной (коэффициент детерминации). Поскольку уравнение оптимизировано для имеющегося набора данных, оно не всегда даст такие же хорошие результаты для других данных.

Мы начали эту главу с рассказа о тренере по фитнесу, который хотел предсказать число затрачиваемых человеком калорий по продолжительности выполнения им упражнений, его возрасту, полу и индексу массы тела. Если подобрать для этих данных уравнение МНК-регрессии, то получатся значения параметров модели, максимизирующие коэффициент детерминации этого конкретного набора данных. Однако наш тренер хочет использовать это уравнение для

предсказания числа затраченных калорий вообще, а не только для тех, кто участвовал в исследовании. Понятно, что уравнение не будет работать так же хорошо для нового набора наблюдений, но какая доля точности будет потеряна? Перекрестная проверка – полезный метод оценки применимости модели к генеральной совокупности.

При перекрестной проверке часть данных используется как обучающая выборка, а часть – как контрольная. Уравнение регрессии подгоняется для обучающей выборки, а затем применяется к контрольной. Поскольку контрольная выборка не использовалась для подбора параметров модели, то точность предсказаний модели на этой выборке может послужить хорошей оценкой применимости полученных параметров модели к новым данным.

При k-кратной перекрестной проверке выборка делится на k подвыборок. Каждая из них играет роль контрольной выборки, а объединенные данные из оставшихся $k-1$ подвыборок используются как обучающая группа. Результат применения k уравнений к k контрольным выборкам фиксируется и усредняется. Если $k = n$ (общему числу наблюдений), то такой подход называется оценкой по методу *складного ножа* (последовательного исключения значений выборки с возвратом – *jackknifing*).

Выполнить k-кратную перекрестную проверку можно при помощи функции `crossval()` из пакета `bootstrap`. Код в листинге 8.13 определяет функцию `shrinkage()` для k-кратной перекрестной проверки коэффициента детерминации.

Листинг 8.13. Функция для k-кратной перекрестной проверки

```
shrinkage <- function(fit, k=10, seed=1){
  require(bootstrap)

  theta.fit <- function(x,y){lsfit(x,y)}
  theta.predict <- function(fit,x){cbind(1,x)%*%fit$coef}

  x <- fit$model[,2:ncol(fit$model)]
  y <- fit$model[,1]

  set.seed(seed)
  results <- crossval(x, y, theta.fit, theta.predict, ngroup=k)
  r2    <- cor(y, fit$fitted.values)^2
  r2cv  <- cor(y, results$cv.fit)^2
  cat("Original R-square =", r2, "\n")
  cat(k, "Fold Cross-Validated R-square =", r2cv, "\n")
}
```

С помощью этой функции можно вычислить коэффициент детерминации и стандартную ошибку остатков и провести анализ методом перекрестной проверки (подробнее о методах бутстреп-анализа рассказывается в главе 12).

Ниже показан пример применения функции `shrinkage()` для 10-кратной перекрестной проверки с использованием всех четырех независимых переменных из набора данных `states`:

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
          "Illiteracy", "Income", "Frost")])
> fit <- lm(Murder ~ Population + Income + Illiteracy + Frost, data=states)
> shrinkage(fit)

Original R-square = 0.567
10 Fold Cross-Validated R-square = 0.356
```

Как видите, коэффициент детерминации, вычисленный для нашей выборки (`0.567`), чересчур оптимистичен. Лучшая оценка доли изменчивости уровня преступности, объясняемой нашей моделью на новых данных, – это коэффициент детерминации, полученный методом перекрестной проверки (`0.356`). Учтите, что наблюдения разделяются на *k* групп случайно, поэтому при каждом использовании функции `shrinkage()` могут получаться немного отличающиеся результаты.

Перекрестную проверку можно использовать для выбора переменных, отдавая предпочтение моделям, лучше соответствующим генеральной совокупности. Например, модель с двумя независимыми переменными (`Population` и `Illiteracy`) характеризуется меньшим снижением коэффициента детерминации при перекрестной проверке (`0.03`), по сравнению с `0.12` для модели, включающей все переменные:

```
> fit2 <- lm(Murder ~ Population + Illiteracy,data=states)
> shrinkage(fit2)

Original R-square = 0.567
10 Fold Cross-Validated R-square = 0.515
```

Это обстоятельство может послужить аргументом в пользу модели с двумя переменными.

При прочих равных условиях уравнение регрессии, полученное на основании большей обучающей выборки и лучше соответствующее генеральной совокупности, при перекрестной проверке будет оценено выше. В этом случае коэффициент детерминации будет уменьшаться не так сильно, а предсказания получатся более точными.

8.7.2. Относительная важность

До этого момента мы задавались вопросом: «Какие переменные полезны для предсказания результата?» Однако иногда больше интересует ответ на вопрос: «Какие переменные наиболее важны для предсказания результата?» В таких случаях было бы желательно ранжировать переменные согласно их относительной важности.

Для этого второго вопроса могут существовать практические основания. Например, имея возможность ранжировать лидерские качества по их важности для организационного успеха, можно ориентировать управленцев на более продуктивное поведение.

Если бы независимые переменные не коррелировали друг с другом, эта задача была бы простой. Можно было бы просто упорядочить независимые переменные по степени их корреляции с зависимой переменной. Однако в большинстве случаев независимые переменные коррелируют друг с другом, и это значительно усложняет дело.

Было предпринято множество попыток разработать способы оценки относительной важности независимых переменных. Самый простой – сравнить стандартизированные коэффициенты регрессии. Они характеризуют ожидаемое изменение зависимой переменной (выраженное в числе стандартных отклонений) при изменении отдельной независимой переменной на одно стандартное отклонение при постоянных значениях прочих независимых переменных. В R стандартизированные регрессионные коэффициенты можно получить, предварительно преобразовав все переменные при помощи функции `scale()` так, чтобы их среднее было равно 0, а стандартное отклонение – 1. Но имейте в виду, что функция `scale()` возвращает матрицу, а функция `lm()` принимает таблицы данных, поэтому необходимо выполнить промежуточный шаг, изменяющий формат данных. Программный код и результаты его применения к нашей задаче множественной регрессии приведены ниже:

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
                          "Illiteracy", "Income", "Frost")])
> zstates <- as.data.frame(scale(states))
> zfit <- lm(Murder~Population + Income + Illiteracy + Frost, data=zstates)
> coef(zfit)

(Intercept)  Population      Income  Illiteracy       Frost
 -9.406e-17   2.705e-01   1.072e-02   6.840e-01   8.185e-03
```

Здесь видно, что увеличение уровня неграмотности (`Illiteracy`) на одно стандартное отклонение влечет увеличение уровня преступности (`Murder`) на `0.68` стандартного отклонения при постоянных значениях численности населения (`Population`), дохода (`Income`) и морозности (`Frost`). Исходя из стандартизированных регрессионных коэффициентов, неграмотность – наиболее важный параметр, а морозность – наименее важный.

Было предпринято много других попыток количественной оценки относительной важности независимых переменных. Относительную важность можно выражать в виде вклада каждой переменной в коэффициент детерминации (как поодиночке, так и в сочетании с другими независимыми переменными). Несколько возможных подходов к оценке относительной важности реали-

зованы в пакете `relaimpo`, созданном Ульрике Грёмпингом (Ulrike Grömping) (http://mng.bz/KDYF).

Новый метод, названный *относительным взвешиванием*, выглядит довольно многообещающим. Этот метод довольно точно аппроксимирует среднее увеличение коэффициента детерминации при добавлении независимой переменной во все возможные подмодели (Johnson, 2004; Johnson, Lebreton, 2004; LeBreton, Tonidandel, 2008). Функция для вычисления относительных весов представлена в листинге 8.14.

Листинг 8.14. Функция relweights() для вычисления относительной важности переменных

```
relweights <- function(fit,...){
  R <- cor(fit$model)
  nvar <- ncol(R)
  rxx <- R[2:nvar, 2:nvar]
  rxy <- R[2:nvar, 1]
  svd <- eigen(rxx)
  evec <- svd$vectors
  ev <- svd$values
  delta <- diag(sqrt(ev))
  lambda <- evec %*% delta %*% t(evec)
  lambdasq <- lambda ^ 2
  beta <- solve(lambda) %*% rxy
  rsquare <- colSums(beta ^ 2)
  rawwgt <- lambdasq %*% beta ^ 2
  import <- (rawwgt / rsquare) * 100
  import <- as.data.frame(import)
  row.names(import) <- names(fit$model[2:nvar])
  names(import) <- "Weights"
  import <- import[order(import),1, drop=FALSE]
  dotchart(import$Weights, labels=row.names(import),
     xlab="% of R-Square", pch=19,
     main="Relative Importance of Predictor Variables",
     sub=paste("Total R-Square=", round(rsquare, digits=3)),
     ...)
return(import)
}
```

ПРИМЕЧАНИЕ. Код в листинге 8.14 – это адаптированная программа SPSS, любезно предоставленная др. Джонсоном (Dr. Johnson). В своей статье (Johnson, 2000, Multivariate Behavioral Research, 35, 1–19) он объясняет, как вычисляются относительные веса.

В листинге 8.15 показано применение функции `relweights()` для предсказания уровня преступности по численности населения, уровню неграмотности, дохода и морозности в наборе данных `states`.

Листинг 8.15. Применение функции relweights()

```
> states <- as.data.frame(state.x77[,c("Murder", "Population",
          "Illiteracy", "Income", "Frost")])
> fit <- lm(Murder ~ Population + Illiteracy + Income + Frost, data=states)
> relweights(fit, col="blue")

           Weights
Income        5.49
Population   14.72
Frost        20.79
Illiteracy   59.00
```

На диаграмме с результатами (рис. 8.13) видно, что общая доля объясняемой моделью дисперсии (коэффициент детерминации, `R-Square = 0.567`) распределяется между независимыми переменными. На долю неграмотности приходится 59 % коэффициента детерминации, на долю морозности – 20.79 % и т. д. Если применить метод относительного взвешивания, то неграмотность имеет наибольшую относительную важность, а дальше следуют (в порядке убывания важности) морозность, численность населения и уровень дохода.

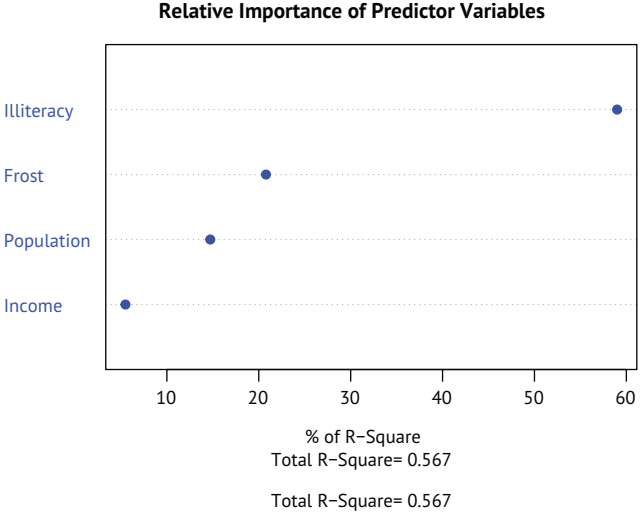


Рис. 8.13. Точечная диаграмма относительных весов для задачи множественной регрессии на наборе данных states. Наибольшие веса соответствуют более важным независимым переменным. Например, на неграмотность приходится 59 % общей объясняемой дисперсии (0,567), тогда как на доход приходится только 5,49 %. Соответственно, неграмотность имеет наибольшую относительную важность, чем доход

Меры относительной важности переменных (и в особенности относительные веса) широко используются на практике. Они находятся гораздо ближе к нашему интуитивному пониманию относительной важности по сравнению со стандартизированными

коэффициентами регрессии. Я уверен, что в ближайшие годы интенсивность использования показателей относительной важности резко возрастет.

Итоги

- Регрессионный анализ – сложный многоэтапный интерактивный процесс, включающий подбор моделей, оценку их соответствия статистическим предположениям, модификацию данных и моделей, а также перенастройку для получения окончательного результата.
- Диагностика регрессионных моделей проводится с целью оценить соответствие данных статистическим предположениям и выбрать методы модификации модели или данных для более точного соответствия этим предположениям.
- Доступно множество самых разных методов выбора переменных для включения в окончательную регрессионную модель, в том числе критерии значимости, статистики оценки соответствия и автоматизации принятия решений, таких как пошаговая регрессия и регрессия всех подмножеств.
- Для оценки вероятной эффективности прогностической модели на новых выборках данных можно использовать перекрестную проверку.
- Метод относительного взвешивания позволяет справиться с решением сложной задачи оценки важности переменных и определить, какие переменные являются наиболее важными для прогнозирования результата.

9 Дисперсионный анализ

В этой главе:

- использование R для моделирования основных типов экспериментов;
- подбор и интерпретация ANOVA-моделей;
- оценка выполнимости допущений, лежащих в основе модели.

В главе 8 мы рассмотрели регрессионные модели для предсказания значений количественной зависимой переменной по значениям количественных независимых переменных. Однако нет никаких причин, по которым в качестве независимых переменных нельзя было бы использовать номинальные или порядковые факторы. Если в набор независимых переменных входят факторы, то акцент обычно смещается с предсказания на исследование межгрупповых различий, и такой метод называется *дисперсионным анализом* (analysis of variance, ANOVA). Этот метод используется для анализа самых разных экспериментов и квазиэкспериментов. В данной главе представлен обзор функций, имеющихся в R, которые можно использовать для анализа результатов исследований, проведенных по стандартным планам.

Для начала познакомимся с терминологией и обсудим общие принципы подгонки ANOVA-моделей в R. Затем рассмотрим не-

сколько примеров, иллюстрирующих анализ распространенных типов экспериментов. По ходу изложения мы будем иметь дело с состояниями тревоги и пониженным уровнем холестерина в крови, а также поможем беременным мышам обзавестись упитанным потомством, убедимся, что у свиней вырастают достаточно длинные зубы, облегчим дыхание растениям и узнаем, каких полок в бакалейных магазинах стоит избегать.

Помимо базовой версии R, мы будем использовать пакеты `car`, `rrcov`, `multcomp`, `effects`, `MASS`, `dplyr`, `ggplot2` и `mvoutlier`. Не забудьте установить их перед опробованием примеров программного кода.

9.1. Краткий обзор терминологии

Планирование экспериментов в целом и дисперсионный анализ в частности имеют свою собственную терминологию. Перед тем как обсуждать анализ по-разному построенных экспериментов, кратко перечислим некоторые важные термины и рассмотрим наиболее значимые концепции на примере ряда усложняющихся экспериментов.

Допустим, вы интересуетесь лечением тревожных состояний. Два широко распространенных метода лечения таких состояний – это когнитивная трудотерапия (КТ) и десенсибилизация и переработка движением глаз (ДПДГ). Вы выбрали десять человек, страдающих повышенной тревожностью, и случайным образом распределяете их по двум равным группам. В течение пяти недель одна группа подвергается КТ, а другая – ДПДГ. По окончании лечения каждого пациента вы предлагаете каждому клиенту заполнить специальную анкету самооценки уровня тревожности. План эксперимента представлен в табл. 9.1.

Таблица 9.1. Однофакторный дисперсионный анализ для независимых переменных

Способ лечения	
КТ	ДПДГ
п1	п6
п2	п7
п3	п8
п4	п9
п5	п10

При такой организации эксперимента способ лечения – это *межгрупповой* фактор с двумя уровнями (КТ и ДПДГ). Этот фактор называется межгрупповым, потому что каждый пациент может быть отнесен к одной и только одной группе. Ни один из пациентов не

подвергался действию и КТ, и ДПДГ. Буква «п» в таблице означает «пациент». Результат анкетирования – это *зависимая переменная*, а тип лечения – *независимая*. Поскольку число наблюдений для каждого способа лечения равно, мы имеем дело со *сбалансированным планом*. Когда размер выборок для разных типов воздействия не одинаков, говорят, что эксперимент имеет *несбалансированный план*.

План, представленный в табл. 9.1, называется *однофакторным дисперсионным анализом*, потому что имеется только одна классифицирующая переменная. Конкретнее в данном случае мы имеем дело с однофакторным дисперсионным анализом для независимых переменных. Эффекты в разных планах сначала оцениваются при помощи F-критерия. Если такой критерий для способа лечения дал значимый результат, можно считать, что средние баллы тревожности после пяти недель лечения разными способами действительно различаются.

Если бы вы заинтересовались результатом применения КТ по прошествии времени, то могли бы поместить всех пациентов в одну группу с КТ и опросить их по окончании терапии и еще через шесть месяцев. План этого эксперимента показан в табл. 9.2.

Таблица 9.2. Однофакторный дисперсионный анализ для зависимых переменных

Пациент	Время	
	5 недель	6 месяцев
п1		
п2		
п3		
п4		
п5		
п6		
п7		
п8		
п9		
п10		

Время – это *внутригрупповой фактор* с двумя уровнями (пять недель, шесть месяцев). Этот фактор называется внутригрупповым, потому что каждый пациент исследуется при обоих значениях этого фактора. Соответствующая статистическая процедура называется *однофакторным дисперсионным анализом для зависимых наблюдений*. Поскольку каждый субъект исследования оценивается более

чем один раз, такой план еще называют *дисперсионным анализом повторных измерений*. Если F-критерий для переменной «время» дал значимый результат, то это значит, что средний уровень тревожности, отмеченный у пациентов через пять недель и шесть месяцев, действительно различается.

Если бы вы интересовались различиями между способами лечения и изменениями результатов со временем, то могли бы объединить описанные выше подходы, случайно распределив пациентов по двум равным группам (с применением КТ и ДПДГ) и проанализировав их уровень тревожности по завершении терапии (пять недель) и через шесть месяцев (табл. 9.3).

Таблица 9.3. Двухфакторный дисперсионный анализ с одной парой зависимых и одной парой независимых переменных

		Пациент	Время	
			5 недель	6 месяцев
Способ лечения	КТ	п1 п2 п3 п4 п5		
	ДПДГ	п6 п7 п8 п9 п10		

Используя и время, и способ лечения как факторы, можно изучить влияние способа лечения (усредненного по времени), времени (усредненного по способам лечения) и комбинацию способа лечения и времени. Первые два подхода называются *главными эффектами*, а комбинация (что неудивительно) называется *эффектом комбинирования*.

Такое исследование сочетания нескольких факторов называется *многофакторным планом дисперсионного анализа*. Сочетание двух факторов – это двухфакторный дисперсионный анализ, сочетание трех факторов – трехфакторный и т. д. Когда многофакторный анализ включает и межгрупповые, и внутригрупповые факторы, он также называется *смешанной моделью дисперсионного анализа* (mixed-model ANOVA). Описанный выше эксперимент – это двухфакторный дисперсионный анализ со смешанными эффектами (уф!).

В этом случае вычисляются три F-критерия: один – для способа лечения, один – для временного интервала и один – для оценки взаимовлияния этих двух переменных. Если критерий для способа лечения статистически значим, то это означает, что КТ и ДПДГ различаются по их влиянию на уровень тревожности. Если ста-

тистически значим критерий для временно́го интервала, то это свидетельствует о том, что уровень тревожности после окончания лечения и через шесть месяцев различается. Если статистически значим критерий для взаимовлияния времени и способа лечения, то это значит, что два способа терапии имеют разные временны́е эффекты (т. е. уровень тревожности по-разному изменяется с течением времени для этих двух способов лечения).

Теперь мы немного расширим план эксперимента. Известно, что депрессия влияет на ход лечения, а также что депрессия и тревожность часто сопутствуют друг другу. Пациенты распределялись по группам случайно, но вполне возможно, что пациенты, получавшие разную терапию, изначально различались уровнем депрессии. Любые различия между этими группами могут объясняться исходными различиями в уровне депрессивности пациентов, а не экспериментальным воздействием. Поскольку уровень депрессии также может объяснять межгрупповые различия зависимой переменной, его называют *смешивающим фактором* (confounding factor). А поскольку вы не интересуетесь депрессией, это *мешающая переменная* (nuisance variable).

Если бы вы регистрировали уровень депрессивности при помощи анкеты для самооценки, то могли бы учесть межгрупповые различия, которые имеют место из-за разницы в уровне депрессии, перед оценкой различий, связанных со способом лечения. В таком случае уровень депрессивности назывался бы *ковариатой* (covariate), а план исследования носил бы название *ковариационный анализ* (analysis of covariance, ANCOVA).

Наконец, в этом исследовании вы изучали одну зависимую переменную (уровень тревожности). Можно увеличить ценность работы, добавив дополнительные оценки уровня тревожности (по мнению семьи, по отзывам терапевта, показатель влияния тревожности на повседневную жизнь). При наличии более чем одной зависимой переменной такой план называется *многомерным дисперсионным анализом* (multivariate analysis of variance, MANOVA). Если в анализ входят ковариаты, то он будет называться *многомерным ковариационным анализом* (multivariate analysis of covariance, MANCOVA).

Теперь, вооружившись новой терминологией, вы готовы удивлять друзей, изумлять новых знакомых и обсуждать подгонку ANOVA/ANCOVA/MANOVA-модели в R.

9.2. Подгонка ANOVA-моделей

Методологии дисперсионного (ANOVA) и регрессионного анализов развивались по отдельности, но с функциональной точки зрения это два частных случая общей линейной модели. Модели ANOVA можно анализировать при помощи той же функции `lm()`,

которая использовалась для регрессионного анализа в главе 8. Однако в этой главе мы сначала обсудим функцию `aov()`. Результаты применения этих функций равнозначны, но `aov()` представляет результаты в виде, более привычном для людей, работающих с ANOVA. Для полноты изложения я приведу пример использования функции `lm()` в конце этой главы.

9.2.1. Функция aov()

Функция `aov()` имеет следующий синтаксис:

```
aov(формула, data=таблица_данных)
```

В табл. 9.4 перечислены специальные символы, которые можно использовать в формулах. В этой таблице зависимая переменная обозначена как `y`, а буквы `A`, `B` и `C` соответствуют факторам.

Таблица 9.4. Специальные символы, используемые в формулах R

Символ	Описание
`~`	Отделяет зависимые переменные (слева) от независимых (справа). Например, предсказание значений `y` по значениям `A`, `B` и `C` можно представить так: `y ~ A + B + C`
`+`	Разделяет независимые переменные
`:`	Обозначает взаимосвязь (коллинеарность) между независимыми переменными. Предсказание значений `y` по значениям `A` и `B` с учетом взаимосвязи между `x` и `z` можно описать как `y ~ x + z + x:z`
`*`	Кратко обозначает все возможные взаимосвязи. Код `y ~ A * B * C` будет развернут в `y ~ A + B + C + A:B + A:C + B:C + A:B:C`
`^`	Обозначает взаимосвязь до определенного порядка. Код `y ~ (A+B+C)^2` будет развернут в `y ~ A + B + C + A:B + A:C + A:B`
`.`	Символ-заполнитель для всех переменных в таблице данных, кроме зависимой. Например, если таблица данных содержит переменные `y`, `A`, `B` и `C`, то код `y ~ .` будет означать `y ~ A + B + C`

В табл. 9.5 показаны формулы для некоторых наиболее часто используемых видов экспериментов. В этой таблице строчными буквами обозначаются количественные переменные, прописными – группирующие факторы, а `Объект` – это переменная, содержащая уникальный идентификатор объекта.

Таблица 9.5. Формулы для распространенных видов экспериментов

Вид эксперимента	Формула
Однофакторный дисперсионный анализ	`y ~ A`
Однофакторный ковариационный анализ с одной ковариатой	`y ~ x + A`
Двухфакторный дисперсионный анализ	`y ~ A * B`

Окончание табл. 9.5

Вид эксперимента	Формула
Двухфакторный ковариационный анализ с двумя ковариатами	`y ~ x1 + x2 + A * B`
Случайный блоковый	`y ~ B + A` (где B – определяющий блоки фактор)
Однофакторный дисперсионный анализ для зависимых переменных	`y ~ A + Error(Объект/A)`
Дисперсионный анализ с повторными изменениями с одним внутригрупповым фактором (W) и одним межгрупповым фактором (B)	`y ~ B * W + Error(Объект/W)`

Ниже в этой главе мы подробно рассмотрим примеры некоторых из этих видов экспериментов.

9.2.2. Порядок членов в формуле

Порядок членов в формуле важен, если (а) имеется больше одного фактора и план не сбалансирован или (б) есть ковариаты. Если одно из этих двух условий выполняется, то переменные в уравнении справа будут коррелироваться друг с другом. В таком случае не существует однозначного способа разделить их влияние на зависимую переменную. Например, в двухфакторном дисперсионном анализе с неравным числом наблюдений для разных комбинаций факторов модель `y ~ A*B` даст *иной* результат, отличный от результата модели `y ~ B*A`.

По умолчанию в R применяется последовательный подход (I типа) для вычисления эффектов ANOVA (см. врезку «Упорядочите вычисления!»). Первую модель можно записать как `y ~ A + B + A:B`. Итоговая таблица дисперсионного анализа в R будет оценивать:

- влияние `A` на `y`;
- влияние `B` на `y` при постоянных значениях `A`;
- влияние комбинации `A` и `B` при постоянных значениях главных эффектов `A` и `B`.

Упорядочите вычисления!

Если независимые переменные коррелируют друг с другом или имеются ковариаты, то не существует однозначного метода оценки влияния каждой из этих переменных в отдельности на зависимую переменную. Рассмотрим несбалансированный двухфакторный дисперсионный анализ с факторами `A` и `B` и зависимой переменной `y`. У этого эксперимента три эффекта – главные эффекты `A` и `B` и эффект их комбинации. Представим, что данные моделируются при помощи формулы `Y ~ A + B + A:B`, в таком случае существует три основных подхода для распределения дисперсии `y` между эффектами, перечисленными в правой части уравнения.

Тип I (последовательный)

Последующие эффекты в формуле масштабируются по предшествующим. Эффект A не масштабируется, B масштабируется по A. Комбинация A:B масштабируется по A и B.

Тип II (иерархический)

Эффекты масштабируются по другим эффектам того же или более низкого уровня. A масштабируется по B, B масштабируется по A. Комбинация A:B масштабируется и по A, и по B.

Тип III (краевой)

Каждый эффект масштабируется по всем остальным эффектам в модели. A масштабируется по B и A:B. B масштабируется по A и A:B. Комбинация A:B масштабируется по A и B. В R по умолчанию реализуется тип I. В других программах, таких как SAS и SPSS, по умолчанию реализуется тип III.

Чем больше разница в размерах выборок, тем сильнее влияет порядок членов уравнения на результаты. В общем случае более важные эффекты должны стоять в формуле раньше. В частности, сначала нужно указывать ковариаты, затем главные эффекты, потом парные комбинации, далее – комбинации трех переменных и т. д. Из главных эффектов первыми следует указывать наиболее существенные. Вот главная идея: если план исследования не ортогональный (то есть факторы и/или ковариаты коррелированы), то будьте аккуратны при определении порядка эффектов.

Перед тем как перейти к частным примерам, отмечу, что функция `Anova()` из пакета `car` (не путайте с обычной функцией `anova()`) поддерживает подходы типов II и III, но не I. Подход типа I реализован в функции `aov()`. Функцию `Anova()` предпочтительнее использовать, когда желательно, чтобы ваши результаты соответствовали результатам, полученным с помощью других статистических программ, таких как SAS или SPSS. За подробностями обращайтесь к справке `help(Anova, package="car")`.

9.3. Однофакторный дисперсионный анализ

Однофакторный дисперсионный анализ позволяет сравнить средние значения зависимой переменной для двух и более групп, заданных категориальным группирующим фактором. Описываемый далее пример основан на наборе данных `cholesterol` в пакете `multcomp`, заимствованном из публикации Westfall, Tobia, Rom, & Hochberg (1999). Пятьдесят пациентов проходили один из пяти курсов лечения по снижению уровня холестерина (переменная `trt`). Три из этих курсов заключались в использовании одного и того же препарата в количестве 20 мг один раз в день (`1time`), 10 мг дважды в день (`2times`) или 5 мг четыре раза в день (`4times`). Два других курса лечения заключались

в приеме альтернативных препаратов (drugD и drugE). Какой из курсов лечения привел к наиболее заметному снижению уровня холестерина (переменная response)? Проведенный анализ представлен ниже.

Листинг 9.1. Однофакторный дисперсионный анализ

```
> library(dplyr)
> data(cholesterol, package="multcomp")
> plotdata <- cholesterol %>%                                    (1)
    group_by(trt) %>%                                            (1)
    summarize(n = n(),                                           (1)
              mean = mean(response),                             (1)
              sd = sd(response),                                 (1)
              ci = qt(0.975, df = n - 1) * sd / sqrt(n))         (1)
> plotdata

  trt        n  mean    sd    ci
  <fct>  <int> <dbl> <dbl> <dbl>
1 1time     10  5.78  2.88  2.06
2 2times    10  9.22  3.48  2.49
3 4times    10 12.4   2.92  2.09
4 drugD     10 15.4   3.45  2.47
5 drugE     10 20.9   3.35  2.39

> fit <- aov(response ~ trt, data=cholesterol)                   (2)

> summary(fit)

            Df Sum Sq   Mean Sq    F value           Pr(>F)
trt          4   1351       338       32.4     9.8e-13  ***
Residuals   45    469        10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> library(ggplot2)                                               (3)
> ggplot(plotdata,                                               (3)
       aes(x = trt, y = mean, group = 1)) +                      (3)
    geom_point(size = 3, color="red") +                          (3)
    geom_line(linetype="dashed", color="darkgrey") +             (3)
    geom_errorbar(aes(ymin = mean - ci,                          (3)
                      ymax = mean + ci),                         (3)
                  width = .1) +                                  (3)
    theme_bw() +                                                 (3)
    labs(x="Treatment",                                          (3)
         y="Response",                                           (3)
         title="Mean Plot with 95% Confidence Interval")         (3)
```

(1) **Объем выборки в каждой группе, средние, стандартные отклонения и 95%-ные доверительные интервалы.**

(2) **Проверка наличия межгрупповых различий (ANOVA).**

3 Вывод диаграммы со средними значениями групп и доверительными интервалами.

Судя по полученным результатам, каждый способ лечения был опробован на 10 пациентах 1. Сравнение средних значений показывает, что прием препарата `drugE` дает наиболее заметное снижение уровня холестерина, а режим приема `time1` наименее эффективен 2. Стандартные отклонения во всех пяти группах примерно одинаковые и колеблются в диапазоне от `2.88` до `3.48`. Предполагается, что каждая группа в этом исследовании является выборкой из большей совокупности пациентов, которые могли бы получить лечение. Для каждого типа лечения `mean +/- ci` дает интервал, в котором с уверенностью на 95 % находятся истинные средние значения генеральной совокупности. F-критерий для типа лечения (`trt`) статистически значим (`p < .0001`), что свидетельствует о неодинаковой эффективности этих пяти способов лечения 2.

С помощью функций из пакета `ggplot2` строится диаграмма, отображающая средние значения и доверительные интервалы для каждой группы 3. Такая диаграмма с 95%-ными доверительными интервалами представлена на рис. 9.1 . На ней хорошо видны межгрупповые различия.

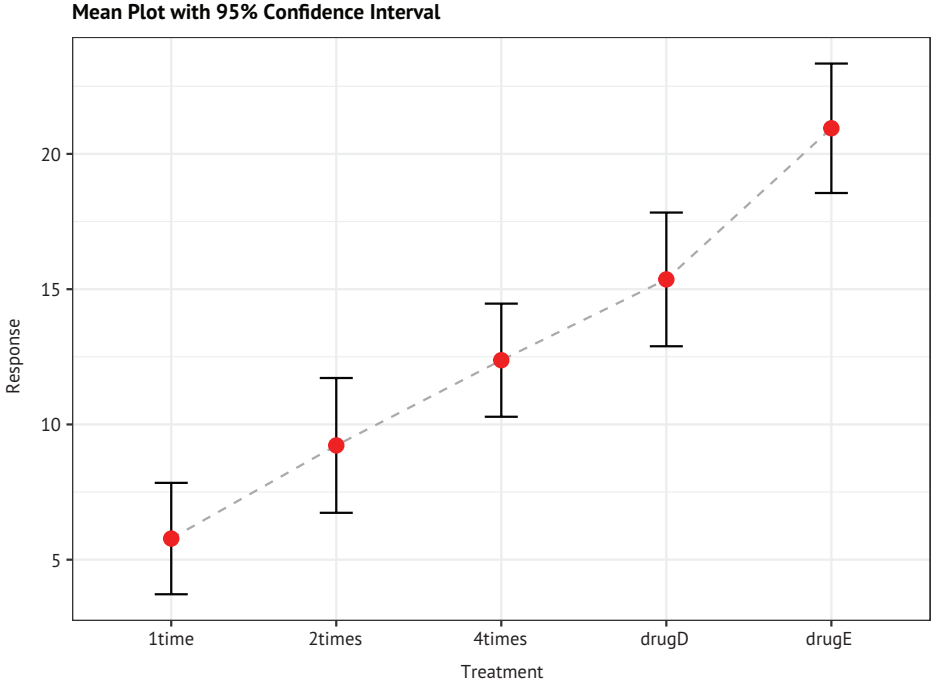


Рис. 9.1. Средние значения и 95%-ные доверительные интервалы для пяти курсов лечения, снижающих уровень холестерина

Добавив доверительные интервалы в диаграмму на рис. 9.1, мы показываем степень уверенности (или неопределенности) в наших оценках средних значений генеральной совокупности.

9.3.1. Множественное сравнение

Результат проверки F-критерия для способа лечения свидетельствует, что пять опробованных методов лечения неодинаково эффективны, однако из этого результата нельзя понять, *какие именно* способы различаются между собой. Для ответа на этот вопрос можно использовать множественное сравнение. К примеру, функция `TukeyHSD()` позволяет попарно проверить различия между средними значениями для всех групп, как показано в листинге 9.2.

Листинг 9.2. Попарное сравнение групп с использованием функции TukeyHSD()

```
> pairwise <- TukeyHSD(fit)                                          (1)
> pairwise

Fit: aov(formula = response ~ trt)

$trt
                diff    lwr   upr p adj
2times-1time    3.44 -0.658  7.54 0.138
4times-1time    6.59  2.492 10.69 0.000
drugD-1time     9.58  5.478 13.68 0.000
drugE-1time    15.17 11.064 19.27 0.000
4times-2times   3.15 -0.951  7.25 0.205
drugD-2times    6.14  2.035 10.24 0.001
drugE-2times   11.72  7.621 15.82 0.000
drugD-4times    2.99 -1.115  7.09 0.251
drugE-4times    8.57  4.471 12.67 0.000
drugE-drugD     5.59  1.485  9.69 0.003

> plotdata <- as.data.frame(pairwise[[1]])                           (2)
> plotdata$conditions <- row.names(plotdata)                         (2)

> library(ggplot2)                                                   (3)
> ggplot(data=plotdata, aes(x=conditions, y=diff)) +                 (3)
    geom_point(size=3, color="red") +                                (3)
    geom_errorbar(aes(ymin=lwr, ymax=upr, width=.2)) +               (3)
    geom_hline(yintercept=0, color="red", linetype="dashed") +       (3)
       labs(y="Difference in mean levels", x="",                     (3)
         title="95% family-wise confidence level") +                 (3)
    theme_bw() +                                                     (3)
    coord_flip()                                                     (3)
```

(1) **Вычисление оценок попарного сравнения.**

(2) **Создание набора данных с результатами.**

(3) **Вывод диаграммы с результатами.**

Например, средние уровни холестерина для протоколов лечения `1time` и `2times` статистически не отличаются друг от друга (`p = 0.138`), тогда как разница между протоколами `1time` и `4times` значима (`p < .001`).

На рис. 9.2 изображены диаграммы с результатами попарного сравнения. На этой диаграмме доверительные интервалы, в которые входит `0`, означают, что между двумя типами лечения нет статистически значимой разницы (`p > 0.5`). Здесь видно, что наибольшая разница между средними значениями наблюдается для переменных `drugE` и `1time` и эта разница является значимой (доверительный интервал не включает 0).

Прежде чем двинуться дальше, я должен отметить, что мы могли бы построить диаграмму, изображенную на рис. 9.3, используя инструменты из базовой версии R. В этом случае код мог бы просто вызывать `plot(pairwise)`. Преимущество использования пакета `ggplot2` в том, что он создает более привлекательные диаграммы и позволяет настраивать их в соответствии со своими требованиями.

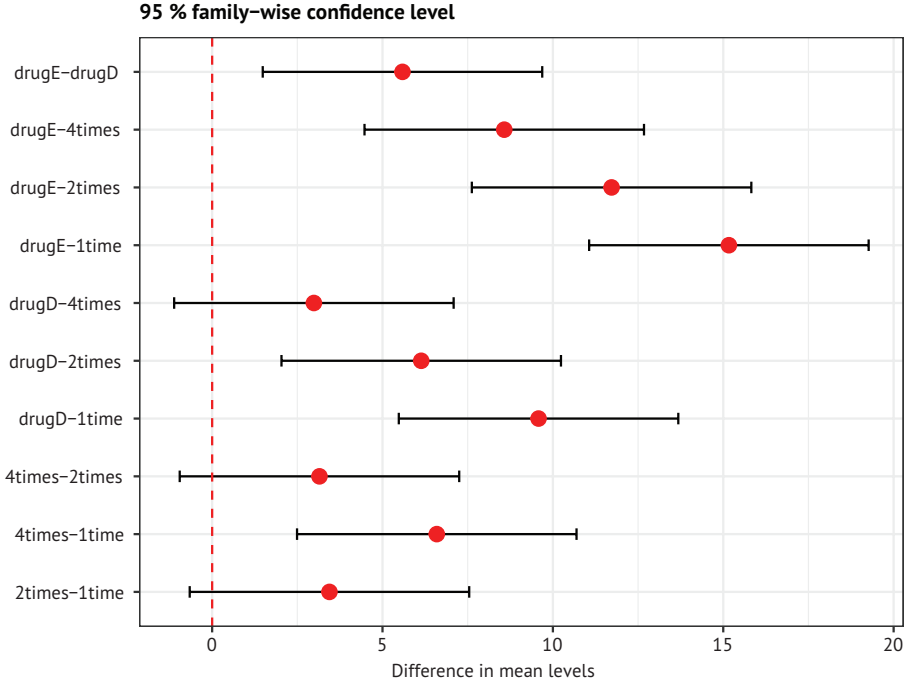


Рис. 9.2. Диаграмма для иллюстрации результатов попарного сравнения средних значений при помощи критерия Тьюки

Функция `glht()` из пакета `multcomp` реализует набор более сложных методов сравнения нескольких средних значений. Эти методы можно использовать как для линейных моделей (которые обсуждаются в этой главе), так и для обобщенных линейных моделей

(описанных в главе 13). Следующий код проверяет оценки достоверности различий Тьюки (Tukey Honest Significant Differences) и представляет результаты в графическом виде (рис. 9.3):

```
> tuk <- glht(fit, linfct=mcp(trt="Tukey"))
> summary(tuk)

     Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts


Fit: aov(formula = response ~ trt, data = cholesterol)

Linear Hypotheses:
                     Estimate Std. Error t value Pr(>|t|)
2times - 1time == 0     3.443      1.443   2.385  0.13812
4times - 1time == 0     6.593      1.443   4.568  < 0.001 ***
drugD - 1time == 0      9.579      1.443   6.637  < 0.001 ***
drugE - 1time == 0     15.166      1.443  10.507  < 0.001 ***
4times - 2times == 0    3.150      1.443   2.182  0.20504
drugD - 2times == 0     6.136      1.443   4.251  < 0.001 ***
drugE - 2times == 0    11.723      1.443   8.122  < 0.001 ***
drugD - 4times == 0     2.986      1.443   2.069  0.25120
drugE - 4times == 0     8.573      1.443   5.939  < 0.001 ***
drugE - drugD == 0      5.586      1.443   3.870  0.00308 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)

> labels1 <- cld(tuk, level=.05)$mcletters$Letters
> labels2 <- paste(names(labels1), "\n", labels1)
> ggplot(data=fit$model, aes(x=trt, y=response)) +
    scale_x_discrete(breaks=names(labels1), labels=labels2) +
    geom_boxplot(fill="lightgrey") +
    theme_bw() +
    labs(x="Treatment",
         title="Distribution of Response Scores by Treatment",
        subtitle="Groups without overlapping letters differ significantly
        (p < .05)")
```

Параметр `level` функции `cld()` позволяет задать уровень значимости (в данном случае `0.05`, или 95 % уверенности).

Группы (которые представлены в виде коробчатых диаграмм) обозначены одинаковыми буквами, если их средние значения статистически не различаются. Отсюда можно видеть, что протоколы лечения `1time` и `2times` не различаются (они оба обозначены буквой `a`). Протоколы `2times` и `4times` также не различаются (обе обозначены буквой `b`), но протоколы `1time` и `4time` различны (kyb обозначены разными буквами). Лично мне проще интерпретировать рис. 9.3, чем рис. 9.2. Диаграмма на рис. 9.3 показывает информацию о распределении оценок в каждой группе.

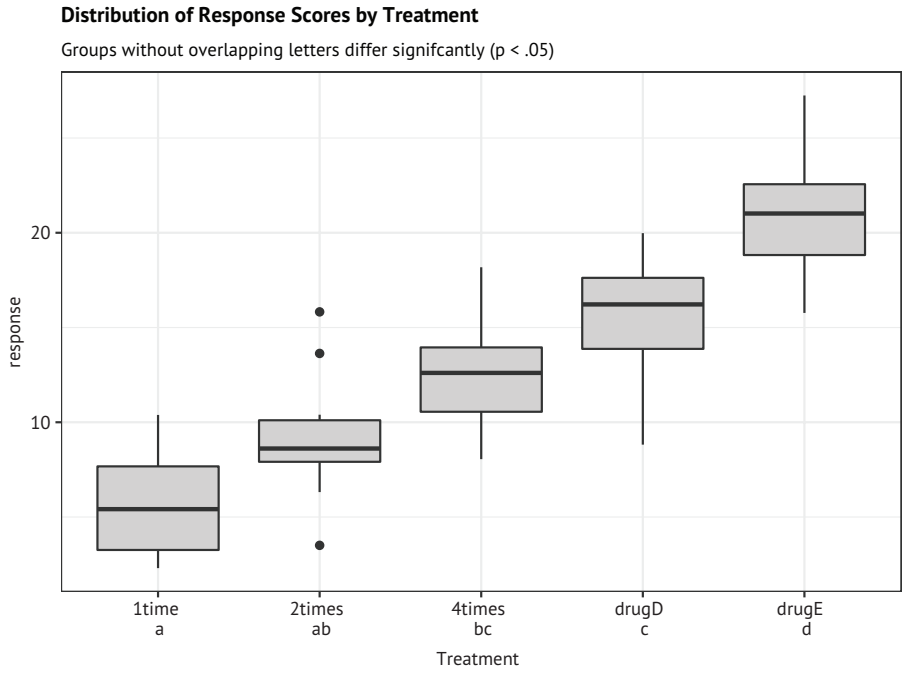


Рис. 9.3. Диаграмма с результатами вычисления критерия Тьюки, реализованная в пакете multcomp

Из этих результатов видно, что принимать лекарство, снижающее уровень холестерина, лучше по 5 мг четыре раза в день, чем по 20 мг один раз в день. Альтернативный препарат `drugD` не показал более высокой эффективности, по сравнению с четырехразовым приемом первого препарата. А вот лекарство `drugE` действует лучше, чем все остальные методы лечения.

Методология множественного сравнения – это сложная и быстро изменяющаяся область исследований. Для получения дальнейшей информации ознакомьтесь с публикацией Bretz, Hothorn и Westfall (2010).

9.3.2. Проверка справедливости предположений

Как мы узнали в предыдущей главе, степень уверенности в результатах зависит от степени соответствия данных предположениям, положенным в основу статистических тестов. В случае однофакторного дисперсионного анализа предполагается, что значения зависимой переменной распределены нормально и имеют одинаковую дисперсию во всех группах. Для проверки нормальности распределения можно использовать Q-Q-диаграмму:

```
> library(car)
> fit <- aov(response ~ trt, data=cholesterol)
> qqPlot(fit, simulate=TRUE, main="Q-Q Plot")
```

Эта диаграмма показана на рис. 9.4. По умолчанию два наблюдения с самыми большими стьюдентизированными остатками идентифицируются номерами строк в таблице данных. Данные хорошо укладываются в 95%-ные доверительные границы, а это значит, что предположение о нормальности распределения вполне справедливо.

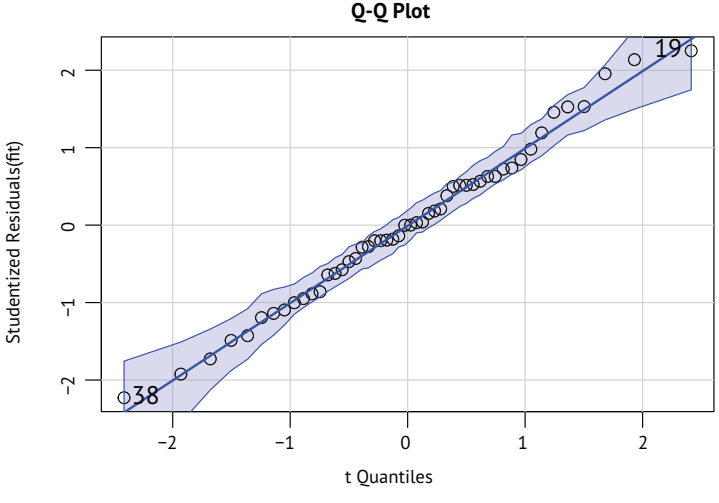


Рис. 9.4. Проверка нормальности распределения стьюдентизированных остатков. Остатки – это разность между фактическими и прогнозируемыми результатами, а стьюдентизированные остатки – это остатки, деленные на оценки их стандартных отклонений. Если стьюдентизированные остатки распределены нормально, они должны группироваться вокруг линии

R предлагает несколько критериев для проверки однородности (гомогенности) дисперсий. Например, вот как можно вычислить критерий Барлетта (Bartlett's test):

```
> bartlett.test(response ~ trt, data=cholesterol)

        Bartlett test of homogeneity of variances

data:  response by trt
Bartlett's K-squared = 0.5797, df = 4, p-value = 0.9653
```

Результат вычисления этого критерия показывает, что дисперсии в пяти группах статистически не различаются (`p = 0.97`). Другие возможные критерии: критерий Флигнера–Киллина (вычисляется при помощи функции `fligner.test()`) и критерий Брауна–Форсайта (вычисляется при помощи функции `hov()` из пакета `HH`). Результаты вычисления этих двух критериев здесь не приводятся, но смею вас уверить, что они позволяют прийти к тому же заключению.

Наконец, методы дисперсионного анализа могут быть чувствительны к выбросам. Проверить данные на наличие выбросов можно при помощи функции `outlierTest()` из пакета `car`:

```
> library(car)
> outlierTest(fit)

No Studentized residuals with Bonferroni  p < 0.05
Largest |rstudent|:
   rstudent unadjusted p-value Bonferroni p
19 2.251149           0.029422           NA
```

Полученный результат означает, что в наборе данных `cholesterol` выбросы отсутствуют (`NA` определяется при `p > 1`). Рассмотрев вместе Q-Q-диаграмму, значение критерия Барлетта и результаты проверки на наличие выбросов, можно сказать, что наши данные достаточно хорошо соответствуют модели дисперсионного анализа. Это, в свою очередь, добавляет уверенности в полученных результатах.

9.4. Однофакторный ковариационный анализ

Однофакторный ковариационный анализ (ANCOVA) расширяет однофакторный дисперсионный анализ добавлением одной или нескольких количественных ковариат. Следующий пример основан на наборе данных `litter` из пакета `multcomp` (Westfall et al., 1999). Беременные мыши были разделены на четыре экспериментальные группы. Мышам из каждой группы давали разные дозы лекарства (0, 5, 50 или 500). Средний вес новорожденных мышат в каждом помете играл роль зависимой переменной, а время беременности было включено в модель как ковариата. Проведенный анализ представлен в листинге 9.3.

Листинг 9.3. Однофакторный ковариационный анализ

```
> library(multcomp)
> library(dplyr)
> litter %>%
    group_by(dose) %>%
    summarise(n=n(), mean=mean(gesttime), sd=sd(gesttime))

  dose      n  mean    sd
  <fct> <int> <dbl> <dbl>
1 0        20  22.1 0.438
2 5        19  22.2 0.451
3 50       18  21.9 0.404
4 500      17  22.2 0.431

> fit <- aov(weight ~ gesttime + dose, data=litter)
> summary(fit)
            Df Sum Sq Mean Sq F value  Pr(>F)
gesttime     1  134.3  134.30   8.049 0.00597 **
dose         3  137.1   45.71   2.739 0.04988 *
Residuals   69 1151.3   16.69
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Функция summarise() сообщает, что мыши из разных групп получали разные дозы препарата: 20 животных не подвергались действию лекарства, а 17 мышей получили 500 единиц препарата. На основании средних значений по группам можно сказать, что для мышей, не получавших лекарство, в целом характерен наибольший вес потомства (32.3). Результаты вычисления F-критерия свидетельствуют о том, что (а) срок беременности связан с весом мышат при рождении и (б) количество лекарства влияло на вес мышат при постоянных значениях срока беременности. Средний вес мышат из разных экспериментальных групп неодинаков при постоянных значениях срока беременности.

Поскольку в анализе участвует ковариата, может появиться желание выяснить скорректированные средние по группам – без учета эффекта ковариаты. Для вычисления скорректированных средних значений можно использовать функцию effect() из пакета effects:

```
> library(effects)
> effect("dose", fit)

 dose effect
dose
   0    5   50  500
32.4 28.9 30.6 29.3
```

Здесь выводится средний вес мышат в каждой группе мышей после статистической корректировки первоначальных различий во времени беременности. В этом случае скорректированные средние немного отличаются от нескорректированных, полученных функцией summarise(). В пакете effects реализован мощный метод вычисления и графического представления скорректированных средних значений для исследований со сложным планом. Более подробную информацию ищите в документации с описанием пакета в CRAN.

Как и в примере с однофакторным дисперсионным анализом из предыдущего раздела, результаты вычисления F-критерия для дозы препарата свидетельствуют, что средний вес мышат в разных экспериментальных группах неодинаков, однако не ясно, какие именно пары средних значений различаются. И снова воспользуемся множественными попарными сравнениями средних значений при помощи пакета multcomp. Кроме того, этот пакет можно использовать для проверки определенных пользовательских гипотез о средних значениях.

Предположим, что нас интересует значимость различий между группой, не получавшей лекарства, и тремя группами, которым давали три разные дозы препарата. Для проверки этой гипотезы можно использовать код в листинге 9.4.

Листинг 9.4. Множественное сравнение с использованием заданных пользователем контрастов

```
> library(multcomp)
> contrast <- rbind("no drug vs. drug" = c(3, -1, -1, -1))
> summary(glht(fit, linfct=mcp(dose=contrast)))

Multiple Comparisons of Means: User-defined Contrasts

Fit: aov(formula = weight ~ gesttime + dose)

Linear Hypotheses:
                       Estimate Std. Error t value Pr(>|t|)
no drug vs. drug == 0    8.284      3.209   2.581   0.0120 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Контраст `c(3, -1, -1, -1)` означает сравнение первой группы с усредненными по трем остальным группам значениями. Гипотеза проверяется при помощи уравнения

$$3\times\mu_0 - 1\times\mu_5 - 1\times\mu_{50} - 1\times\mu_{500} = 0$$

или

$$\mu_0 = \frac{\mu_5 + \mu_{50} + \mu_{500}}{3},$$

где μ_n – средний вес мышат в группе, получавшей лекарство в дозировке n. Гипотеза проверяется с помощью критерия Стьюдента (в данном случае он равен `2.581`), значение которого значимо на уровне `p < 0.05`. Таким образом, можно утверждать, что не получавшая лекарств группа характеризуется бóльшим весом новорожденных мышат, чем все другие. Другие контрасты можно добавлять при помощи функции `rbind()` (более подробную информацию ищите в справке `help(glht)`).

9.4.1. Проверка справедливости предположений

Ковариационный анализ основан на тех же предположениях о нормальности и гомогенности дисперсии, что и дисперсионный анализ. Проверить их справедливость можно при помощи тех же методов, что были описаны в разделе 9.3.2. Кроме того, стандартный план ковариационного анализа предполагает однородность углов наклона линий регрессии. В данном случае предполагается, что наклон регрессионной линии, отражающей зависимость веса мышат от срока беременности, одинаков для всех четырех экспериментальных групп. Проверку на однородность углов наклонов линий регрессии можно провести, включив в ковариационную модель комбинацию со сроком беременности и дозой препарата.

Значимость влияния комбинации будет означать, что характер связи между длительностью беременности и весом мышат зависит от дозы препарата. Программный код и результаты представлены в листинге 9.5.

Листинг 9.5. Проверка гомогенности наклонов линий регрессии

```
> library(multcomp)
> fit2 <- aov(weight ~ gesttime*dose, data=litter)
> summary(fit2)
             Df Sum Sq Mean Sq F value Pr(>F)
gesttime      1    134     134    8.29 0.0054 **
dose          3    137      46    2.82 0.0456 *
gesttime:dose 3     82      27    1.68 0.1789
Residuals    66   1069      16
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Влияние комбинации статистически незначимо, что подтверждает справедливость предположения о равенстве наклонов линий регрессии. Если бы предположение не выполнялось, то можно было бы попробовать преобразовать ковариату или зависимую переменную с использованием модели, которая учитывает только отдельные углы наклона[1]. Также можно было бы применить непараметрический ковариационный анализ, для которого однородность углов наклона линий регрессии необязательна. Этот метод можно реализовать при помощи функции `sm.ancova()` из пакета `sm`.

9.4.2. Визуализация результатов

Для визуализации взаимосвязи между зависимой переменной, ковариатой и фактором можно использовать пакет `ggplot2`. Например, следующий код выведет диаграмму, изображенную на рис. 9.5:

```
pred <- predict(fit)
library(ggplot2)
ggplot(data = cbind(litter, pred),
       aes(gesttime, weight)) + geom_point() +
   facet_wrap(~ dose, nrow=1) + geom_line(aes(y=pred)) +
   labs(title="ANCOVA for weight by gesttime and dose") +
   theme_bw() +
   theme(axis.text.x = element_text(angle=45, hjust=1),
         legend.position="none")
```

[1] На самом деле в одну ковариационную модель могут быть включены разные регрессионные коэффициенты. – *Прим. перев.*

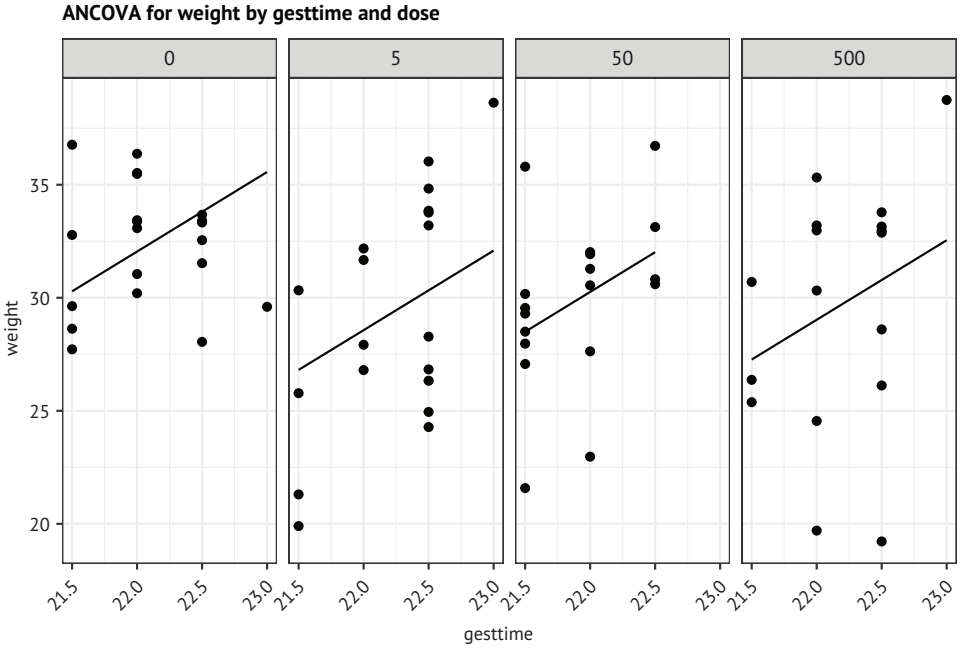


Рис. 9.5. Диаграмма, отображающая зависимость между длительностью беременности и весом новорожденных мышат в каждой экспериментальной группе

Здесь можно видеть, что регрессионные линии, аппроксимирующие зависимость веса новорожденных мышат от длительности беременности в разных экспериментальных группах, параллельны, но имеют разные свободные члены. С ростом длительности беременности растет и вес мышат. Также можно видеть, что в группе, где мыши не получали препарата, свободный член максимален, а в группе, где мыши получали наибольшую дозу препарата, свободный член минимален. Линии параллельны, потому что они были так заданы. Если бы для вывода диаграммы использовался код

```
ggplot(data = litter, aes(gesttime, weight)) +
       geom_point() + geom_smooth(method="lm", se=FALSE) +
       facet_wrap(~ dose, nrow=1)
```

то можно было бы наблюдать межгрупповую изменчивость как наклонов, так и свободных членов. Такой подход полезен в случае, когда требование однородности углов наклона линий регрессии не выполняется.

9.5. Двухфакторный дисперсионный анализ

В двухфакторном дисперсионном анализе объекты распределяются по группам, которые задаются комбинацией двух факторов. Мы по-

знакомимся с двухфакторным дисперсионным анализом для зависимых переменных на примере набора данных ToothGrowth, входящего в состав дистрибутива R. Шестьдесят морских свинок были случайно распределены по группам, получавшим три разных количества аскорбиновой кислоты (0,5, 1 или 2 мг) двумя способами (в виде апельсинового сока или витамина C). Всего было сформировано шесть групп по 10 свинок в каждой. Зависимая переменная – длина зубов. В листинге 9.6 приводится программный код для анализа.

Листинг 9.6. Двухфакторный дисперсионный анализ

```
> library(dplyr)
> data(ToothGrowth)
> ToothGrowth$dose <- factor(ToothGrowth$dose)                    (1)
> stats <- ToothGrowth %>%                                        (2)
    group_by(supp, dose) %>%                                      (2)
    summarise(n=n(), mean=mean(len), sd=sd(len),                  (2)
              ci = qt(0.975, df = n - 1) * sd / sqrt(n))          (2)
> stats

# Таблица: 6×6
# Группы:   supp [2]
  supp  dose      n  mean    sd    ci
  <fct> <fct> <int> <dbl> <dbl> <dbl>
1 OJ    0.5      10 13.2   4.46  3.19
2 OJ    1        10 22.7   3.91  2.80
3 OJ    2        10 26.1   2.66  1.90
4 VC    0.5      10  7.98  2.75  1.96
5 VC    1        10 16.8   2.52  1.80
6 VC    2        10 26.1   4.80  3.43

> fit <- aov(len ~ supp*dose, data=ToothGrowth)                   (3)
> summary(fit)

            Df Sum Sq Mean Sq F value   Pr(>F)
supp         1  205.4   205.4  15.572 0.000231 ***
dose         2 2426.4  1213.2  92.000  < 2e-16 ***
supp:dose    2  108.3    54.2   4.107 0.021860 *
Residuals   54  712.1    13.2
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

(1) **Подготовка данных.**

(2) **Вычисление обобщенных статистик.**

(3) **Подгонка модели двухфакторного дисперсионного анализа.**

Сначала переменная dose преобразуется в фактор, поэтому функция aov() будет рассматривать ее как группирующую переменную, а не как числовую ковариату (1). Затем для каждой группы свинок вычисляются обобщенные статистики (количество, среднее значение, стандартное отклонение и доверительный интервал для

среднего значения) (2). Размеры выборки указывают, что мы имеем сбалансированный план (равные размеры выборок в каждой ячейке плана). Затем производится подгонка модели двухфакторного дисперсионного анализа к данным (3), а функция `summary()` показывает, что оба основных эффекта (форма подачи `supp` и доза `dose`) и их комбинация имеют большое значение.

Результаты можно визуализировать несколькими способами, включая функцию `interaction.plot()` из базовой версии R, функцию `plotmeans()` из пакета `gplots` и функцию `interaction2wt()` из пакета `HH`. В программном коде, следующем за рис. 9.6, используется пакет `ggplot2`. Он строит графики средних значений и 95%-ных доверительных интервалов для средних. Одно из преимуществ пакета `ggplot2` – возможность настроить отображение диаграммы в соответствии с нашими исследовательскими и эстетическими потребностями. Полученная диаграмма показана на рис. 9.6.

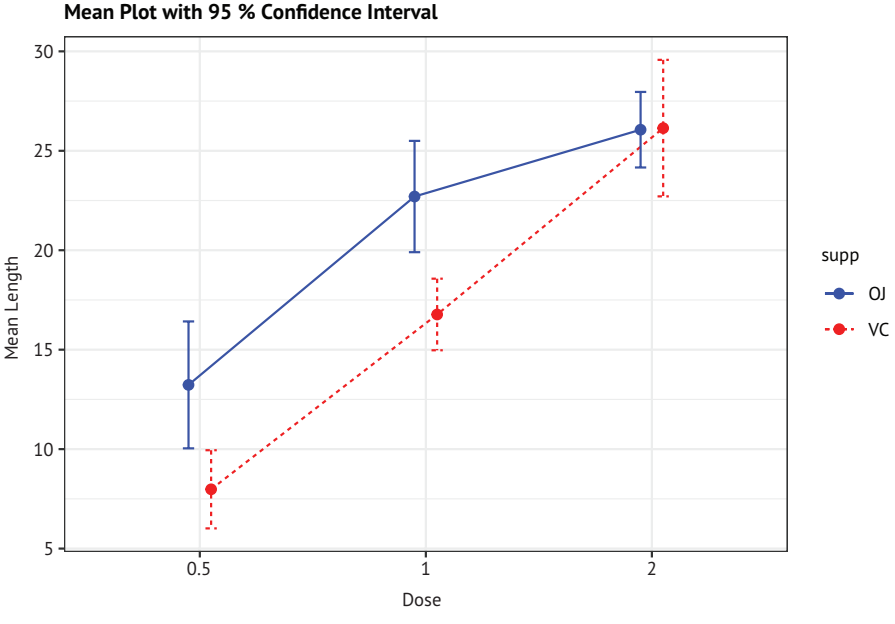


Рис. 9.6. Влияние комбинации дозы и способа ее получения на механизм роста зубов. Графическое отображение средних значений с 95%-ными доверительными интервалами получено при помощи пакета ggplot2

```
library(ggplot2)
pd <- position_dodge(0.2)
ggplot(data=stats,
       aes(x = dose, y = mean,              group=supp,
           color=supp,
           linetype=supp)) +
  geom_point(size = 2,
             position=pd) +
```

```
geom_line(position=pd) +
geom_errorbar(aes(ymin = mean - ci, ymax = mean + ci),
              width = .1,
              position=pd) +
theme_bw() +
scale_color_manual(values=c("blue", "red")) +
labs(x="Dose",
     y="Mean Length",
     title="Mean Plot with 95% Confidence Interval")
```

Как показывает диаграмма, рост зубов ускоряется с увеличением дозы аскорбиновой кислоты, независимо от способа ее получения – в форме апельсинового сока или витамина C. Для доз в 0,5 и 1 мг апельсиновый сок оказывает более выраженный положительный эффект, по сравнению с витамином C. Для дозы в 2 мг обе формы потребления аскорбиновой кислоты одинаково хороши.

Я не буду обсуждать проверку справедливости предположений, лежащих в основе моделей, и сравнение средних – она ничем не отличается от процедур, представленных выше. Кроме того, если план вашего эксперимента сбалансирован, то вам не нужно беспокоиться о порядке эффектов.

9.6. Дисперсионный анализ повторных измерений

В дисперсионном анализе повторных измерений объекты измеряются более одного раза. В этом разделе обсуждается дисперсионный анализ повторных измерений с одним фактором, определяющим группы, и одним внутригрупповым фактором (обычный план эксперимента). Мы рассмотрим пример из области экологической физиологии. Этот раздел биологии исследует, как протекание физиологических и биохимических процессов в живых системах зависит от условий обитания (важнейшая область исследований, учитывая реалии глобального потепления). Набор данных `CO2`, входящий в стандартный дистрибутив R, содержит результаты исследования холодоустойчивости северных и южных популяций злака ежовника (Echinochloa crus-galli: Potvin, Lechowicz, Tardif, 1990). Сравнивалась интенсивность фотосинтеза охлажденных и неохлажденных растений при разных концентрациях углекислого газа в окружающей среде. Половина растений происходила из Квебека, а половина – из штата Миссисипи.

В этом примере мы сосредоточимся на охлажденных растениях. Зависимая переменная – это уровень потребления углекислого газа (`uptake`) в мл/л, а независимые переменные (факторы) – это штат (`type`: Квебек или Миссисипи) и концентрация углекислого газа в окружающей среде (`conc`: семь концентраций от 95 до 1000 μмоль/м^{2}с). Штат – это разделяющий группы фактор, а концентрация – внутригрупповой фактор. Анализ представлен в листинге 9.7.

Листинг 9.7. Дисперсионный анализ повторных измерений с одним межгрупповым и одним внутригрупповым фактором

```
> data(CO2)
> CO2$conc <- factor(CO2$conc)
> w1b1 <- subset(CO2, Treatment=='chilled')
> fit <- aov(uptake ~ conc*Type + Error(Plant/(conc)), w1b1)
> summary(fit)

Error: Plant
          Df Sum Sq Mean Sq F value Pr(>F)
Type       1   2667    2667    60.4 0.0015 **
Residuals  4    177      44
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error: Plant:conc
          Df Sum Sq Mean Sq F value  Pr(>F)
conc       6   1472   245.4    52.5 1.3e-12 ***
conc:Type  6    429    71.5    15.3 3.7e-07 ***
Residuals 24    112     4.7
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> library(dplyr)
> stats <- CO2 %>%
   group_by(conc, Type) %>%
   summarise(mean_conc = mean(uptake))

> library(ggplot2)
> ggplot(data=stats, aes(x=conc, y=mean_conc,
          group=Type, color=Type, linetype=Type)) +
   geom_point(size=2) +
   geom_line(size=1) +
   theme_bw() + theme(legend.position="top") +
   labs(x="Concentration", y="Mean Uptake",
        title="Interaction Plot for Plant Type and Concentration")
```

Результаты дисперсионного анализа, представленные в таблице, свидетельствуют о том, что главные эффекты (штат и концентрация), а также их комбинация значимы на уровне `p < 0.01`. Диаграмма, иллюстрирующая взаимодействия между факторами, представлена на рис. 9.7.

Для демонстрации другого способа графического отображения влияния комбинации тех же данных я использовал функцию `geom_boxplot()` (рис. 9.8).

```
library(ggplot2)
ggplot(data=CO2, aes(x=conc, y=uptake, fill=Type)) +
  geom_boxplot() +
  theme_bw() + theme(legend.position="top") +
  scale_fill_manual(values=c("aliceblue", "deepskyblue"))+
```

```
labs(x="Concentration", y="Uptake",
     title="Chilled Quebec and Mississippi Plants")
```

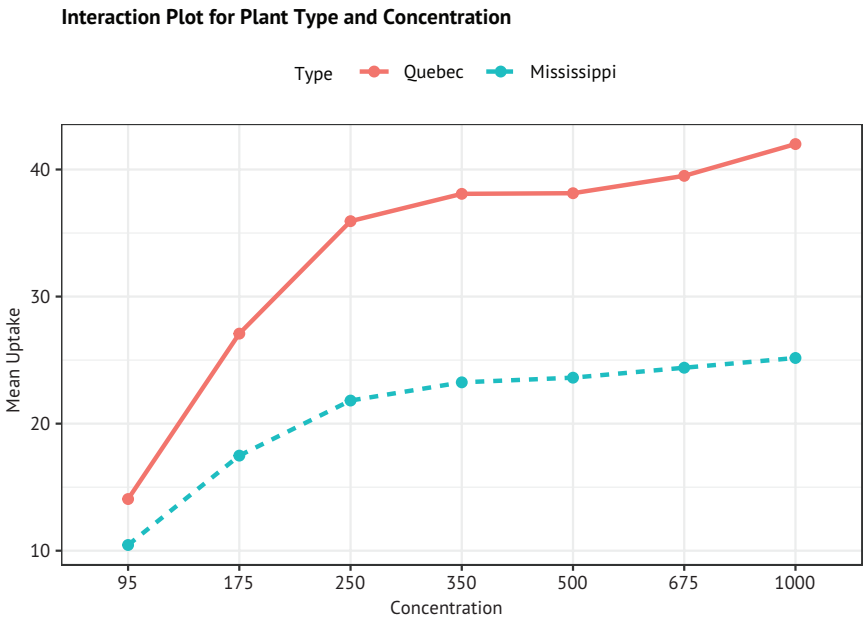


Рис. 9.7. Зависимость потребления углекислого газа от комбинации концентрации углекислого газа в окружающей среде и географического происхождения растения. Диаграмма построена при помощи пакета ggplot2

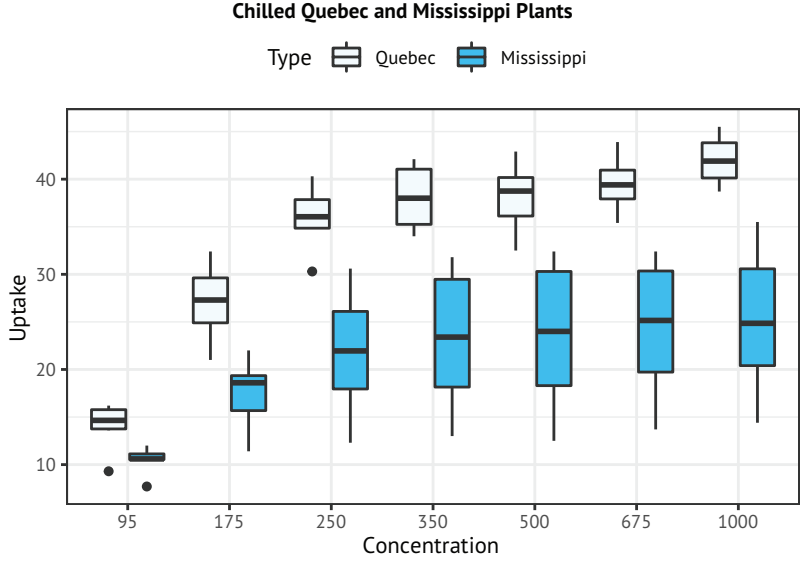


Рис. 9.8. Зависимость потребления углекислого газа от комбинации концентрации углекислого газа в окружающей среде и географического происхождения растения. Диаграмма построена при помощи функции geom_boxplot()

На обеих диаграммах видно, что у растений из Квебека интенсивность потребления углекислого газа выше, чем у растений из Миссисипи. Различия наиболее ярко выражены при высоких концентрациях углекислого газа в окружающей среде.

ПРИМЕЧАНИЕ. Обычно в анализе используются наборы данных в *широком формате*, когда столбцы – это переменные, строки – наблюдения, а каждому объекту соответствует отдельная строка. Хороший пример такого формата – набор данных `litter`, обсуждавшийся в разделе 9.4. При работе с повторными измерениями обычно для подбора моделей нужны данные в *длинном формате*. В таком формате каждому измерению зависимой переменной соответствует своя строка. Набор данных `CO2` представлен именно в таком длинном формате. К счастью, при помощи пакета `tidyr`, описанного в главе 5 (раздел 5.5.2), можно с легкостью преобразовать данные в нужный формат.

Многообразие подходов к анализу экспериментов со смешанными эффектами

Пример с углекислым газом был проанализирован с использованием традиционного дисперсионного анализа повторных наблюдений. Этот подход подразумевает, что ковариационная матрица для любого внутригруппового фактора обладает определенным свойством, известным как *сферичность*. Это значит, что дисперсии разниц между любыми двумя уровнями внутригруппового фактора одинаковы. Маловероятно, чтобы такое допущение выполнялось в реальном мире. Это соображение привело к возникновению ряда альтернативных подходов, включая:

- *использование функции* `lmer()` *из пакета* `lme4` *для подгонки линейных моделей со смешанными эффектами (Bates, 2005);*
- *использование функции* `Anova()` *из пакета* `car` *для корректировки обычных статистик с учетом отсутствия сферичности, например поправка Гейссера–Гринхауса (Geisser-Greenhouse);*
- *использование функции* `gls()` *из пакета* `nlme` *для подгонки обобщенных моделей методом наименьших квадратов с заданной структурой дисперсии и ковариации (UCLA, 2009);*
- *использование многомерного дисперсионного анализа для моделирования результатов повторных измерений (Hand, 1987).*

Обсуждение этих подходов выходит за рамки данной книги. Если вы хотите узнать больше, обращайтесь к публикациям Pinheiro, Bates (2000) и Zuur et al. (2009).

До этого момента все методы подразумевали наличие одной зависимой переменной. В следующем разделе мы кратко рассмотрим эксперименты с более чем одной зависимой переменной.

9.7. Многомерный дисперсионный анализ

Многомерный дисперсионный анализ (multivariate analysis of variance, MANOVA) позволяет исследовать несколько зависимых переменных одновременно. Мы рассмотрим этот подход на примере набора данных `UScereal` из пакета `MASS` (Venables, Ripley, 1999). В этом примере мы узнаем, как зависит содержание калорий (`calories`), жиров (`fat`) и углеводов (`sugars`) в американских сухих завтраках от того, на какой полке в магазине они стоят (`shelf`: 1 – нижняя полка, 2 – средняя, 3 – верхняя). Количество калорий, жиров и углеводов – это зависимые переменные, а номер полки – независимая переменная с тремя уровнями (1, 2 и 3). Анализ представлен в листинге 9.8.

Листинг 9.8. Однофакторный многомерный дисперсионный анализ

```
> data(UScereal, package="MASS")
> shelf <- factor(UScereal$shelf)
> shelf <- factor(shelf)
> y <- cbind(UScereal$calories, UScereal$fat, UScereal$sugars)
> colnames(y) <- c("calories", "fat", "sugars")
> aggregate(y, by=list(shelf=shelf), FUN=mean)

    shelf calories   fat sugars
1       1      119 0.662    6.3
2       2      130 1.341   12.5
3       3      180 1.945   10.9

> cov(y)

          calories   fat sugars
calories   3895.2 60.67 180.38
fat          60.7  2.71   4.00
sugars      180.4  4.00  34.05

> fit <- manova(y ~ shelf)
> summary(fit)

           Df Pillai approx F num Df den Df Pr(>F)
shelf       2  0.402     5.12      6    122  1e-04 ***
Residuals 62
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary.aov(fit)          (1)

Response calories :
            Df Sum Sq Mean Sq F value  Pr(>F)
shelf        2  50435   25218    7.86 0.00091 ***
Residuals   62 198860    3207
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Response fat :
            Df Sum Sq Mean Sq F value Pr(>F)
shelf        2   18.4    9.22    3.68  0.031 *
Residuals   62  155.2    2.50
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

 Response sugars :
            Df Sum Sq Mean Sq F value Pr(>F)
shelf        2    381     191    6.58 0.0026 **
Residuals   62   1798      29
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1 Вывод результатов для каждого показателя

Прежде всего переменная `shelf` преобразуется в фактор, чтобы ее можно было использовать в роли группирующей переменной. Далее для создания матрицы трех зависимых переменных (калории, жиры, углеводы) вызывается функция `cbind()`. При помощи функции `aggregate()` вычисляются средние значения для полок, а функция `cov()` вычисляет дисперсию и ковариацию по злакам.

Функция `manova()` осуществляет многомерный тест на межгрупповые различия. Статистически значимая величина F-критерия свидетельствует о том, что наши три группы злаков различаются по питательной ценности. Обратите внимание, что переменная `shelf` была преобразована в фактор, поэтому она может играть роль группирующей переменной.

Поскольку результаты вычисления многомерного критерия значимы, можно использовать функцию `summary.aov()` для выполнения ряда однофакторных дисперсионных анализов. В этом случае различия по каждому показателю пищевой ценности анализируются отдельно. Наконец, можно сравнить средние значения (при помощи функции `TukeyHSD`), чтобы выяснить, какие полки отличаются от каких по каждой из зависимых переменных (результат здесь не приводится для экономии места).

9.7.1. Проверка справедливости предположений

В основе однофакторного многомерного дисперсионного анализа лежат два допущения – нормальность многомерного распределения и однородность матриц дисперсий-ковариаций. Первое предположение заключается в том, что все зависимые переменные одновременно подчиняются закону нормального распределения. Для проверки справедливости этого предположения можно использовать Q-Q-диаграмму (статистическое обоснование приводится во врезке «Немного теории»).

Немного теории

Если у вас есть нормально распределенный многомерный вектор x с размерностью $p \times 1$, средним значением μ и ковариационной матрицей Σ, то квадрат расстояния Махаланобиса между x и μ имеет распределение хи-квадрат с p степенями свободы. Q-Q-диаграмма отображает зависимость выборочных квантилей распределения хи-квадрат от квадрата расстояний Махаланобиса. Мы можем быть уверенными в многомерном нормальном распределении данных в той степени, в какой точки ложатся на линию $x=y$.

В листинге 9.9 показан код, выполняющий подобную проверку, а полученная диаграмма представлена на рис. 9.9.

Листинг 9.9. Проверка нормальности многомерного распределения

```
> center <- colMeans(y)
> n <- nrow(y)
> p <- ncol(y)
> cov <- cov(y)
> d <- mahalanobis(y,center,cov)
> coord <- qqplot(qchisq(ppoints(n),df=p),
    d, main="Q-Q Plot Assessing Multivariate Normality",
    ylab="Mahalanobis D2")
> abline(a=0,b=1)
> identify(coord$x, coord$y, labels=row.names(UScereal))
```

Если данные соответствуют многомерному нормальному распределению, то точки ложатся на линию. Функция `identify()` позволяет выводить метки у точек на диаграмме в интерактивном режиме, нужно лишь щелкнуть мышью на интересующей точке. Нажатие клавиши Esc или щелчок на кнопке Finish (Стоп) в правом верхнем углу диаграммы отключают интерактивный режим. В нашем случае обнаруживается отклонение от многомерного нормального распределения в основном из-за сухих завтраков «Уитиз» (Wheaties Honey Gold и Wheaties). Возможно, эти случаи стоит удалить, а затем повторить анализ.

Однородность матриц дисперсии-ковариации подразумевает равенство ковариационных матриц в каждой группе. Это предположение обычно проверяют при помощи М-критерия Бокса. В R нет функции для вычисления этого критерия, но в интернете можно найти нужный программный код. К сожалению, этот критерий чувствителен к отклонениям от нормальности, что в большинстве случаев приводит к отрицательным результатам. Это значит, что пока у нас нет хорошего рабочего метода для проверки этого важного предположения (однако есть альтернативные подходы, описанные в публикациях Anderson (2006) и Silva et al. (2008), которые пока не реализованы в R).

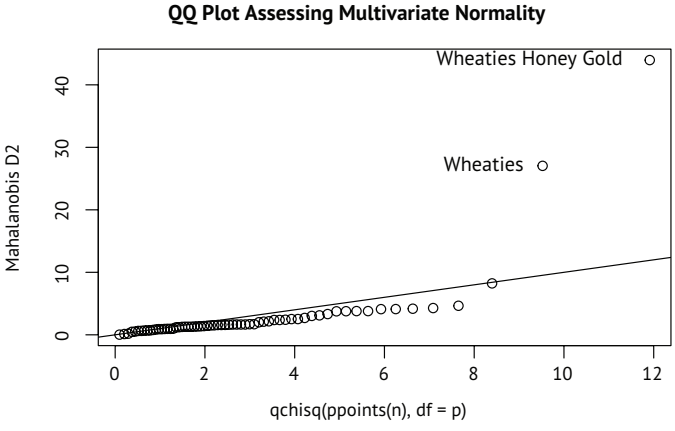


Рис. 9.9. Q-Q-диаграмма для проверки данных на соответствие многомерному нормальному распределению

Наконец, можно проверить выборку на наличие многомерных выбросов, используя функцию `aq.plot()` из пакета `mvoutlier`:

```
library(mvoutlier)
outliers <- aq.plot(y)
outliers
```

Попробуйте выполнить его и посмотрите, что получится!

9.7.2. Устойчивый многомерный дисперсионный анализ

Если предположения о нормальности многомерного распределения или гомогенности матриц дисперсии-ковариации не подтверждаются, или беспокоит возможность наличия многомерных выбросов, можно попробовать применить устойчивую (робастную) или непараметрическую версию многомерного дисперсионного анализа. Для этого можно использовать функцию `Wilks.test()` из пакета `rrcov`. Функция `adonis()` из пакета `vegan` реализует непараметрический аналог многомерного дисперсионного анализа. Код в листинге 9.10 демонстрирует применение функции `Wilks.test()` к нашему примеру.

Листинг 9.10. Устойчивый однофакторный многомерный дисперсионный анализ

```
> library(rrcov)
> Wilks.test(y,shelf,method="mcd")

        Robust One-way MANOVA (Bartlett Chi2)

data:  x
Wilks' Lambda = 0.511, Chi2-Value = 23.96, DF = 4.98, p-value =
0.0002167
```

```
sample estimates:
  calories    fat  sugars
1      120  0.701    5.66
2      128  1.185   12.54
3      161  1.652   10.35
```

Судя по результатам, критерий устойчивости, нечувствительный и к выбросам, и к отклонениям от предположений многомерного дисперсионного анализа, по-прежнему указывает на неравноценность сухих завтраков на верхней, средней и нижней полках магазинов по их питательной ценности.

9.8. Дисперсионный анализ как регрессия

В разделе 9.2 отмечалось, что дисперсионный анализ и регрессия – это два частных случая одной и той же общей линейной модели. В таком случае все примеры в этой главе можно было бы проанализировать при помощи функции `lm()`. Однако чтобы понять результаты применения этой функции, нужно узнать, как R обрабатывает категориальные данные при подгонке моделей.

Рассмотрим задачу одномерного дисперсионного анализа из раздела 9.3, в которой сравниваются пять способов снижения уровня холестерина (закодированных в переменной `trt`):

```
> library(multcomp)
> levels(cholesterol$trt)

[1] "1time"  "2times" "4times" "drugD"  "drugE"
```

Сначала выполним подгонку модели с помощью функции `aov()`:

```
> fit.aov <- aov(response ~ trt, data=cholesterol)
> summary(fit.aov)

            Df   Sum Sq  Mean Sq  F value     Pr(>F)
trt          4  1351.37   337.84   32.433  9.819e-13 ***
Residuals   45   468.75    10.42
```

Затем используем функцию `lm()`. В этом случае получаются следующие результаты (листинг 9.11):

Листинг 9.11. Регрессионный подход к задаче дисперсионного анализа из раздела 9.3

```
> fit.lm <- lm(response ~ trt, data=cholesterol)
> summary(fit.lm)

Coefficients:
            Estimate Std. Error t value   Pr(>|t|)
(Intercept)    5.782      1.021   5.665   9.78e-07 ***
trt2times      3.443      1.443   2.385     0.0213 *
trt4times      6.593      1.443   4.568   3.82e-05 ***
```

```
trtdrugD       9.579      1.443   6.637   3.53e-08 ***
trtdrugE      15.166      1.443  10.507   1.08e-13 ***

Residual standard error: 3.227 on 45 degrees of freedom
Multiple R-squared: 0.7425,     Adjusted R-squared: 0.7196
F-statistic: 32.43 on 4 and 45 DF,  p-value: 9.819e-13
```

Что мы видим? Поскольку для линейных моделей нужны числовые независимые переменные, то, обнаружив фактор, функция `lm()` замещает его набором числовых переменных, соответствующих контрастам между уровнями исходного фактора. Если фактор имеет k уровней, для него будет создано $k - 1$ переменных. В R есть пять встроенных методов создания таких контрастных переменных (табл. 9.6). Вы можете определить свой метод (но мы не будем обсуждать эту возможность здесь). По умолчанию контрасты комбинаций условий (`contr.treatment`) применяются для неупорядоченных факторов, а ортогональные полиномы (`contr.poly`) – для упорядоченных.

Таблица 9.6. Встроенные контрасты

Контраст	Описание
`contr.helmert`	Контрасты между первым уровнем и вторым, между третьим уровнем и средним значением для первых двух, между четвертым уровнем и средним значением для первых трех и т. д.
`contr.poly`	Контрасты для анализа трендов (линейных, квадратичных, кубических и т. д.) основаны на ортогональных полиномах. Применяются для упорядоченных факторов с одинаково различающимися уровнями
`contr.sum`	Контрасты ограничены условием равенства их суммы нулю. Также называемые *контрастами отклонения*, они сравнивают среднее значение для каждого уровня с общим средним по всем уровням
`contr.treatment`	Контрасты каждого уровня с базовым уровнем (по умолчанию первым). Также называется *кодированием при помощи индикаторных переменных* (dummy coding)
`contr.SAS`	Сходен с предыдущим способом, однако в качестве базового используется последний уровень. В результате получаются коэффициенты, сходные с контрастами, реализованными в большинстве методов SAS

В случае контрастов комбинаций условий первый уровень фактора становится эталонной группой и каждый следующий уровень сравнивается с ним. С этой схемой перекодирования можно познакомиться при помощи функции `contrasts()`:

```
> contrasts(cholesterol$trt)
       2times  4times  drugD  drugE
1time       0       0      0      0
2times      1       0      0      0
4times      0       1      0      0
drugD       0       0      1      0
drugE       0       0      0      1
```

Если пациент находится в группе drugD, то эта переменная равна 1, а переменные 2times, 4times и drugE будут равны 0. Для первой группы переменная не нужна, поскольку нули во всех четырех индикаторных переменных однозначно указывают, что такой пациент находится в группе 1times.

В листинге 9.11 переменная trt2times соответствует контрасту между уровнями фактора 1time и 2times. Аналогично trt4times – это контраст между 1time и 4times и т. д. По значениям вероятностей видно, что все способы лечения статистически значимо отличаются от первого (1time).

Контрасты, используемые функцией lm() по умолчанию, можно изменить при помощи параметра contrasts. Например, вот как можно задать контрасты Хелмерта:

```
fi t.lm <- lm(response ~ trt, data=cholesterol, contrasts="contr.helmert")
```

Изменять контрасты по умолчанию, используемые на протяжении всей сессии работы в R, можно при помощи функции options(). Например, выражение

```
options(contrasts = c("contr.SAS", "contr.helmert"))
```

привело бы к вычислению контрастов по умолчанию для неупорядоченных факторов по алгоритму contr.SAS, а для упорядоченных факторов – по алгоритму contr.helmert. Мы ограничили обсуждение использованием контрастов в линейных моделях, но имейте в виду, что они применяются и в других функциях подгонки моделей в R. Это справедливо и для обобщенных линейных моделей, обсуждаемых в главе 13.

Итоги

- Дисперсионный анализ (ANOVA) – это набор статистических методов, часто используемых для анализа данных экспериментальных и квазиэкспериментальных исследований.
- Дисперсионный анализ особенно востребован в исследованиях связей между количественной зависимой переменной и одной или несколькими категориальными независимыми переменными.
- Если количественная зависимая переменная связана с категориальной объясняющей переменной с *более чем* двумя уровнями, проводятся апостериорные проверки, чтобы определить, какие уровни/группы различаются по этому результату.
- При наличии двух или более категориальных объясняющих переменных можно использовать факторный дисперсионный анализ для изучения их влияния на зависимую переменную вместе и по отдельности.

- Когда влияние одной или нескольких количественных мешающих переменных статистически контролируется (удаляется), план называется ковариационным анализом (ANCOVA).
- Когда имеется более одной зависимой переменной, план называется многомерным дисперсионным или многомерным ковариационным анализом.
- Дисперсионный анализ и множественная регрессия – это два частных случая общей линейной модели. Различные терминологии, функции и выходные форматы этих двух подходов отражают их отдельное происхождение в разных областях исследований. Когда исследования сосредоточены на групповых различиях, результаты дисперсионного анализа проще интерпретировать.

10 Анализ мощности

В этой главе:

- определение необходимого размера выборки;
- вычисление размеров эффектов;
- оценка статистической мощности.

Будучи консультантом по статистике, я часто слышу вопрос: «Сколько объектов должно участвовать в исследовании?» Иногда вопрос принимает такую форму: «Я могу обследовать *x* человек. Есть ли смысл проводить исследование?» Ответы на подобные вопросы можно получить с помощью *анализа мощности* – набора методов, важных для планирования эксперимента.

Анализ мощности позволяет определить размер выборки, необходимый для выявления эффекта заданной величины с заданной долей уверенности. Верно и обратное, этот анализ позволяет оценить вероятность обнаружения эффекта заданной величины с заданной долей уверенности при данном объеме выборки. Если вероятность неприемлемо мала, будет разумно изменить эксперимент или отказаться от него.

В этой главе вы узнаете, как провести анализ мощности, применяя разные статистические критерии, включая критерии пропорций, Стьюдента, хи-квадрат, сбалансированный однофакторный дисперсионный анализ, корреляционные критерии и линейные

модели. Поскольку анализ мощности используется при проверке статистических гипотез, мы начнем с короткого обзора проверки значимости нулевых гипотез. Затем рассмотрим порядок выполнения анализа мощности в R, в основном при помощи пакета `pwr`. Наконец, обсудим другие подходы к анализу мощности, реализованные в R.

10.1. Краткий обзор проверки значимости гипотез

Чтобы помочь вам понять этапы анализа мощности, мы кратко обсудим общие вопросы проверки статистических гипотез. Если у вас уже есть представление об этом, можете спокойно перейти к разделу 10.2.

При проверке статистических гипотез формулируется гипотеза о параметре генеральной совокупности (*нулевая гипотеза*, H_0). Затем из данной генеральной совокупности извлекается выборка и вычисляется статистика, позволяющая строить суждения об этом параметре. Допуская, что нулевая гипотеза справедлива, вычисляется вероятность получить такое же значение статистики в генеральной совокупности, что и в выборке. Если вероятность достаточно мала, то нулевая гипотеза отвергается в пользу противоположной (ее называют *альтернативной*, или *исследовательской*, *гипотезой*, H_1).

Этот процесс проще понять на примере. Допустим, перед вами была поставлена задача выяснить, как влияют разговоры по мобильному телефону на время реакции водителя. Вы выдвинули следующую нулевую гипотезу H_0: $\mu_1 - \mu_2 = 0$, где μ_1 – среднее время реакции водителей, разговаривающих по мобильному телефону, а μ_2 – среднее время реакции водителей, не разговаривающих по мобильному телефону (в данном случае интересующим вас параметром генеральной совокупности является разность $\mu_1 - \mu_2$). Если нулевая гипотеза отвергается, то остается альтернативная H_1: $\mu_1 - \mu_2 \neq 0$. Это означает то же, что $\mu_1 \neq \mu_2$, то есть среднее время реакции при разных условиях не одинаково.

Затем вы отобрали участников эксперимента и случайным образом разделили их на две группы. Водители в первой группе должны были, работая на тренажере, реагировать на ряд сложных дорожных ситуаций, разговаривая по мобильному телефону. Водители во второй группе – выполнять те же задания, но уже без мобильного телефона. Для каждого водителя фиксировалось общее время реакции.

На основании выборочных данных можно вычислить статистику $(\bar{X}_1 - \bar{X}_2)/(s - \sqrt{n})$, где $\bar{X}_1$ и $\bar{X}_2$ – средние значения времени реакции для каждой группы, s – стандартное отклонение для объединенной выборки, а n – число участников в каждой группе. Если

нулевая гипотеза верна и можно утверждать, что значения времени реакции нормально распределены, то значения выборочной статистики будут подчиняться распределению Стьюдента с $2n - 2$ степенями свободы. Используя этот факт, можно вычислить вероятность того, что выборочная статистика будет не меньше значения этой статистики для генеральной совокупности. Если вероятность не достигает некоторого принятого критического значения (скажем, `p < 0.05`), то нулевая гипотеза отвергается в пользу альтернативной. Заранее определенное критическое значение (`0.05`) называется *уровнем значимости* критерия.

Обратите внимание: чтобы сделать выводы о генеральной совокупности, мы используем *выборочные* данные, полученные из этой совокупности. В данном случае нулевая гипотеза утверждает, что среднее время реакции *всех* водителей, говорящих по мобильному телефону, не отличается от среднего времени реакции *всех* водителей, не говорящих по нему. Здесь не идет речь только о водителях из выборки. Принятое решение может иметь четыре возможных последствия:

- если нулевая гипотеза ложна и была отвергнута в соответствии со статистическим критерием, то мы приняли верное решение; мы верно заключили, что использование мобильного телефона влияет на время реакции;
- если нулевая гипотеза верна и не была отвергнута, то мы снова приняли верное решение, потому что время реакции действительно не зависит от использования мобильного телефона;
- если нулевая гипотеза верна и была отвергнута, мы совершили статистическую ошибку первого рода, решив, что использование мобильного телефона влияет на время реакции, хотя это не так;
- если нулевая гипотеза ложна и не была отвергнута, мы совершили статистическую ошибку второго рода; мобильные телефоны влияют на время реакции, но мы не смогли это выявить.

Все эти случаи сведены в табл. 10.1.

Таблица 10.1. Проверка гипотезы

		Решение	
		Отвергнуть H_0	**Принять H_0**
В действительности	**H_0 верна**	**Ошибка первого рода**	Верное решение
	H_0 ложна	Верное решение	**Ошибка второго рода**

Полемика вокруг проверки значимости нулевой гипотезы

Подход, основанный на проверке значимости нулевой гипотезы, продолжает вызывать споры. Скептики выдвигают многочисленные аргументы против применения этого подхода, особенно в психологии. Они указывают на непонимание значений p широкими массами, опору на статистическую значимость, а не значимость, вытекающую из здравого смысла, на тот факт, что нулевая гипотеза никогда не выполняется абсолютно точно и будет обязательно отвергнута при достаточно большом объеме выборки, а также на целый ряд логических неувязок, возникающих при проверке значимости нулевой гипотезы.

Подробное обсуждение этой темы выходит за рамки данной книги. Те, кому это интересно, могут обратиться к публикации Harlow, Mulaik, Steiger (1997).

При планировании исследования ученый обычно обращает особое внимание на четыре величины (рис. 10.1):

размер выборки – число наблюдений при каждом типе воздействия или в каждой группе;

уровень значимости (часто обозначаемый как *альфа*) – вероятность совершения статистической ошибки первого рода. Уровень значимости можно также трактовать как вероятность нахождения несуществующей закономерности;

мощность (вероятность совершения статистической ошибки второго рода). Мощность можно также понимать как вероятность не обнаружить *существующую* закономерность;

размер эффекта – величина эффекта, который является предметом альтернативной гипотезы. Формула для определения размера эффекта зависит от статистической методологии, используемой при проверке гипотезы.

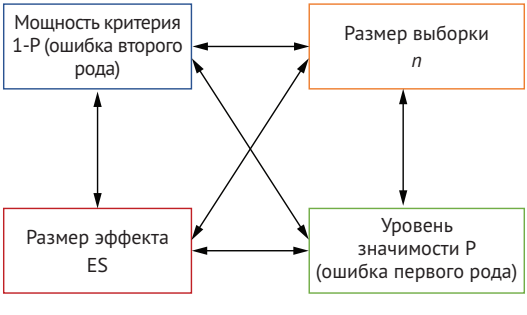


Рис. 10.1. Четыре основные величины, которые рассматриваются при анализе мощности. Зная любые три, можно вычислить четвертую

Размер выборки и уровень значимости находятся под непосредственным контролем исследователя, но мощность и размер эффекта определить сложнее. Например, если увеличить критическое значение уровня значимости (другими словами, повысить вероят-

ность отказа от нулевой гипотезы), то мощность возрастает. Точно так же мощность увеличивается с увеличением размера выборки.

Обычно задача исследователя состоит в том, чтобы повысить мощность статистических критериев, сохранив приемлемый уровень значимости при минимально возможной выборке, т. е. увеличить шансы обнаружения существующей закономерности и свести к минимуму вероятность обнаружить отсутствующую закономерность при разумном бюджете.

Эти четыре величины (размер выборки, уровень значимости, мощность и величина эффекта) тесно взаимосвязаны. *Зная значения любых трех величин, можно определить четвертую.* Мы используем данный факт при выполнении различных анализов мощности на протяжении всей главы. В следующем разделе мы познакомимся со способами проведения анализа мощности с использованием пакета `pwr`. Позже мы кратко рассмотрим некоторые узкоспециализированные функции, которые используются в биологии и генетике.

10.2. Проведение анализа мощности при помощи пакета pwr

Пакет `pwr`, созданный Стефаном Чемпили (Stéphane Champely), позволяет выполнить анализ мощности по алгоритмам, описанным в публикации Cohen (1988). В табл. 10.2 перечислены некоторые важные функции из этого пакета. При вызове каждой функции можно указать три из четырех величин (размер выборки, уровень значимости, мощность и размер эффекта), а четвертая будет вычислена автоматически.

Таблица 10.2. Функции из пакета pwr

Функция	Вычисляет мощность для
`pwr.2p.test()`	Двух пропорций (равные объемы выборок)
`pwr.2p2n.test()`	Двух пропорций (неодинаковые объемы выборок)
`pwr.anova.test()`	Сбалансированного однофакторного дисперсионного анализа
`pwr.chisq.test()`	Критерия хи-квадрат
`pwr.f2.test()`	Общей линейной модели
`pwr.p.test()`	Пропорции (одна выборка)
`pwr.r.test()`	Корреляции
`pwr.t.test()`	Критерия Стьюдента (одна выборка, две выборки, для зависимых переменных)
`pwr.t2n.test()`	Критерия Стьюдента (две выборки разного объема)

Труднее всего из четырех величин вычислить размер эффекта. Для этого обычно требуется некоторый опыт работы с использу-

емыми метриками и знание результатов прошлых исследований. Но что делать, если вы не имеете ни малейшего представления об ожидаемой величине размера эффекта в конкретном исследовании? Мы ответим на этот сложный вопрос в разделе 10.2.7, а в оставшейся части этого раздела рассмотрим применение функций из пакета `pwr` к обычным статистическим критериям. Перед вызовом этих функций обязательно установите и загрузите пакет `pwr`.

10.2.1. Критерий Стьюдента

Для случаев, когда требуется использовать критерий Стьюдента, функция `pwr.t.test()` предоставляет много полезных возможностей. Она имеет следующий синтаксис:

```
pwr.t.test(n=, d=, sig.level=, power=, type=, alternative=)
```

где:

- `n` – размер выборки;
- `d` – величина размера эффекта, выраженная в стандартизированной разнице средних значений, вычисляемой по формуле:

$$d = \frac{|\mu_1 - \mu_2|}{\sigma},$$

  где μ_1 – среднее значение для первой группы,
  μ_2 – среднее значение для второй группы,
  σ – дисперсия общей ошибки;
- `sig.level` – уровень значимости (по умолчанию `0.05`);
- `power` – мощность;
- `type` – тип критерия: для двух выборок (`"two.sample"`, по умолчанию), для одной выборки (`"one.sample"`), для зависимых выборок (`"paired"`);
- `alternative` – двухсторонний (`"two.sided"`, по умолчанию) или односторонний (`"less"` или `"greater"`).

Рассмотрим применение этой функции на примере. Продолжим обсуждение эксперимента с мобильным телефоном и скоростью реакции из раздела 10.1 и допустим, что вы используете двухсторонний критерий Стьюдента для независимых выборок, чтобы сравнить среднее время реакции для водителей, разговаривающих по мобильному телефону, и водителей, выполняющих задание без воздействия отвлекающих факторов.

Предположим, что вам известны результаты предыдущих исследований, где стандартное отклонение времени реакции составляет 1,25 с. Также предположим, что различие во времени реакции в 1 с считается существенным. Таким образом, в проводимом исследовании ожидается обнаружить эффект величиной `d = 1/1.25 = 0.8` или больше. В дополнение к этому вы хотите быть уверенными на

90 % в том, что действительно обнаружите различия, если они существуют, и на 95 %, что не назовете значимым различие, которое на самом деле является результатом случайных флуктуаций. Сколько человек вам нужно обследовать?

Передав эту информацию функции `pwr.t.test()`, вы получите:

```
> library(pwr)
> pwr.t.test(d=.8, sig.level=.05, power=.9, type="two.sample",
             alternative="two.sided")

     Two-sample t test power calculation

              n = 34
              d = 0.8
      sig.level = 0.05
          power = 0.9
    alternative = two.sided

 NOTE: n is number in *each* group
```

Судя по результатам, чтобы определить величину эффекта 0,8 с достоверностью 90 % и вероятностью ошибочного вывода о наличии разницы не более 5 %, когда на самом деле это не так, вам понадобится отобрать группы по 34 в каждой (всего 68 участников).

Изменим вопрос. Предположим, что при сравнении вы хотите обнаружить разницу между средними значениями генеральной совокупности, составляющую 0,5 стандартного отклонения; ограничить вероятность ложного объявления отличающимися средними значениями генеральной совокупности до 0,01; а кроме того, можете позволить себе включить в исследование только 40 участников. Какова вероятность, что при таких ограничениях вы обнаружите такую большую разницу между средними значениями генеральной совокупности?

Предполагая, что в обеих группах будет равное количество участников, вы получите:

```
> pwr.t.test(n=20, d=.5, sig.level=.01, type="two.sample",
             alternative="two.sided")

       Two-sample t test power calculation

              n = 20
              d = 0.5
      sig.level = 0.01
          power = 0.14
    alternative = two.sided

 NOTE: n is number in *each* group
```

При 20 участниках в каждой группе, заданном уровне значимости 0,01 и стандартном отклонении зависимой величины в 1,25 с вероятность того, что разница в 0,625 с (`d = 0.5 = 0.625/1.25`) или меньше будет значимой, составляет 14 %. Соответственно, вероят-

ность пропустить искомый эффект составляет 86 %. Возможно, вам нужно еще раз серьезно подумать, стоит ли тратить время и силы на решение этой задачи в том виде, в каком она сформулирована.

В предыдущих примерах подразумевалось, что размеры групп одинаковы. Если размер выборок различается, можно использовать функцию:

```
pwr.t2n.test(n1=, n2=, d=, sig.level=, power=, alternative=)
```

где `n1` и `n2` – размеры выборок, а назначение остальных параметров совпадает с назначением параметров функции `pwr.t.test()`. Попробуйте вызывать `pwr.t2n()` с разными значениями параметров и посмотрите, что получится.

10.2.2. Дисперсионный анализ

Функция `pwr.anova.test()` позволяет провести анализ мощности для сбалансированного однофакторного дисперсионного анализа. Она имеет следующий синтаксис:

```
pwr.anova.test(k=, n=, f=, sig.level=, power=)
```

где `k` – число групп, а `n` – размер выборки в каждой группе.

Для однофакторного дисперсионного анализа величина эффекта отражена в параметре `f`, который вычисляется по формуле

$$f = \sqrt{\frac{\sum_{i=1}^{k} p_i \times (\mu_1 - \mu_2)}{\sigma^2}},$$

где $p_i = n_i / N$,
n_i – число наблюдений в группе i,
N – общее число наблюдений,
μ_i – среднее значение для группы i,
σ^2 – дисперсия ошибок внутри групп.

Попробуем разобрать пример. Для однофакторного дисперсионного анализа, включающего пять групп, нужно найти размер групп, при котором мощность составит 0,8, размер эффекта – 0,25 и уровень значимости – 0,05. Решение выглядит так:

```
> pwr.anova.test(k=5, f=.25, sig.level=.05, power=.8)

     Balanced one-way analysis of variance power calculation

              k = 5
              n = 39
              f = 0.25
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

Общий размер выборки должен быть не менее 5×39, или 195. Учтите, что этот пример требует оценить средние значения для групп и общую дисперсию. Если у вас нет ни малейшего представления об этих значениях, вам могут пригодиться подходы, описанные в разделе 10.2.7.

10.2.3. Корреляции

Функция `pwr.r.test()` позволяет провести анализ мощности для коэффициентов корреляции. Она имеет следующий синтаксис:

```
pwr.r.test(n=, r=, sig.level=, power=, alternative=)
```

где `n` – число наблюдений, `r` – размер эффекта (величина линейного коэффициента корреляции), `sig.level` – уровень значимости, `power` – уровень мощности, а параметр `alternative` определяет тип критерия: двухсторонний (`"two.sided"`) или односторонний (`"less"` или `"greater"`).

Например, допустим, что мы изучаем связь между депрессией и одиночеством и выдвигаем следующие гипотезы:

$$H_0: \rho \leq 0{,}25 \text{ и } H_1: \rho > 0{,}25,$$

где ρ – коэффициент корреляции между этими двумя психологическими параметрами в генеральной совокупности. Мы устанавливаем уровень значимости равным 0,05 и хотим быть уверены на 90 %, что нулевая гипотеза будет отвергнута, если она ложна для генеральной совокупности. Сколько наблюдений нужно сделать? Ответ дает следующий программный код:

```
> pwr.r.test(r=.25, sig.level=.05, power=.90, alternative="greater")

     approximate correlation power calculation (arctangh transformation)

              n = 134
              r = 0.25
      sig.level = 0.05
          power = 0.9
    alternative = greater
```

То есть нам нужно исследовать 134 человека, чтобы быть на 90 % уверенными в том, что нулевая гипотеза будет отвергнута, если она действительно окажется ложной.

10.2.4. Линейные модели

Для анализа мощности линейных моделей (таких как множественная регрессия) можно использовать функцию `pwr.f2.test()`. Она имеет следующий синтаксис:

```
pwr.f2.test(u=, v=, f2=, sig.level=, power=)
```

где u и v – это числитель и знаменатель степеней свободы, а f2 – размер эффекта, который вычисляется по формулам:

$$f^2 = \frac{R^2}{1-R^2},$$

где R^2 – квадрат множественного коэффициента корреляции для генеральной совокупности, и

$$f^2 = \frac{R^2_{AB} - R^2_A}{1-R^2_{AB}},$$

где R^2_A – дисперсия в генеральной совокупности, объясняемая переменными из множества *A*,
R^2_{AB} – дисперсия в генеральной совокупности, объясняемая переменными из множеств *A* и *B*.

Первая формула подходит для оценки влияния набора независимых переменных на одну зависимую. Вторая формула применяется, когда требуется выявить влияние одного набора независимых переменных наряду с влиянием другого набора независимых переменных (или ковариат).

Скажем, вы интересуетесь, влияет ли стиль руководства на удовлетворенность сотрудников своей работой наряду с уровнем зарплаты и чаевыми. Стиль руководства описан четырьмя переменными, а размер зарплаты и чаевых – тремя. На основании имеющегося опыта можно сказать, что удовлетворенность работой примерно на 30 % определяется зарплатой. С практической точки зрения интересно, добавит ли стиль руководства хотя бы 5 % к этому значению. Сколько людей нужно опросить, чтобы выявить такой размер эффекта с 90 % уверенности при уровне значимости 0,05?

В данном случае `sig.level=0.05`, `power=0.90`, `u=3` (число независимых переменных за вычетом независимых переменных во втором наборе), а размер эффекта составит `f2 = (.35-.30)/(1-.35) = 0.0769`. Передав эти значения функции, получаем:

```
> pwr.f2.test(u=3, f2=0.0769, sig.level=0.05, power=0.90)

     Multiple regression power calculation

              u = 3
              v = 184.2426
             f2 = 0.0769
      sig.level = 0.05
          power = 0.9
```

В случае множественной регрессии знаменатель степеней свободы равен $N - k - 1$, где N – число наблюдений, а k – число независимых переменных. В данном случае $N - 7 - 1 = 185$, а необходимый размер выборки составит $N = 185 + 7 + 1 = 193$.

10.2.5. Сравнение пропорций

Функцию `pwr.2p.test()` можно применять для анализа мощности при сравнении двух пропорций. Она имеет следующий синтаксис:

```
pwr.2p.test(h=, n=, sig.level=, power=)
```

где `h` – величина эффекта, а `n` – общий объем выборки для двух групп. Величина эффекта рассчитывается по формуле

$$h = 2\arcsin\left(\sqrt{p_1}\right) - 2\arcsin\left(\sqrt{p_2}\right)$$

и может быть вычислена при помощи функции `ES.h(p1, p2)`[1].

Для выборок разного объема можно использовать функцию

```
pwr.2p2n.test(h =, n1 =, n2 =, sig.level=, power=)
```

В параметре `alternative=` передается тип критерия: двухсторонний (`"two.sided"`, по умолчанию) или односторонний (`"less"` или `"greater"`).

Допустим, ожидается, что обычное лекарство облегчает симптомы болезни у 60 % пациентов. Новое (и более дорогое) средство будет иметь успех на рынке, если оно улучшит состояние 65 % больных. Сколько испытуемых нужно исследовать, чтобы обнаружить указанное различие между лекарствами по эффективности?

Предположим, вы хотите быть на 90 % уверенными, что новое лекарство зарекомендовало себя лучше, и с 95%-ной вероятностью прийти к этому заключению не по ошибке. Для проверки следует использовать односторонний критерий, потому что требуется проверить лишь наличие превосходства нового лекарства над обычным. Вот как выглядит программный код, решающий эту задачу:

```
> pwr.2p.test(h=ES.h(.65, .6), sig.level=.05, power=.9,
              alternative="greater")

     Difference of proportion power calculation for binomial
     distribution (arcsine transformation)

              h = 0.1033347
              n = 1604.007
      sig.level = 0.05
          power = 0.9
    alternative = greater

 NOTE: same sample sizes
```

Судя по полученным результатам, вам нужно исследовать 1605 человек, принимающих новый препарат, и 1605 человек, по-

[1] Параметры `p1` и `p2` соответствуют сравниваемым пропорциям. – *Прим. перев.*

лучающих обычное лекарство, чтобы проверить гипотезу при заданных условиях.

10.2.6. Критерий хи-квадрат

Критерий хи-квадрат часто используется для оценки связи между двумя категориальными переменными. Роль нулевой гипотезы обычно играет предположение о независимости переменных, а роль альтернативной гипотезы – предположение об их зависимости. При вычислении критерия хи-квадрат для оценки мощности, размера эффекта или требуемого объема выборки можно использовать функцию `pwr.chisq.test()`. Она имеет следующий синтаксис:

```
pwr.chisq.test(w =, N = , df = , sig.level =, power = )
```

где `w` – размер эффекта, `N` – общий объем выборки, а `df` – число степеней свободы. В данном случае размер эффекта вычисляется по формуле

$$w = \sqrt{\sum_{i=1}^{m} \frac{\left(p0_i - p1_i\right)^2}{p0_i}},$$

где $p0_i$ – вероятность в ячейке таблицы сопряженности при справедливой H_0,
$p1_i$ – вероятность в ячейке таблицы сопряженности при справедливой H_1.

Суммирование происходит от 1 до m, где m – число ячеек в таблице сопряженности. Вычисление размера эффекта, соответствующего альтернативной гипотезе для двумерной таблицы сопряженности, можно выполнить с помощью функции `ES.w2(P)`, где `P` – двухмерная таблица вероятностей.

В качестве простого примера предположим, что вы ищете связь между этнической принадлежностью и продвижением по службе. Вы ожидаете, что 70 % выборки будет представлено белыми, 10 % – афроамериканцами и 20 % – латиноамериканцами. Далее вы предполагаете, что 60 % белых продвинутся по службе, для афроамериканцев эта величина составит 30 %, а для латиноамериканцев – 50 %. Альтернативная гипотеза состоит в том, что вероятность продвижения по службе будет соответствовать значениям, представленным в табл. 10.3.

Таблица 10.3. Предполагаемая доля людей, которые продвинутся по службе, в соответствии с альтернативной гипотезой

Этническая принадлежность	Продвинулись по службе	Не продвинулись
Белые	0,42	0,28
Афроамериканцы	0,03	0,07
Латиноамериканцы	0,10	0,10

К примеру, вы ожидаете, что 42 % генеральной совокупности будет представлено получившими повышение белыми (0,42 = 0,10 × 0,60), а 7 % генеральной совокупности – это непродвинувшиеся по службе афроамериканцы (0,07 = 0,10 × 0,70). Примем уровень значимости равным 0,05 и уровень мощности 0,90. Число степеней свободы в двухмерной таблице сопряженности рассчитывается как $(r-1) \times (c-1)$, где r – это число строк, а c – число столбцов. Следующий программный код вычисляет ожидаемый размер эффекта:

```
> prob <- matrix(c(.42, .28, .03, .07, .10, .10), byrow=TRUE, nrow=3)
> ES.w2(prob)

[1] 0.1853198
```

Используя эту информацию, можно определить необходимый размер выборки:

```
> pwr.chisq.test(w=.1853, df=2, sig.level=.05, power=.9)

     Chi squared power calculation

              w = 0.1853
              N = 368.5317
             df = 2
      sig.level = 0.05
          power = 0.9

 NOTE: N is the number of observations
```

Полученный результат свидетельствует о том, что для обнаружения взаимосвязи между этнической принадлежностью и продвижением по службе при заданных размере эффекта, мощности и уровне значимости достаточно 369 человек.

10.2.7. Выбор размера эффекта в незнакомых ситуациях

Размер эффекта – это тот параметр, который сложнее всего определить при анализе мощности. Обычно для этого нужно иметь опыт работы с объектом исследования и с используемыми метриками. Например, для вычисления размера эффектов используются данные предыдущих исследований, а полученные результаты затем применяются для планирования новых работ.

Но что делать в совершенно новой ситуации, когда нет прошлого опыта, к которому можно было бы обратиться? В области бихевиоризма (науки о поведении) Cohen (1988) предложил ориентировочные значения для «малых», «средних» и «больших» размеров эффектов для разных статистических критериев. Они приведены в табл. 10.4.

Таблица 10.4. Ориентировочные значения размеров эффектов, предложенные Коэном (Cohen)

Статистический метод	Метрика размера эффекта	Предлагаемые ориентировочные значения для разных размеров эффектов		
		Малый	Средний	Большой
Критерий Стьюдента	d	0,20	0,50	0,80
Дисперсионный анализ	f	0,10	0,25	0,40
Линейные модели	f2	0,02	0,15	0,35
Критерий пропорций	h	0,20	0,50	0,80
Критерий хи-квадрат	w	0,10	0,30	0,50

Если у вас нет ни малейшего представления о возможном размере эффекта, можно руководствоваться этой таблицей. Например, определим вероятность отвергнуть ложную нулевую гипотезу (то есть найти действительно имеющуюся закономерность) при использовании однофакторного дисперсионного анализа с пятью группами, 25 объектами в каждой группе и уровне значимости 0,05.

Используя функцию `pwr.anova.test()` и значение `f` из табл. 10.3:

```
pwr.anova.test(k=5, n=25, sig.level=0.05, f=c(.10, .25, .40))

     Balanced one-way analysis of variance power calculation
              k = 5
              n = 25
              f = 0.10, 0.25, 0.40
      sig.level = 0.05
          power = 0.1180955, 0.5738000, 0.9569163

       NOTE: n is number in each group
```

выясняем, что для малого эффекта мощность будет `0.118`, для умеренного эффекта – `0.574`, а для большого – `0.957`. Учитывая ограниченность размера выборки, наиболее вероятно, что вы обнаружите эффект, только если он будет большим.

Важно помнить, что предложенные Коэном ориентировочные значения – это только общие идеи, сформулированные на основании ряда социологических исследований, и они могут быть неприменимыми в вашей области исследования. Альтернативный подход: изменение исследуемых параметров и наблюдение за тем, как это отразится, например, на размере выборки и мощности. Для примера предположим, что нам опять нужно сравнить пять групп методом однофакторного дисперсионного анализа с уровнем значимости 0,05. В листинге 10.1 показано, как определить объем выборок, необходимых для обнаружения эффектов разной величины, и графически отобразить результаты (рис. 10.2).

Листинг 10.1. Размеры выборок для обнаружения статистически значимых эффектов в однофакторном дисперсионном анализе

```
library(pwr)
es <- seq(.1, .5, .01)
nes <- length(es)

samsize <- NULL
for (i in 1:nes){
    result <- pwr.anova.test(k=5, f=es[i], sig.level=.05, power=.9)
    samsize[i] <- ceiling(result$n)
}

plotdata <- data.frame(es, samsize)
library(ggplot2)
ggplot(plotdata, aes(x=samsize, y=es)) +
  geom_line(color="red", size=1) +
  theme_bw() +
  labs(title="One Way ANOVA (5 groups)",
       subtitle="Power = 0.90,  Alpha = 0.05",
       x="Sample Size (per group)",
       y="Effect Size")
```

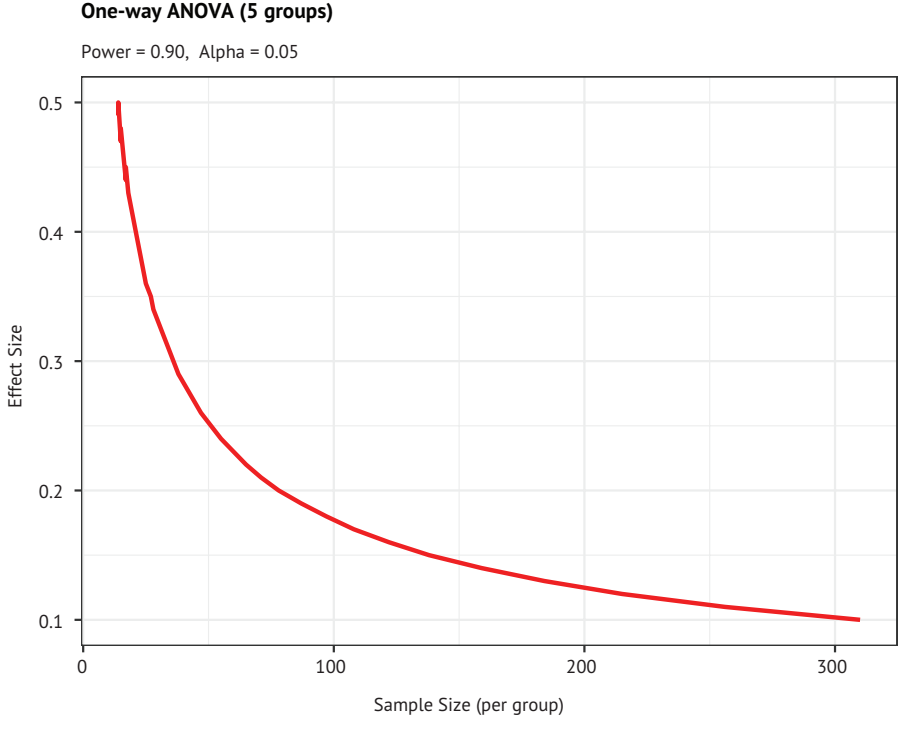


Рис. 10.2. Размеры выборок, необходимые для обнаружения эффектов разной величины в однофакторном дисперсионном анализе с пятью группами (при заданной мощности 0,9 и уровне значимости 0,05)

Диаграммы вроде той, что представлена на рис. 10.2, могут пригодиться для оценки разных параметров эксперимента. Например, похоже, что вы не получите много дополнительных преимуществ, создав выборку, насчитывающую более 200 наблюдений. В следующем разделе мы увидим другой пример графического представления результатов.

10.3. Графический анализ мощности

Прежде чем расстаться с пакетом `pwr`, рассмотрим более сложный пример графического представления результатов. Предположим, вам нужно узнать размер выборки, необходимый для обнаружения значимого коэффициента корреляции, при разных размерах эффекта и уровнях значимости. Для решения этой задачи можно использовать функцию `pwr.r.test()` и цикл `for`, как показано в листинге 10.2.

Листинг 10.2. Графическое изображение зависимости объема выборки от искомой величины корреляции

```
library(pwr)
r <- seq(.1,.5,.01)                                                    1
p <- seq(.4,.9,.1)                                                     1

df <- expand.grid(r, p)
colnames(df) <- c("r", "p")

for (i in 1:nrow(df)){                                                 2
    result <- pwr.r.test(r = df$r[i],                                  2
                         sig.level = .05, power = df$p[i],             2
                         alternative = "two.sided")                    2
    df$n[i] <- ceiling(result$n)                                       2
}                                                                      2

library(ggplot2)                                                       3
ggplot(data=df,                                                        3
       aes(x=r, y=n, color=factor(p))) +                               3
  geom_line(size=1) +                                                  3
  theme_bw() +                                                         3
  labs(title="Sample Size Estimation for Correlation Studies",         3
       subtitle="Sig=0.05 (Two-tailed)",                               3
       x="Correlation Coefficient (r)",                                3
       y="Samsple Size (n)",                                           3
       color="Power")                                                  3
```

1. **Определение диапазона значений мощности и коэффициентов корреляции.**
2. **Определение размеров выборок.**
3. **Вывод кривых мощности.**

В листинге 10.2 для создания ряда значений размера эффекта `r` (коэффициентов корреляции при H_1) и уровней мощности `p` (1) использована функция `seq()`. Функция `expand.grid()` создает таблицу данных со всеми комбинациями этих двух переменных. Далее цикл `for` перебирает все эти размеры эффектов и уровни мощности и вычисляет необходимые объемы выборок, сохраняя их в результате `result` (2). Затем пакет `ggplot2` создает кривые значений мощности (3). Полученный график представлен на рис. 10.3. Если вы читаете эту книгу в черно-белом варианте, то цвета линий могут быть трудно различимы. Поэтому уточню, что линии представляют мощности (снизу вверх) 0,4, 0,5, 0,6 и т. д., вплоть до 0,9.

Как видите, для обнаружения корреляции с коэффициентом 0,2 с 40%-ной уверенностью требуется выборка объемом примерно в 75 наблюдений. Вам понадобится еще примерно 185 наблюдений (n = 260) для обнаружения корреляции такой же силы с 90%-ной вероятностью. После несложных модификаций такой подход можно использовать для оценки необходимого размера выборки при разной мощности для широкого диапазона статистических критериев.

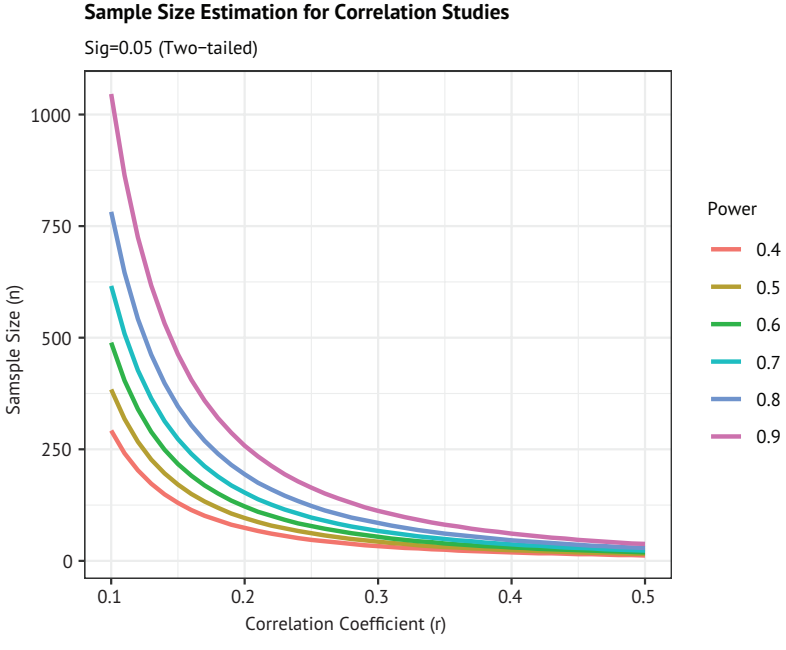


Рис. 10.3. Зависимость объема выборки от искомой величины корреляции при разных уровнях мощности

Мы закончим эту главу кратким обзором других функций, имеющихся в R, которые можно использовать при анализе мощности.

10.4. Другие пакеты

В R имеется множество других пакетов, которые могут пригодиться при планировании исследований. Некоторые из них перечислены в табл. 10.5. В одних реализованы общие подходы, а в других – узкоспециализированные. Исследования геномных ассоциаций (Genome-Wide Association Studies, GWAS) проводятся для определения генетической обусловленности наблюдаемых признаков. К примеру, такие исследования могут быть направлены на выяснение причины определенного типа порока сердца у некоторых людей.

Таблица 10.5. Специализированные пакеты для проведения анализа мощности

Пакет	Описание
`asypow`	Расчет мощности при помощи методов асимптотического отношения правдоподобия
`longpower`	Расчет объема выборки для анализа данных, собираемых за длительные промежутки времени
`PwrGSD`	Анализ мощности для группового последовательного анализа
`pamm`	Анализ мощности для случайных эффектов в смешанных моделях
`powerSurvEpi`	Расчет мощности и объема выборок для анализа выживаемости в эпидемиологических исследованиях
`powerMediation`	Расчет мощности и объема выборки для анализа эффектов опосредования в линейной и логистической регрессиях, в регрессиях Пуассона и Кокса
`semPower`	Анализ мощности для моделей структурных уравнений (Structural Equation Models, SEM)
`powerpkg`	Анализ мощности для пар потомков одних и тех же родителей, на одного из которых оказывается воздействие, и для тестов неравновесной передачи аллелей
`powerGWASinteraction`	Расчет мощности для исследований геномных ассоциаций
`gap`	Функции вычисления мощности и объема выборок для обсервационных исследований (case-cohort design)
`ssize.fdr`	Расчет объема выборок для экспериментов с микрочипами

Наконец, пакеты `MBESS` и `WebPower` предлагают обширный набор функций для различных форм анализа мощности и определения объема выборки. Эти функции особенно важны для исследователей в области поведенческих, образовательных и социальных наук.

Итоги

- Анализ мощности помогает определить объем выборки, необходимый для выявления эффекта требуемого размера с требуемой степенью достоверности. Он также позволяет решить обратную задачу – определить вероятность обнаружения такого эффекта при данном объеме выборки. Как вы могли видеть, анализ мощности предполагает компромисс между ограничением вероятности ошибочного объявления эффекта значимым (ошибка первого рода) и вероятностью правильного определения действительно существующего эффекта (мощность).
- С помощью функций, предлагаемых пакетом `pwr`, можно определять мощность и объем выборки для обычных статистических методов (включая критерии Стьюдента, хи-квадрат и пропорций, дисперсионный анализ и регрессию). В последнем разделе 10.4 было перечислено несколько более специализированных методов.
- Обычно анализ мощности – это интерактивный процесс. Исследователь изменяет параметры – объем выборки, размер эффекта, желаемые уровни значимости и мощности, – чтобы оценить их влияние друг на друга. Результаты используются для планирования исследований, которые с бóльшей вероятностью дадут значимые результаты. Информацию из прошлых исследований (особенно в отношении величины эффекта) тоже можно использовать для планирования более эффективных и результативных исследований.
- Важное побочное преимущество анализа мощности – переход от проверки бинарных гипотез (т. е. существование или отсутствие эффекта) к оценке *размера* рассматриваемого эффекта. Редакторы журналов все чаще требуют, чтобы авторы включали оценки размеров эффектов и *p*-значения в описание результатов исследования. Это помогает читателям определить практическую ценность исследования и получить информацию, которую можно использовать для планирования будущих исследований.

11 Диаграммы средней сложности

В этой главе:

- визуализация взаимосвязей между двумя и более переменными;
- диаграммы рассеяния и линейные графики;
- коррелограммы;
- мозаичные диаграммы.

В главе 6 (базовые диаграммы) мы рассмотрели различные типы диаграмм, отображающих распределение значений одной категориальной или непрерывной переменной. Глава 8 (регрессия) включает обзор графических методов для предсказания непрерывной зависимой переменной по набору независимых. В главе 9 (дисперсионный анализ) мы рассмотрели методы, позволяющие визуализировать межгрупповые различия в значениях непрерывной переменной. Эта глава во многом продолжает и расширяет затронутые раньше темы.

В этой главе мы сосредоточимся на графическом отображении взаимосвязей между двумя (двухмерные взаимосвязи) и несколькими переменными (многомерные взаимосвязи). Например:

- Какова связь между расходом топлива и весом автомобиля? Изменяется ли характер этой связи в зависимости от числа цилиндров?
- Можно ли отразить взаимосвязи между расходом топлива, весом автомобиля, объемом двигателя и передаточным числом заднего моста на одной диаграмме?
- Как решить проблему наложения точек друг на друга, которая часто наблюдается при изображении взаимосвязи между двумя переменными с большим числом (скажем, 10 000) наблюдений? Иными словами, что делать, если диаграмма представляет собой одну большую кляксу?
- Как можно одновременно изобразить взаимосвязи между тремя переменными (учитывая, что мы располагаем двухмерным экраном компьютера или листом бумаги и бюджетом, несколько меньшим, чем бюджет сериала «Звездные войны»)?
- Как изобразить динамику роста нескольких деревьев?
- Как визуально представить корреляции между десятком переменных на одной диаграмме? Как это поможет понять структуру данных?
- Как можно визуализировать взаимоотношения между полом и возрастом пассажиров Титаника, классом, в котором они путешествовали, и вероятностью их выживания? Что можно узнать из такой диаграммы?

На все подобные вопросы можно ответить при помощи описанных в данной главе методов. Наборы данных, которые мы будем использовать, – это лишь примеры. Главное здесь – принципиальные подходы. Если темы характеристик автомобиля или роста деревьев не интересны вам, используйте собственные данные!

Мы начнем с диаграмм рассеяния и составленных из них матриц. Затем исследуем разнообразные графики. Эти подходы хорошо известны и широко используются в научных исследованиях. Потом мы познакомимся с коррелограммами, которые применяются для визуализации корреляций, и мозаичными диаграммами, иллюстрирующими многомерные связи между категориальными переменными. Это тоже полезные инструменты, но не такие известные среди исследователей. Вы увидите примеры применения каждого из этих подходов для лучшего понимания данных и доведения результатов до сведения других.

11.1. Диаграммы рассеяния

Как вы уже знаете, диаграммы рассеяния иллюстрируют взаимосвязь между двумя непрерывными переменными. Мы начнем

этот раздел с изображения взаимосвязи между одной парой переменных (*x* и *y*). Затем исследуем способы совершенствования этой диаграммы путем наложения дополнительной информации. Потом узнаем, как поместить несколько диаграмм рассеяния в общую матрицу, чтобы можно было анализировать несколько двухмерных взаимосвязей одновременно. Также затронем особый случай, когда на диаграмме изображается слишком много точек и они накладываются друг на друга, ограничивая возможности визуального анализа данных, и обсудим несколько способов решения этой проблемы. Наконец, мы расширим двухмерную диаграмму в третье измерение, добавив третью непрерывную переменную. К таким диаграммам относятся трехмерные диаграммы рассеяния и пузырьковые диаграммы. С их помощью можно одновременно исследовать взаимосвязи между тремя переменными.

Начнем с визуализации зависимости между весом автомобиля и расходом топлива (величиной пробега на одном галлоне). Пример представлен в листинге 11.1.

Листинг 11.1. Диаграмма рассеяния с линиями наилучшей аппроксимации

```
data(mtcars)                                                    1
ggplot(mtcars, aes(x=wt, y=mpg)) +  geom_point()                2
  geom_smooth(method="lm", se=FALSE, color="red") +             3
  geom_smooth(method="loess", se=FALSE,                         4
              color="blue", linetype="dashed") +
  labs(title = "Basic Scatter Plot of MPG vs. Weight",          5
       x = "Car Weight (lbs/1000)",
       y = "Miles Per Gallon")
```

1. **Загрузка данных.**
2. **Создание диаграммы рассеяния.**
3. **Добавление аппроксимирующей прямой.**
4. **Добавление аппроксимирующей кривой.**
5. **Добавление подписей.**

Код в листинге 11.1 загружает таблицу данных `mtcars` 1 и создает базовую диаграмму рассеяния, на которой наблюдения обозначены кружками 2. Как и ожидалось, величина пробега на одном галлоне топлива уменьшается с увеличением веса машины, хотя эта связь не идеально линейная. Первый вызов функции `geom_smooth()` добавляет аппроксимирующую прямую (красного цвета) 3. Параметр `se=FALSE` подавляет вывод 95%-ного доверительного интервала вокруг линии. Второй вызов `geom_smooth()` добавляет аппроксимирующую *кривую* (синяя пунктирная линия) 4. Аппроксимирующая кривая – это непа-

раметрическая аппроксимирующая кривая, основанная на локально взвешенной полиномиальной регрессии. Детально этот алгоритм описан в публикации Cleveland (1981). Джош Стармер (Josh Starmer) дает очень интуитивное объяснение таких аппроксимирующих кривых на YouTube (http://www.youtube.com/watch?v=Vf7oJ6z2LCc).

Полученная диаграмма представлена на рис. 11.1.

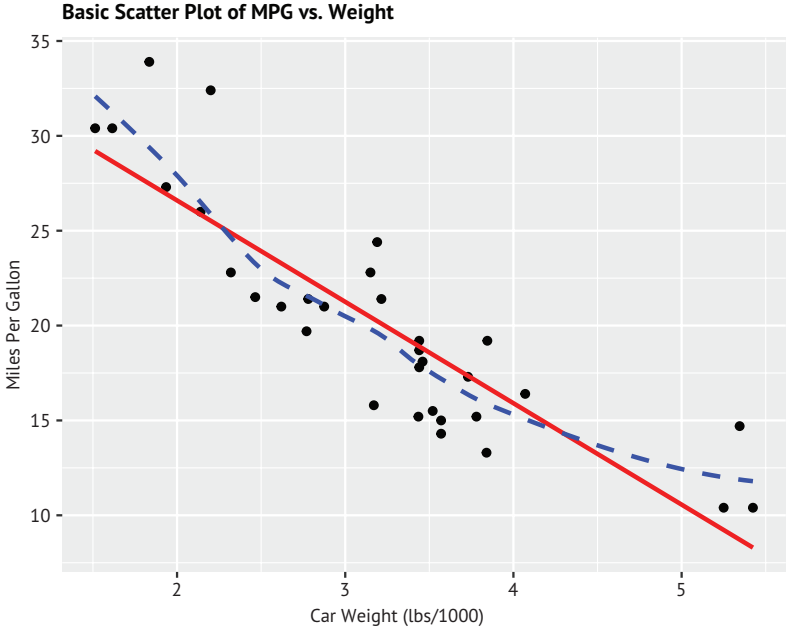


Рис. 11.1. Диаграмма рассеяния для зависимости между величиной пробега на одном галлоне топлива и весом машины с наложенными аппроксимирующими линиями

А как изобразить взаимосвязь между весом и экономичностью отдельно для 4-, 6- и 8-цилиндровых моделей? Это легко сделать с помощью `ggplot2`, добавив нескольких простых изменений в предыдущий код (листинг 11.2). Диаграмма показана на рис. 11.2.

Листинг 11.2. Диаграмма рассеяния с отдельными аппроксимирующими линиями отдельно для 4-, 6- и 8-цилиндровых моделей

```
ggplot(mtcars,
       aes(x=wt, y=mpg,
           color=factor(cyl),
           shape=factor(cyl))) +
  geom_point(size=2) +
  geom_smooth(method="lm", se=FALSE) +
```

```
geom_smooth(method="loess", se=FALSE, linetype="dashed") +
labs(title = "Scatter Plot of MPG vs. Weight",
     subtitle = "By Number of Cylinders",
     x = "Car Weight (lbs/1000)",
     y = "Miles Per Gallon",
     color = "Number of \nCylinders",
     shape = "Number of \nCylinders") +
theme_bw()
```

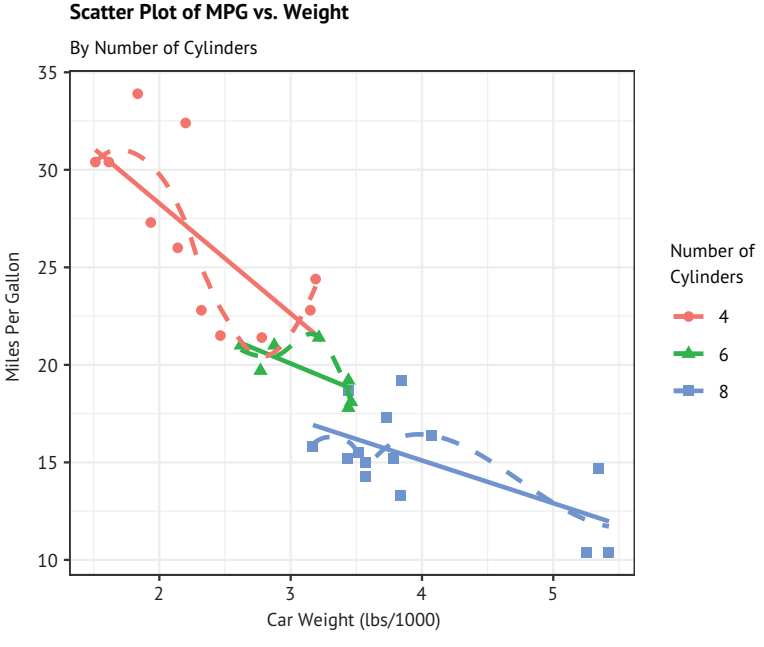


Рис. 11.2. Диаграмма рассеяния с обозначенными подгруппами и аппроксимирующими линиями для каждой подгруппы

Вызовом функции `aes()` группам наблюдений (модели с 4, 6 и 8 цилиндрами) назначаются разные цвета и графические символы, а также обеспечивается вывод отдельных аппроксимирующих линий. Поскольку переменная `cyl` является числовой, ее значения преобразуются в дискретные категории вызовом `factor(cyl)`.

Гладкостью аппроксимирующей кривой можно управлять с помощью параметра `span`. По умолчанию используются параметры `geom_smooth(method="loess", span=0.75)`. Бóльшие значения дают более плавную подгонку. В этом примере аппроксимирующие кривые перекрывают данные (слишком точно повторяют точки). Значение `span=4` (здесь не показано) обеспечивает гораздо более гладкую подгонку.

Диаграммы рассеяния помогают визуализировать взаимосвязи между одной парой количественных переменных одновременно. Но как быть, если необходимо исследовать попарные взаимосвязи между расходом топлива, весом автомобиля, объемом двигателя и передаточным числом заднего моста? Одна из возможностей – объединить эти шесть диаграмм рассеяния в одну матрицу, о которой мы сейчас поговорим.

11.1.1. Матрицы диаграмм рассеяния

В R имеется множество полезных функций для создания матриц диаграмм рассеяния. Стандартная функция `pairs()`, например, создает простые матрицы диаграмм рассеяния. В разделе 8.2.4 (множественная линейная регрессия) был показан пример создания матриц диаграмм рассеяния с использованием функции `scatterplotMatrix()` из пакета `car`.

В этом разделе мы используем функцию `ggpairs()` из пакета `GGally` для создания более совершенной версии матрицы диаграмм рассеяния. Как вы увидите, этот подход обладает широкими возможностями настройки графиков. Прежде чем продолжить, обязательно установите пакет `GGally` (`install.packages("GGally")`).

Для начала создадим простую матрицу диаграмм рассеяния для переменных `mpg`, `disp`, `drat` и `wt` из таблицы данных `mtcars`:

```
library(GGally)
ggpairs(mtcars[c("mpg","disp","drat", "wt")])
```

По умолчанию главная диагональ матрицы содержит кривую ядерной оценки функции плотности для каждой переменной (подробности ищите в разделе 6.5). Графики распределения величины пробега на галлон имеют правую асимметрию (имеется небольшое количество высоких значений), а распределение передаточного числа заднего моста выглядит бимодальным. Ниже главной диагонали расположено шесть диаграмм рассеяния. Диаграмма рассеяния, отражающая зависимость между пробегом в милях на галлон и рабочим объемом двигателя, находится на пересечении двух соответствующих переменных (первый столбец [`mpg`] и вторая строка [`disp`]). Как видите, эта зависимость отрицательная. Выше главной диагонали показаны коэффициенты корреляции Пирсона. Корреляция между пробегом в милях на галлон и объемом двигателя равна –0,848 (первая строка [`mpg`], второй столбец [`disp`]) и подтверждает наш вывод о том, что с увеличением объема двигателя величина пробега на одном галлоне уменьшается (т. е. расход топлива увеличивается).

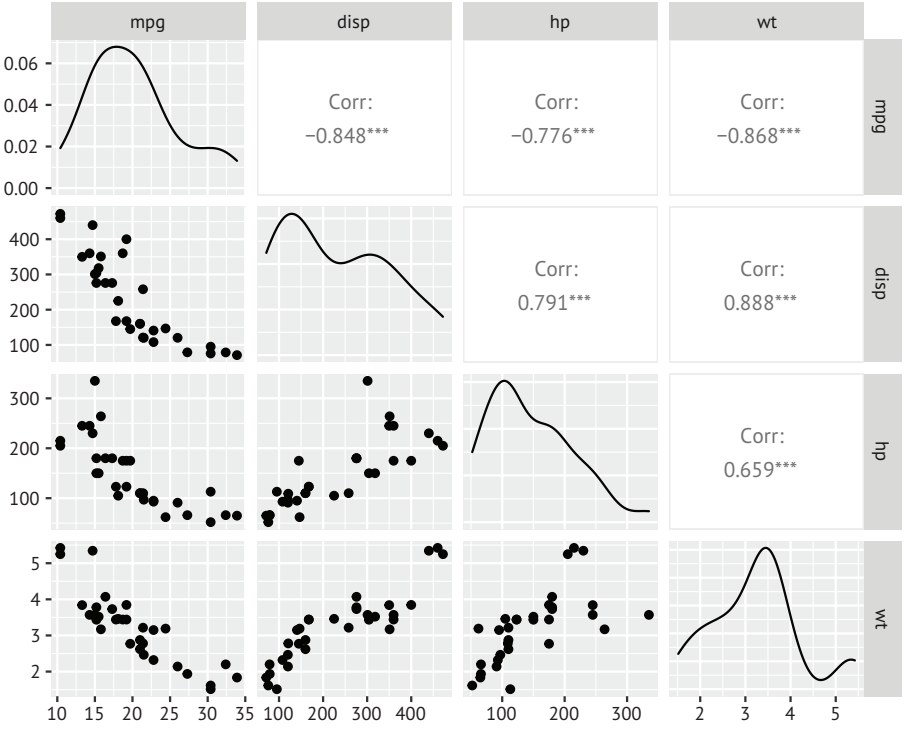


Рис. 11.3. Матрица диаграмм рассеяния, созданная при помощи функции ggpairs()

Далее улучшим матрицу диаграмм рассеяния, добавив аппроксимирующие линии, гистограммы и настроив оформление. Функция `ggpairs()` позволяет указать отдельные функции для создания графиков ниже и выше главной диагонали. Соответствующий код показан в листинге 11.3.

Листинг 11.3. Матрица диаграмм рассеяния с аппроксимирующими линиями, гистограммами и коэффициентами корреляции

```
library(GGally)

diagplots <- function(data, mapping) {                                    1
  ggplot(data = data, mapping = mapping) +                                1
    geom_histogram(fill="lightblue", color="black")                       1
}                                                                         1

lowerplots <- function(data, mapping) {                                   2
    ggplot(data = data, mapping = mapping) +                              2
      geom_point(color="darkgrey") +                                      2
     geom_smooth(method = "lm", color = "steelblue", se=FALSE) +          2
   geom_smooth(method="loess", color="red", se=FALSE, linetype="dashed")  2
}                                                                         2
```

```
upperplots <- function(data, mapping) {                                    3
   ggally_cor(data=data, mapping=mapping,                                  3
              display_grid=FALSE, size=3.5, color="black")                 3
}                                                                          3

mytheme <-  theme(strip.background = element_blank(),                      4
                  panel.grid       = element_blank(),                      4
                  panel.background = element_blank(),                      4
                 panel.border = element_rect(color="grey20", fill=NA))     4


ggpairs(mtcars,                                                            5
       columns=c("mpg","disp", "drat", "wt"),                              5
       columnLabels=c("MPG", "Displacement",                               5
                      "R Axle Ratio", "Weight"),                           5
      title = "Scatterplot Matrix with Linear and Loess Fits",             5
       lower = list(continuous = lowerplots),                              5
       diag =  list(continuous = diagplots),                               5
       upper = list(continuous = upperplots)) +                            5
        mytheme                                                            5
```

1. **Функция для диаграмм на главной диагонали.**
2. **Функция для диаграмм под главной диагональю.**
3. **Функция для диаграмм над главной диагональю.**
4. **Настройка оформления.**
5. **Создание матрицы диаграмм рассеяния.**

Сначала определяется функция для создания гистограммы в виде синих столбиков с черными границами 1. Затем определяется функция для создания диаграммы рассеяния с темно-серыми точками, синей аппроксимирующей прямой и пунктирной красной аппроксимирующей кривой. Доверительные интервалы скрыты (`se=FALSE`) 2. Третья функция отображает коэффициенты корреляции 3. Она использует функцию `ggally_cor()` для получения и вывода коэффициента, ее параметры `size` и `color` определяют внешний вид, а параметр `displayGrid` скрывает линии сетки. Также добавлена настраиваемая тема 4. Этот необязательный шаг устраняет линии сетки в матрице и заключает каждую ячейку в серую рамку.

Наконец, функция `ggpairs()` 5 использует все эти функции для создания диаграммы, изображенной на рис. 11.4. Параметр `columns` задает переменные, а параметр `columnLabels` – описательные имена. Параметры `lower`, `diag` и `upper` определяют функции для создания графиков в ячейках матрицы. Этот подход дает большую гибкость при проектировании диаграмм.

R предоставляет множество других способов создания матриц диаграмм рассеяния. Я также советую познакомиться с функциями `splom()` (в пакете `lattice`), `pairs2()` (в пакете `TeachingDemos`),

`xysplom()` (в пакете `HH`), `kdepairs()` (в пакете `ResourceSelection`) и `pairs.mod()` (в пакете `SMPracticals`). Каждая имеет свои уникальные особенности. Аналитикам должны понравиться матрицы диаграмм рассеяния!

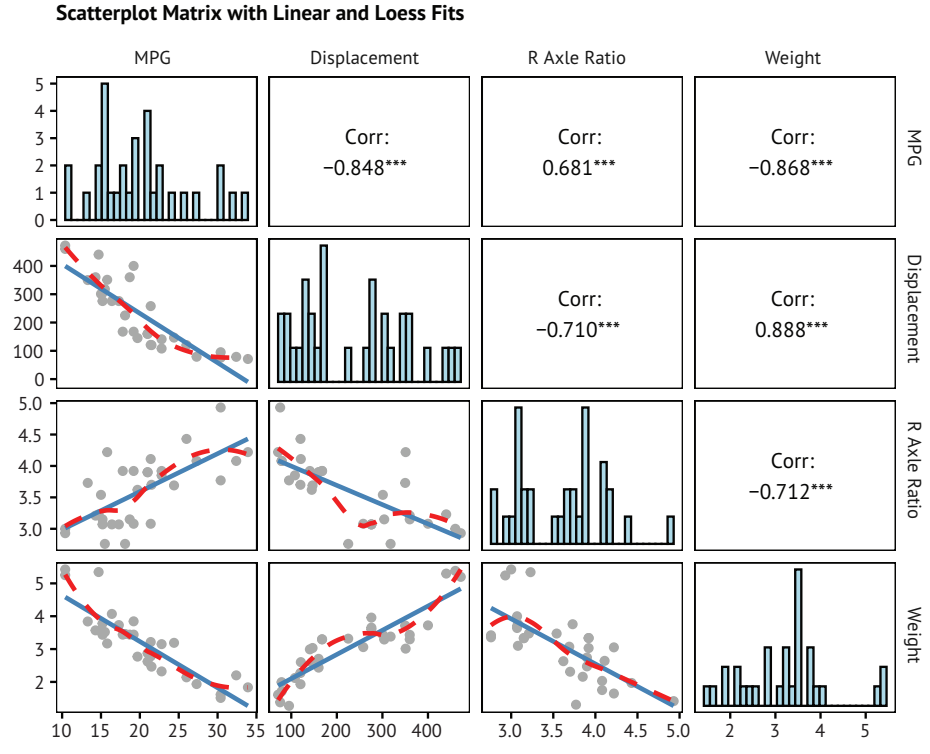


Рис. 11.4. Матрица диаграмм рассеяния, созданная с помощью функции ggpairs() и пользовательских функций, конструирующих графики и гистограммы и отображающих коэффициенты корреляции

11.1.2. Диаграммы рассеяния высокой плотности

На диаграммах рассеяния со значительным наложением точек друг на друга сложнее увидеть взаимосвязи. Рассмотрим такой пример с 10 000 наблюдений, образующих два перекрывающихся кластера:

```
set.seed(1234)
n <- 10000
c1 <- matrix(rnorm(n, mean=0, sd=.5), ncol=2)
c2 <- matrix(rnorm(n, mean=3, sd=2), ncol=2)
mydata <- rbind(c1, c2)
mydata <- as.data.frame(mydata)
names(mydata) <- c("x", "y")
```

Если для этих переменных построить стандартную диаграмму рассеяния при помощи следующего кода:

```
ggplot(mydata, aes(x=x, y=y)) + geom_point() +
  ggtitle("Scatter Plot with 10,000 Observations")
```

то получите диаграмму, изображенную на рис. 11.5.

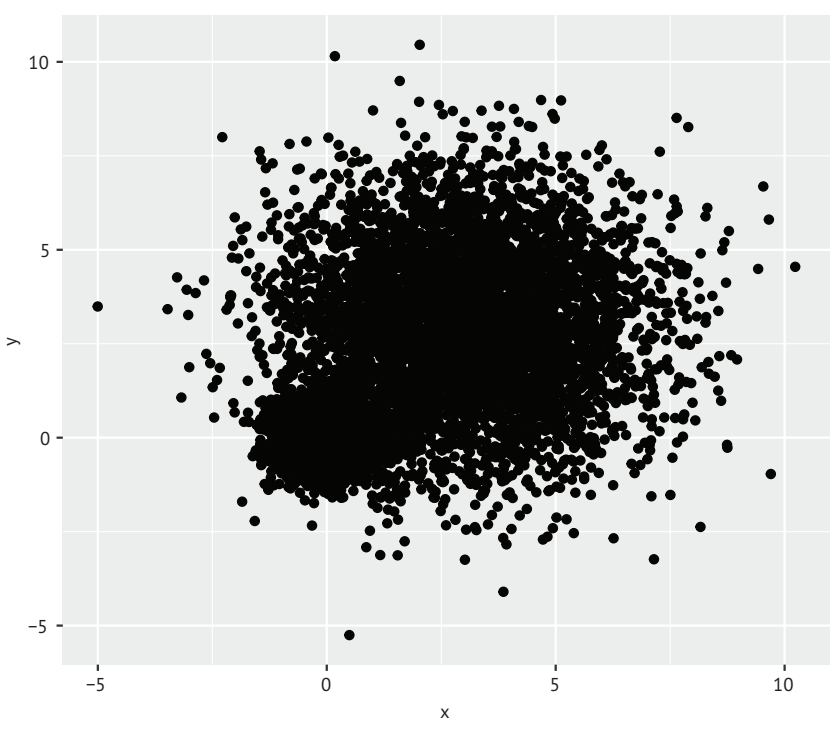


Рис. 11.5. Диаграмма рассеяния для 10 000 наблюдений и с заметным наложением точек друг на друга. Обратите внимание, что наложение точек затрудняет обнаружение области с наивысшей их концентрацией

Наложение точек на рис. 11.5 затрудняет понимание характера взаимосвязи между *x* и *y*. В R реализовано несколько подходов к отображению данных, которые можно использовать в таких случаях. К таким подходам относится объединение точек и использование цвета и прозрачности для обозначения количества наложенных данных в заданной области на диаграмме.

Функция `smoothScatter()` использует ядерную оценку функции плотности для изображения плотности точек на диаграмме при помощи оттенков цвета. Код

```
with(mydata,
     smoothScatter(x, y,
                   main="Scatter Plot Colored by Smoothed Densities"))
```

создаст диаграмму, изображенную на рис. 11.6.

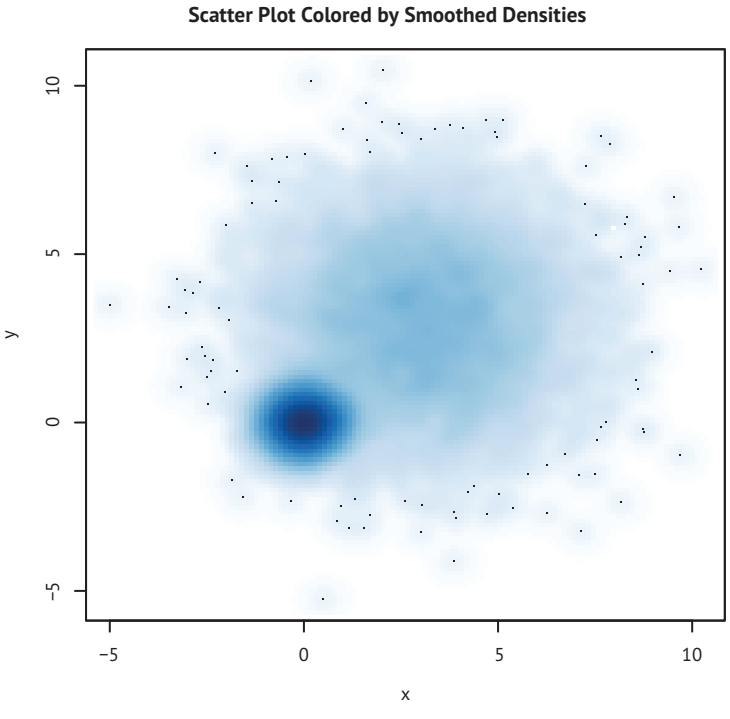


Рис. 11.6. Диаграмма рассеяния, построенная при помощи функции `smoothScatter()` и отражающая сглаженную оценку плотности точек. Плотность точек на графике легко оценить

Другой подход, предлагаемый функцией `geom_hex()` из пакета `ggplot2`, заключается в двухмерном объединении данных в шестиугольные ячейки (это выглядит лучше, чем звучит). По сути, она покрывает область диаграммы сеткой шестиугольных ячеек и отображает количество точек в каждой ячейке с помощью цвета или штриховки. Применив эту функцию к нашему набору данных

```
ggplot(mydata, aes(x=x, y=y)) +
  geom_hex(bins=50) +
  scale_fill_continuous(trans = 'reverse') +
  ggtitle("Scatter Plot with 10,000 Observations")
```

вы получите диаграмму рассеяния, представленную на рис. 11.7.

По умолчанию `geom_hex()` обозначает области с большей плотностью более светлыми цветами. В вашем примере функция `scale_fill_continuous(trans = 'reverse')` выполняет противоположную настройку, заставляя `geom_hex()` отображать области с большей плотностью более темными цветами. Я думаю, что это более соответствует интуитивному пониманию и подходам, используемым в других функциях R для визуализации больших наборов данных.

Обратите внимание, что функцию `hexbin()` из пакета `hexbin` вместе с функцией `iplot()` из пакета `IDPmisc` тоже можно использовать для создания легко интерпретируемых матриц диаграмм рассеяния с большой плотностью. Ищите примеры в `?hexbin` и `?iplot`.

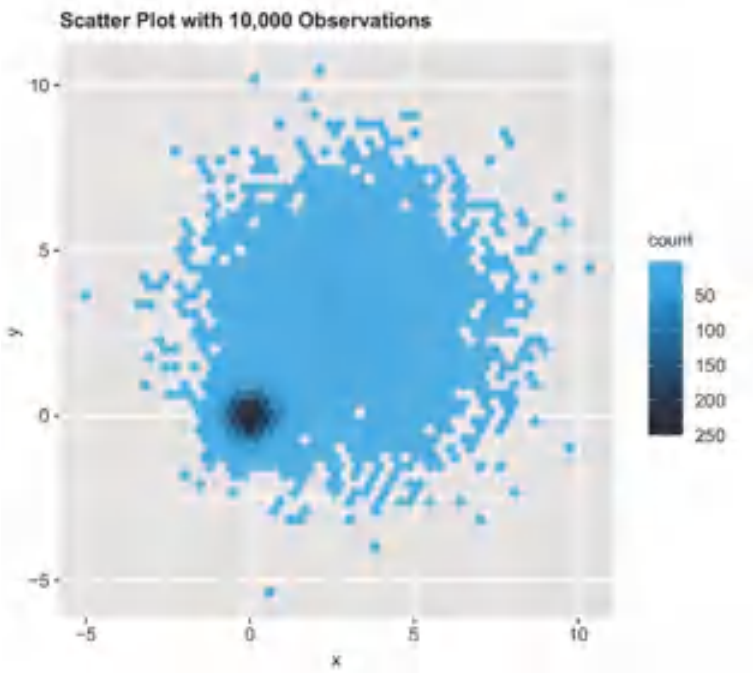


Рис. 11.7. Диаграмма рассеяния с объединением точек в шестиугольные ячейки для наглядного изображения числа наблюдений в каждой точке диаграммы. Скопления данных легко обнаружить, информация об их числе содержится в условных обозначениях

11.1.3. Трехмерные диаграммы рассеяния

Диаграммы рассеяния и матрицы диаграмм рассеяния отображают попарные зависимости, а что, если потребуется визуализировать взаимозависимости между тремя количественными переменными сразу? В этом случае можно использовать трехмерные диаграммы рассеяния.

Например, представьте, что вас заинтересовала взаимосвязь между расходом топлива, объемом двигателя и весом автомобилей. Для графического отображения этой взаимосвязи можно использовать функцию `scatterplot3d()` из пакета `scatterplot3d`. Она имеет следующий синтаксис:

```
scatterplot3d(x, y, z)
```

где *x* – переменная, значения которой откладываются по горизонтали, *y* – по вертикали, а *z* – в перспективе. Следующий код:

```
library(scatterplot3d)
with(mtcars,
     scatterplot3d(wt, disp, mpg,
                   main="Basic 3D Scatter Plot"))
```

выведет трехмерную диаграмму рассеяния, представленную на рис. 11.8.

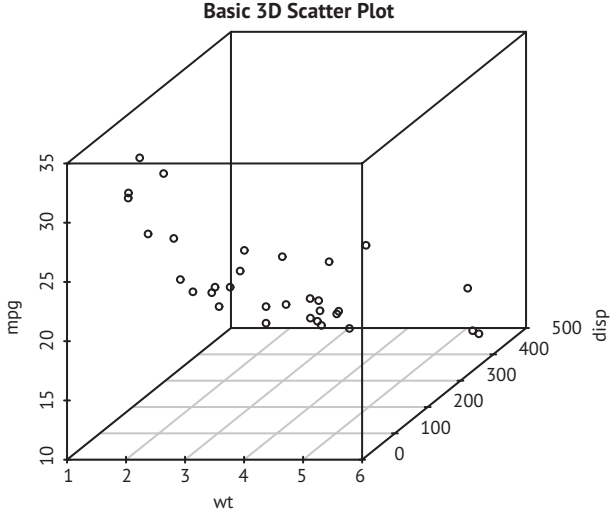


Рис. 11.8. Трехмерная диаграмма рассеяния, иллюстрирующая зависимость между расходом топлива, объемом двигателя и весом автомобилей

Функция `scatterplot3d()` принимает много параметров, позволяющих настраивать отображение символов, осей, линий, координатной сетки, яркость и углы обзора. Например, код

```
library(scatterplot3d)
with(mtcars,
     scatterplot3d(wt, disp, mpg,
                   pch=16,
                   highlight.3d=TRUE,
                   type="h",
                   main="3D Scatter Plot with Vertical Lines"))
```

создаст трехмерную диаграмму рассеяния с градиентом яркости для увеличения иллюзии перспективы и вертикальными линиями, соединяющими точки с горизонтальной плоскостью (рис. 11.9).

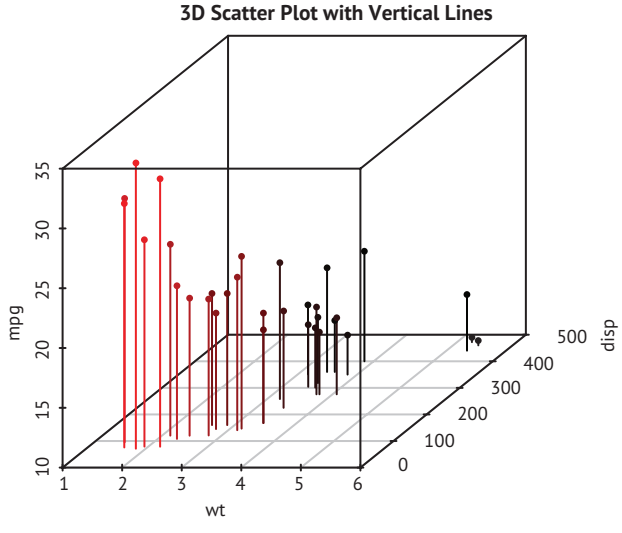


Рис. 11.9. Трехмерная диаграмма рассеяния с вертикальными линиями и градиентом яркости

В качестве последнего примера возьмем предыдущую диаграмму и добавим к ней регрессионную плоскость:

```
library(scatterplot3d)
s3d <-with(mtcars,
           scatterplot3d(wt, disp, mpg,
                         pch=16,
                         highlight.3d=TRUE,
                         type="h",
         main="3D Scatter Plot with Vertical Lines and Regression Plane"))
fit <- lm(mpg ~ wt+disp, data=mtcars)
s3d$plane3d(fit)
```

Полученная диаграмма показана на рис. 11.10.

Эта диаграмма позволяет визуализировать прогноз расхода топлива на основе веса автомобиля и объема двигателя, сделанный при помощи уравнения множественной регрессии. Плоскость определяет предсказанные значения, а точки представляют реальные наблюдения. Длина вертикальных отрезков между точками и плоскостью – это остатки. Регрессионные оценки значений, лежащих над плоскостью точек, занижены, а для лежащих под плоскостью точек – завышены. Множественная регрессия обсуждается в главе 8.

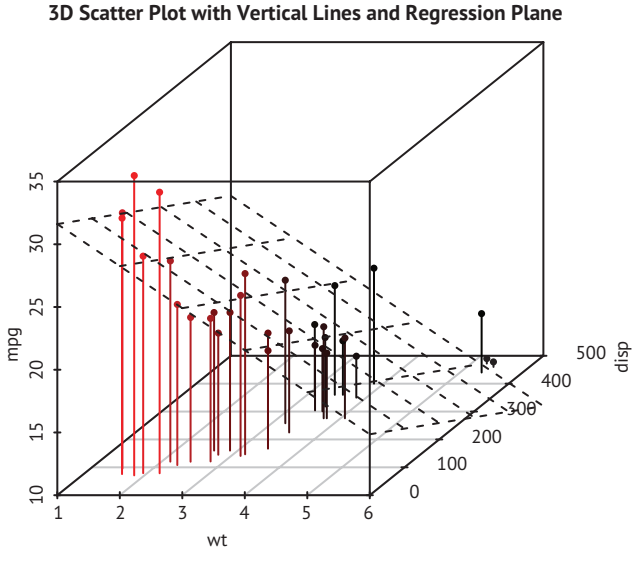


Рис. 11.10. Трехмерная диаграмма рассеяния с вертикальными линиями, градиентом яркости и наложенной регрессионной плоскостью

11.1.4. Вращение трехмерных диаграмм рассеяния

Трехмерные диаграммы рассеяния гораздо легче интерпретировать, если есть возможность взаимодействовать с ними. В R реализовано несколько способов вращения диаграмм, позволяющих рассмотреть точки с разных сторон.

Например, интерактивную трехмерную диаграмму рассеяния можно создать при помощи функции `plot3d()` из пакета `rgl`. Эта функция создает трехмерную диаграмму рассеяния, которую можно вращать при помощи мыши. Она имеет следующий синтаксис:

```
plot3d(x, y, z)
```

где *x*, *y* и *z* – это числовые векторы, определяющие координаты точек. Можно также добавить параметры `col` и `size`, чтобы задать цвет и размер точек. Продолжая наш пример, попробуйте выполнить следующий код:

```
library(rgl)
with(mtcars,
     plot3d(wt, disp, mpg, col="red", size=5))
```

У вас должна получиться диаграмма, изображенная на рис. 11.11. Используйте мышь, чтобы повернуть оси. Я надеюсь, вы согласитесь, что возможность поворачивать диаграмму рассеяния в трехмерном пространстве делает ее гораздо более доступной для понимания.

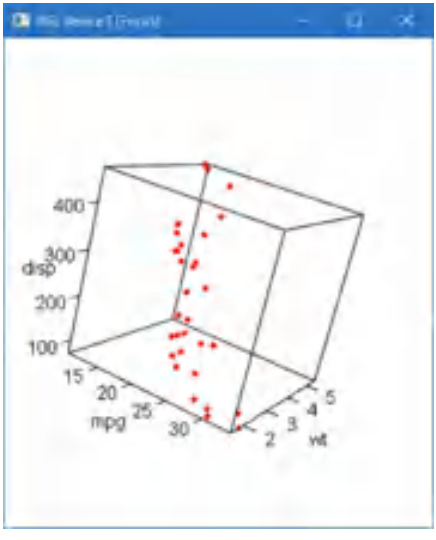


Рис. 11.11. Вращающаяся диаграмма рассеяния, полученная при помощи функции plot3d() из пакета rgl

Примерно тот же результат можно получить, используя функцию `scatter3d()` из пакета `car`:

```
library(car)
with(mtcars,
     scatter3d(wt, disp, mpg))
```

Результат показан на рис. 11.12.

При помощи функции `scatter3d()` можно изображать поверхности для разных типов регрессии, таких как линейная, квадратичная, сглаженная и аддитивная. По умолчанию изображается поверхность линейной регрессии. Существуют дополнительные возможности обозначения точек в интерактивном режиме. Дополнительную информацию ищите в `help(scatter3d)`.

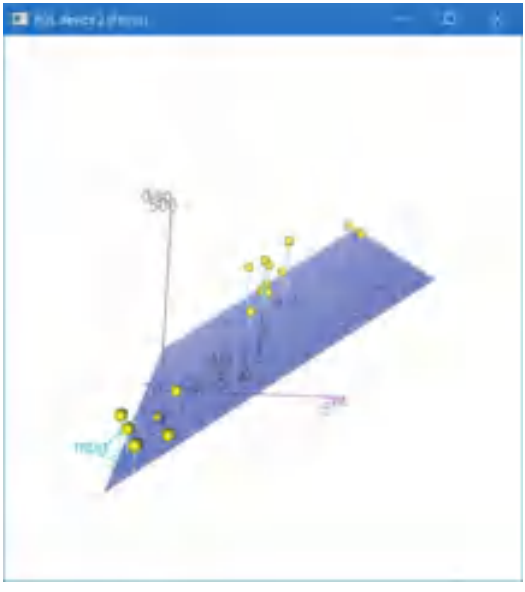


Рис. 11.12. Вращающаяся диаграмма рассеяния, полученная при помощи функции scatter3d() из пакета car

11.1.5. Пузырьковые диаграммы

В предыдущем разделе мы отображали взаимосвязи между тремя количественными переменными при помощи трехмерной диаграммы рассеяния. Другой способ – создать двумерную диаграмму рассеяния и использовать размер точек для представления значений третьей переменной. В результате получится так называемая *пузырьковая диаграмма* (bubble plot).

Вот простой пример создания пузырьковой диаграммы:

```
ggplot(mtcars,
   aes(x = wt, y = mpg, size = disp)) +
   geom_point() +
   labs(title="Bubble Plot with point size proportional to displacement",
        x="Weight of Car (lbs/1000)",
        y="Miles Per Gallon")
```

Эта диаграмма рассеяния (рис. 11.13) показывает взаимосвязь между весом автомобиля и экономичностью, а размер точек пропорционален рабочему объему двигателя.

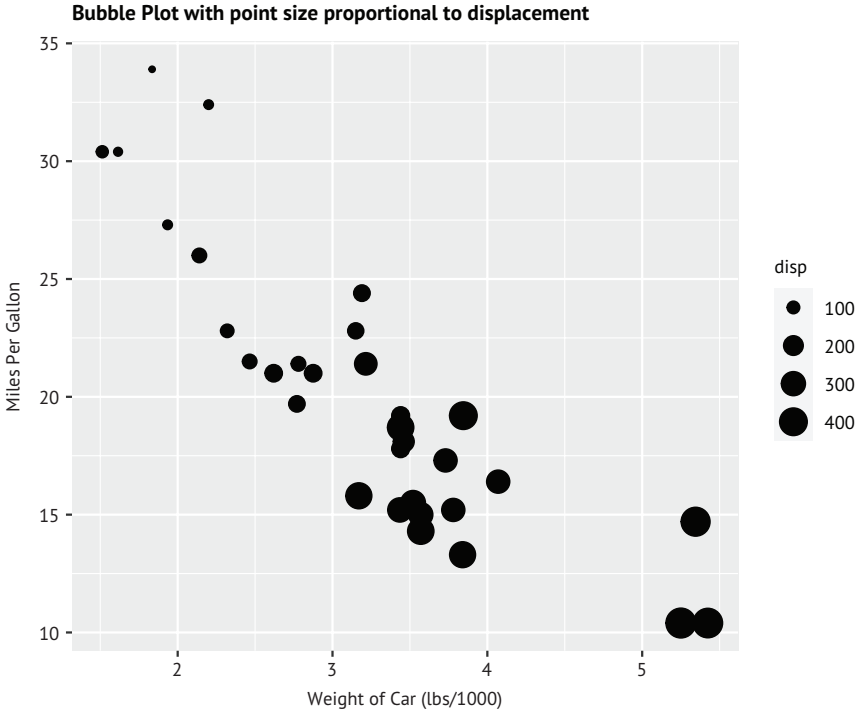


Рис. 11.13. Пузырьковая диаграмма, отражающая зависимость между весом автомобиля и расходом топлива (размер точек пропорционален объему двигателя)

Внешний вид по умолчанию можно улучшить, выбрав другую форму и цвет точек и добавив прозрачность, чтобы упростить распознавание перекрывающихся точек, а также увеличить диапазон размеров пузырьков. Наконец, можно использовать цвет, чтобы отобразить четвертую переменную – количество цилиндров. В листинге 11.4 показан код, создающий диаграмму, изображенную на рис. 11.14. Цвета трудно различить в черно-белой книге, но они хорошо различимы в цвете.

Листинг 11.4. Улучшенная пузырьковая диаграмма

```
ggplot(mtcars,
       aes(x = wt, y = mpg, size = disp, fill=factor(cyl))) +
  geom_point(alpha = .5,
             color = "black",
             shape = 21) +
  scale_size_continuous(range = c(1, 10)) +
  labs(title = "Auto mileage by weight and horsepower",
       subtitle = "Motor Trend US Magazine (1973-74 models)",
       x = "Weight (1000 lbs)",
       y = "Miles/(US) gallon",
```

```
      size = "Engine\ndisplacement",
      fill = "Cylinders") +
theme_minimal()
```

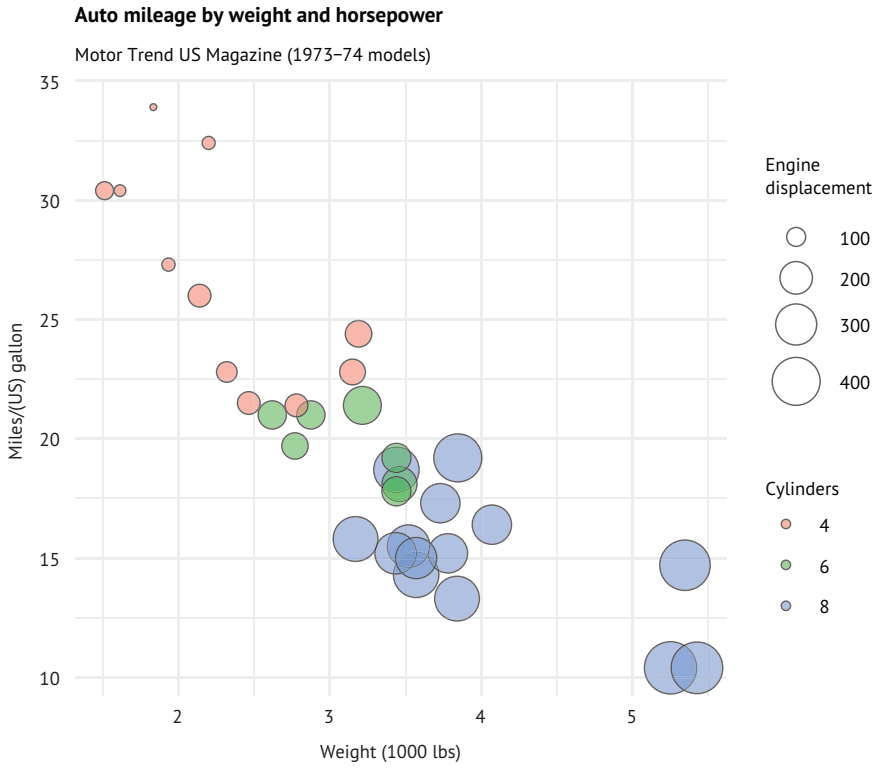


Рис. 11.14. Улучшенная пузырьковая диаграмма. Как правило, автомобили с большим весом оборудуются двигателями с большим числом цилиндров и имеют худшую экономичность

Статистики, использующие R, обычно избегают пузырьковых диаграмм по тем же причинам, по которым они не любят круговых диаграмм. Людям, как правило, труднее сравнивать объемы, чем расстояния. Однако пузырьковые диаграммы весьма популярны в деловом мире, поэтому я добавил их в книгу для полноты изложения.

Мне, конечно, пришлось долго рассказывать про диаграммы рассеяния. Это внимание к деталям частично обусловлено главенствующим положением, которое занимают диаграммы рассеяния в анализе данных. Несмотря на свою простоту, они помогают наглядно представить данные и связи, которые иначе могли остаться незамеченными.

11.2. Линейные графики

Если последовательно соединить точки диаграммы рассеяния, двигаясь слева направо, то получится линейный график. Набор данных `Orange`, входящий в состав дистрибутива R, содержит сведения о возрасте и обхвате пяти апельсиновых деревьев. Изучим рост первого дерева, данные о котором представлены на рис. 11.15. Слева показана диаграмма рассеяния, а справа – линейный график. Как видите, линейные графики особенно полезны, когда нужно проследить за изменениями значений анализируемой величины. Графики на рис. 11.15 созданы с помощью кода из листинга 11.5.

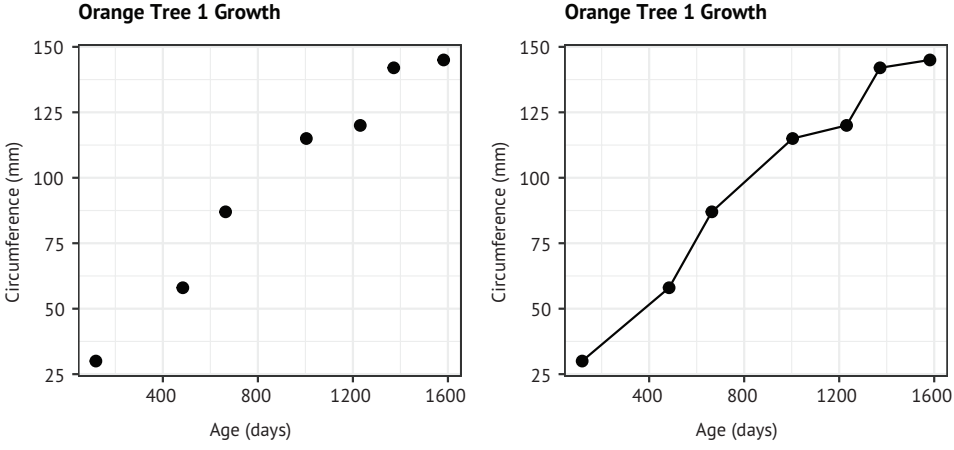


Рис. 11.15. Сравнение диаграммы рассеяния и линейного графика. Линейный график помогает увидеть, как протекал рост дерева

Листинг 11.5. Создание диаграммы рассеяния и линейного графика

```
library(ggplot2)
tree1 <- subset(Orange, Tree == 1)
ggplot(data=tree1,
       aes(x=age, y=circumference)) +
  geom_point(size=2) +
  labs(title="Orange Tree 1 Growth",
       x = "Age (days)",
       y = "Circumference (mm)") +
  theme_bw()

ggplot(data=tree1,
       aes(x=age, y=circumference)) +
  geom_point(size=2) +
  geom_line() +
  labs(title="Orange Tree 1 Growth",
       x = "Age (days)",
       y = "Circumference (mm)") +
  theme_bw()
```

Единственная разница между инструкциями, создающими графики, заключается в добавлении функции `geom_line()`. В табл. 11.1 перечислены общие параметры этой функции. Каждому можно присвоить значение или поставить ему в соответствие категориальную переменную.

Таблица 11.1. Параметры функции geom_line()

Параметр	Описание
size	Толщина линии
color	Цвет линии
linetype	Тип линии (например, dashed – пунктирная)

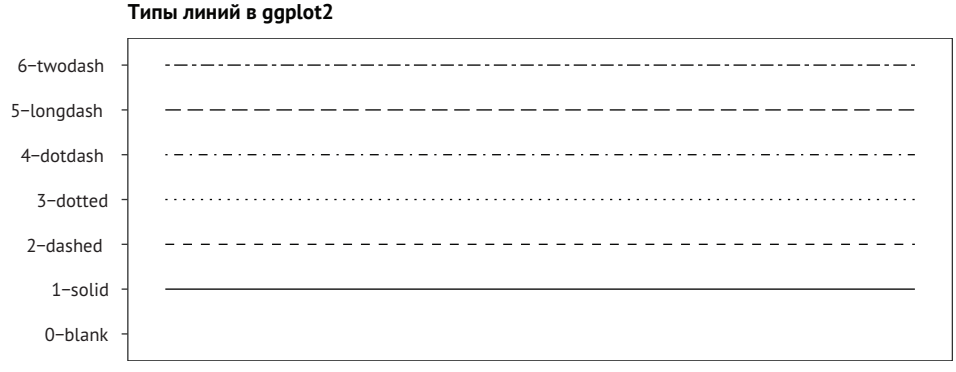


Рис. 11.16. Типы линий, которые можно передать функциям ggplot2. Тип линии можно задать именем или ее порядковым номером

В качестве примера создания более сложного линейного графика попробуем одновременно изобразить рост всех пяти апельсиновых деревьев с течением времени. Каждому дереву будет соответствовать линия своего типа и цвета. Код показан в листинге 11.6, а его результаты представлены на рис. 11.17.

Листинг 11.6. График, отображающий рост пяти апельсиновых деревьев с течением времени

```
library(ggplot2)
ggplot(data=Orange,
       aes(x=age, y=circumference, linetype=Tree, color=Tree)) +
  geom_point() +
  geom_line(size=1) +
  scale_color_brewer(palette="Set1") +
  labs(title="Orange Tree Growth",
       x = "Age (days)",
       y = "Circumference (mm)") +
  theme_bw()
```

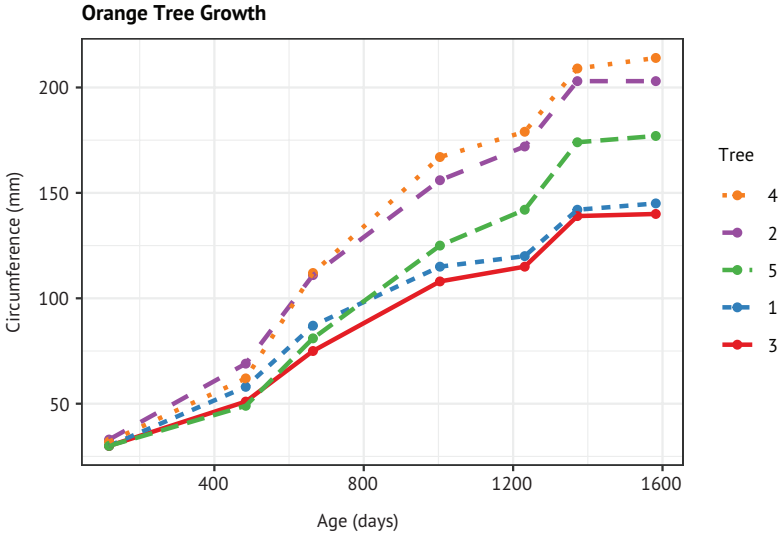


Рис. 11.17. График, отображающий рост пяти апельсиновых деревьев с течением времени

Настройка отображения номеров деревьев и соответствующих линий в листинге 11.6 выполняется с помощью функции `aes()`. Настройка палитры цветов выполняется с помощью функции `scale_color_brewer()`. Я не художник, и у меня нет развитого вкуса к цветам (у меня никак не получается подобрать хорошие цвета), поэтому я почти всегда полагаюсь на предопределенные цветовые палитры, например предоставляемые пакетом `RColorBrewer`. Подробнее о палитрах мы поговорим в главе 19 (продвинутые методы работы с диаграммами).

На рис. 11.17 видно, что дерево 4 и дерево 2 росли быстрее в течение периода, когда производились измерения, и дерево 4 обогнало дерево 2 примерно через 664 дня. По умолчанию в легенде диаграммы линии отображаются в порядке, обратном порядку их появления на диаграмме (сверху вниз в легенде и снизу вверх на графике). Чтобы порядок совпадал, нужно в листинг 11.6 добавить код:

```
+ guides(color = guide_legend(reverse = TRUE),
         linetype = guide_legend(reverse = TRUE))
```

В следующем разделе мы рассмотрим способы визуализации множества коэффициентов корреляции одновременно.

11.3. Кореллограммы

Матрицы корреляции – это один из основных элементов многомерной статистики. Какие переменные из рассматриваемых сильно коррелируют друг с другом, а какие – нет? Существуют ли клас-

теры переменных, которые связаны между собой определенным способом? С увеличением числа переменных ответить на такие вопросы становится все сложнее. *Кореллограммы* – это сравнительно недавно появившийся способ для визуализации корреляционных матриц.

Проще всего объяснить, что такое коррелограмма, – это изобразить ее. Рассмотрим корреляции между переменными в наборе данных `mtcars`. В нем есть 11 переменных, каждая из которых описывает некоторую характеристику 32 автомобилей. Корреляционную матрицу[1] можно вычислить так:

```
> round(cor(mtcars), 2)
       mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
mpg   1.00 -0.85 -0.85 -0.78  0.68 -0.87  0.42  0.66  0.60  0.48 -0.55
cyl  -0.85  1.00  0.90  0.83 -0.70  0.78 -0.59 -0.81 -0.52 -0.49  0.53
disp -0.85  0.90  1.00  0.79 -0.71  0.89 -0.43 -0.71 -0.59 -0.56  0.39
hp   -0.78  0.83  0.79  1.00 -0.45  0.66 -0.71 -0.72 -0.24 -0.13  0.75
drat  0.68 -0.70 -0.71 -0.45  1.00 -0.71  0.09  0.44  0.71  0.70 -0.09
wt   -0.87  0.78  0.89  0.66 -0.71  1.00 -0.17 -0.55 -0.69 -0.58  0.43
qsec  0.42 -0.59 -0.43 -0.71  0.09 -0.17  1.00  0.74 -0.23 -0.21 -0.66
vs    0.66 -0.81 -0.71 -0.72  0.44 -0.55  0.74  1.00  0.17  0.21 -0.57
am    0.60 -0.52 -0.59 -0.24  0.71 -0.69 -0.23  0.17  1.00  0.79  0.06
gear  0.48 -0.49 -0.56 -0.13  0.70 -0.58 -0.21  0.21  0.79  1.00  0.27
carb -0.55  0.53  0.39  0.75 -0.09  0.43 -0.66 -0.57  0.06  0.27  1.00
```

Какие пары переменных связаны теснее всего? Какие переменные относительно независимы? Есть ли тут какая-нибудь структура? Не так-то легко ответить на такие вопросы, просто глядя на эту корреляционную матрицу и не затратив много времени и усилий (и, возможно, не вооружившись набором цветных ручек для заметок).

Эту же самую корреляционную матрицу можно представить графически при помощи функции `corrgram()` из пакета `corrgram` (рис. 11.18). Вот необходимый код:

```
library(corrgram)
corrgram(mtcars, order=TRUE, lower.panel=panel.shade,
         upper.panel=panel.pie, text.panel=panel.txt,
         main="Corrgram of mtcars intercorrelations")
```

[1] Конечно, вычислять коэффициент корреляции Пирсона (а именно их, как вы помните, вычисляет по умолчанию функция cor) для дихотомических переменных, таких как am, некорректно. – *Прим. перев.*

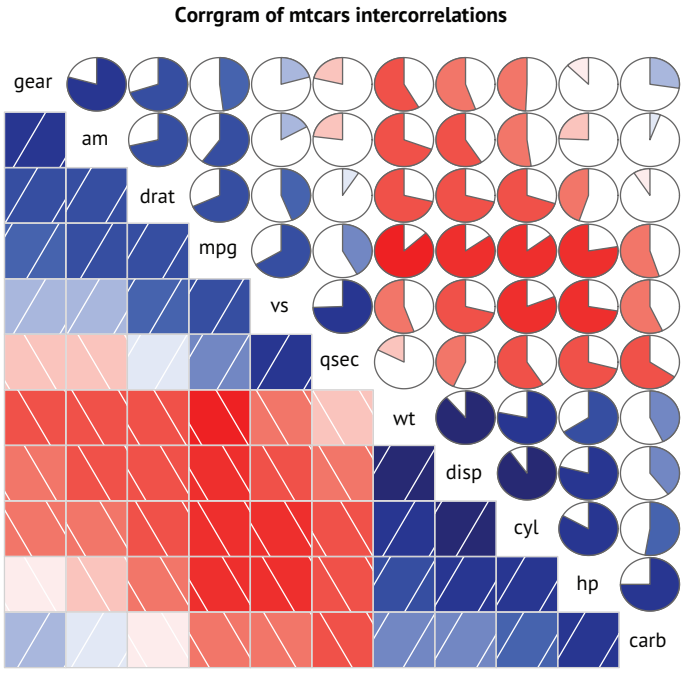


Рис. 11.18. Кореллограмма, отражающая взаимосвязи (корреляции) между переменными из таблицы данных mtcars. Порядок строк и столбцов изменен, согласно результатам анализа методом главных компонент

Начнем интерпретацию этой диаграммы с нижнего треугольника ячеек (ячеек, находящихся ниже главной диагонали). По умолчанию синий цвет и штриховка слева направо и снизу вверх соответствуют положительной корреляции между двумя переменными, на пересечении которых находится данная ячейка. Напротив, красный цвет и штриховка сверху вниз и слева направо соответствуют отрицательной корреляции. Чем насыщеннее цвет, тем сильнее корреляция. Слабые, близкие к нулю корреляции изображаются «выцветшими» ячейками. На представленной диаграмме порядок строк и столбцов автоматически изменен по результатам анализа методом главных компонент (о котором мы поговорим в главе 14), чтобы переменные со сходной корреляционной структурой формировали кластеры.

Можно видеть, что переменные `gear`, `am`, `drat` и `mpg` положительно коррелируют друг с другом. Также заметно, что переменные `wt`, `disp`, `cyl`, `hp` и `carb` тоже положительно коррелируют между собой. Однако переменные из первой группы отрицательно коррелируют с переменными второй группы. Очевидно и то, что корреляция между переменными `carb` и `am` слабая; это же справедливо для таких пар переменных, как `vs` и `gear`, `vs` и `am`, `drat` и `qsec`.

В верхнем треугольнике та же информация представлена в виде круговых диаграмм. Здесь цвета имеют такое же значение, а сила корреляции выражена в размере закрашенного сегмента круговой диаграммы. Сегменты, соответствующие положительным корреляциям, начинаются от положения «12 часов» и заполняют круг по часовой стрелке. Сегменты, соответствующие отрицательным корреляциям, заполняют круг против часовой стрелки.

Функция `corrgram()` имеет следующий синтаксис:

```
corrgram(x, order=, panel=, text.panel=, diag.panel=),
```

где *x* – таблица данных с одним наблюдением на строку. Если `order=TRUE`, то порядок переменных изменяется согласно результатам анализа корреляционной матрицы методом главных компонент. Такая сортировка переменных помогает обнаружить закономерности в корреляционной структуре.

Параметр `panel` определяет вид диаграммы (кроме главной диагонали – ее свойства задаются отдельно). Вместо него можно использовать параметры `lower.panel` и `upper.panel`, чтобы отдельно определять вид нижней и верхней (по отношению к главной диагонали) половин диаграммы. Параметры `text.panel` и `diag.panel` относятся к главной диагонали. Допустимые значения параметра `panel` перечислены в табл. 11.2.

Таблица 11.2. Значения параметра panel функции corrgram()

Положение	Значение параметра	Описание
Не на главной диагонали	`panel.pie`	Закрашенный сегмент круговой диаграммы соответствует силе корреляции
	`panel.shade`	Интенсивность цвета соответствует силе корреляции
	`panel.ellipse`	Изображаются доверительный эллипс и сглаживающая линия
	`panel.pts`	Изображается диаграмма рассеяния
	`panel.conf`	Выводит корреляции и их доверительные интервалы
	`panel.cor`	Выводит корреляции без доверительных интервалов
Главная диагональ	`panel.minmax`	Приводятся минимальное и максимальное значения переменной
	`panel.txt`	Отображается название переменной
	`panel.density`	Выводит диаграмму функции плотности и имя переменной

Рассмотрим второй пример:

```
library(corrgram)
```

```
corrgram(mtcars, order=TRUE, lower.panel=panel.ellipse,
         upper.panel=panel.pts, text.panel=panel.txt,
         diag.panel=panel.minmax,
         main="Corrgram of mtcars data using scatter plots
               and ellipses")
```

Этот код создаст диаграмму, представленную на рис. 11.19. Здесь в нижнем треугольнике дополнительно выводятся доверительные эллипсы и сглаживающие линии, а в верхнем – диаграммы рассеяния.

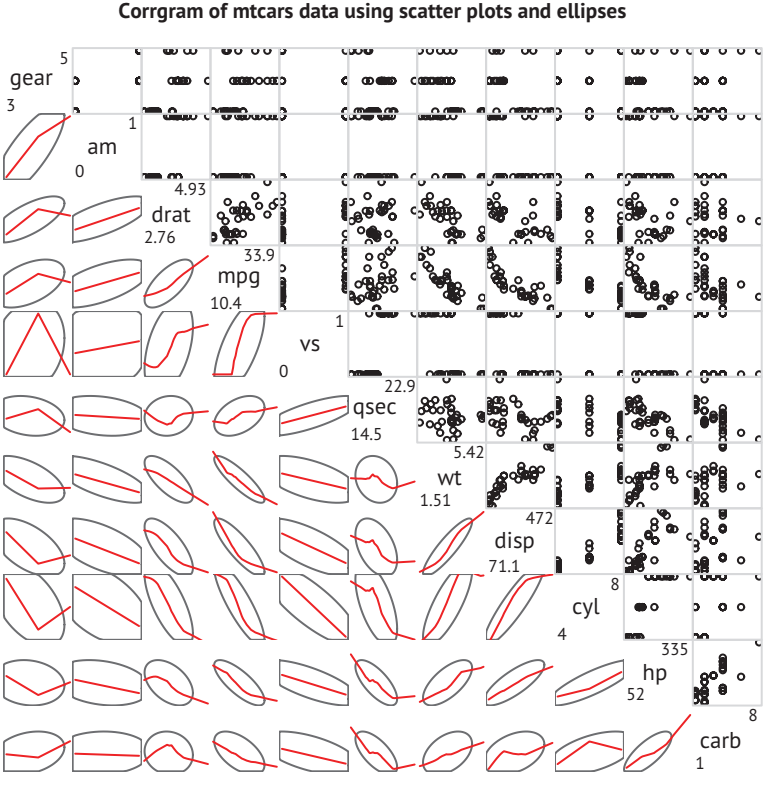


Рис. 11.19. Коррелограмма для корреляционной матрицы между переменными из таблицы данных mtcars. В нижнем треугольнике показаны сглаживающие линии вместе с доверительными эллипсами, а в верхнем – диаграммы рассеяния. На главной диагонали показаны минимальные и максимальные значения. Строки и столбцы упорядочены согласно результатам анализа методом главных компонент

Почему диаграммы рассеяния выглядят такими странными?

Некоторые из переменных, изображенных на рис. 11.19, имеют ограниченные допустимые значения. Например, количество передач равно 3, 4 или 5. Количество цилиндров равно 4, 6 или 8. А переменные `am` (тип трансмиссии) и `vs` (расположение цилиндров – рядное или

V-образное) вообще имеют только два значения. Это объясняет странный вид диаграмм рассеяния над главной диагональю.

Всегда следите за тем, чтобы выбранные вами статистические методы соответствовали форме данных. Может пригодиться преобразование таких переменных в упорядоченные или неупорядоченные факторы. Если переменная является категориальной или порядковой, то R пытается применить соответствующие статистические методы.

В заключение рассмотрим еще один пример:

```
corrgram(mtcars, order=TRUE, lower.panel=panel.shade,
         upper.panel=panel.cor,
         main="Corrgram of mtcars data using shading and coefficients")
```

Этот код создаст диаграмму, изображенную на рис. 11.20. В данном случае используются оттенки цвета в нижнем треугольнике, порядок переменных отражает закономерности в корреляциях, а в верхнем треугольнике выводятся значения коэффициентов корреляции.

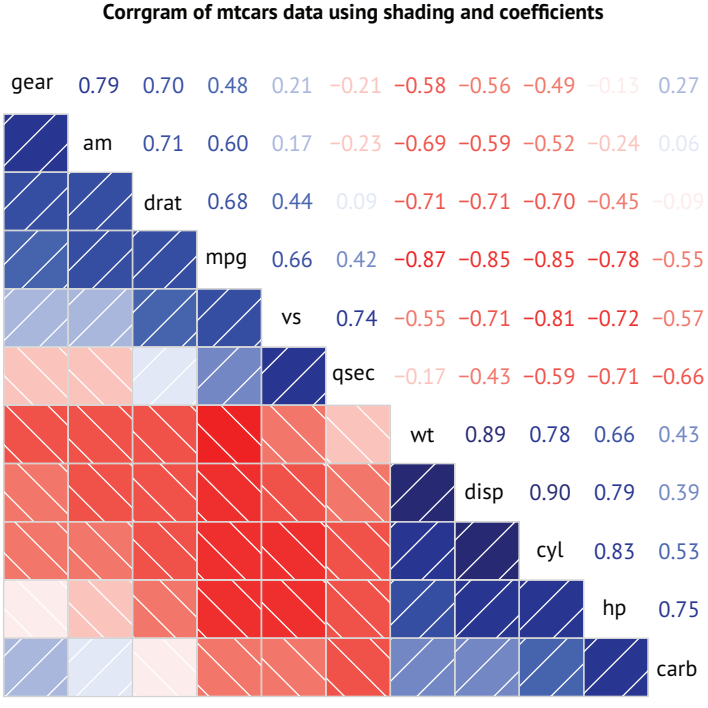


Рис. 11.20. Коррелограмма для корреляционной матрицы между переменными из таблицы данных mtcars. Оттенки цвета в нижнем треугольнике отражают силу и направление корреляций. Строки и столбцы переупорядочены в соответствии с результатами анализа методом главных компонент. В верхнем треугольнике выводятся коэффициенты корреляции

Прежде чем двинуться дальше, хочу отметить, что функция `corrgram()` позволяет управлять цветами. Для этого нужно вызвать функцию `colorRampPalette()` и передать ее результат в параметре `col.regions`. Например:

```
library(corrgram)
cols <- colorRampPalette(c("darkgoldenrod4", "burlywood1",
                           "darkkhaki", "darkgreen"))
corrgram(mtcars, order=TRUE, col.regions=cols,
         lower.panel=panel.shade,
         upper.panel=panel.conf, text.panel=panel.txt,
         main="A Corrgram (or Horse) of a Different Color")
```

Попробуйте выполнить этот код и посмотрите, что получится.

Коррелограммы могут быть хорошим способом изучения большого числа парных связей между количественными переменными. Поскольку это относительно новый метод, самое сложное – объяснить людям, как эти диаграммы нужно интерпретировать. Дополнительную информацию вы найдете в статье Майкла Фрэндли (Michael Friendly) «Corrgrams: Exploratory Displays for Correlation Matrices», доступной по адресу https://www.datavis.ca/papers/corrgram.pdf.

11.4. Мозаичные диаграммы

До сих пор мы исследовали методы визуализации взаимоотношений между количественными или непрерывными переменными. А что, если переменные – категориальные? Исследуя одну категориальную переменную, можно использовать столбиковую или круговую диаграмму. Для двух категориальных переменных можно построить составную столбиковую диаграмму (раздел 6.1.2). Но что делать, если категориальных переменных больше?

Одно из возможных решений – построить *мозаичную диаграмму*. В этих диаграммах частоты из многомерной таблицы сопряженности представлены в виде вложенных прямоугольников с размерами, пропорциональными частотам. Для отображения отклонений от подобранной модели можно использовать цвета и/или штриховку. Подробности ищите в публикации Meyer, Zeileis и Hornick (2006) или в превосходном руководстве Майкла Фрэндли (https://cran.r-project.org/web/packages/vcdExtra/vignettes/vcd-tutorial.pdf).

Мозаичные диаграммы можно создавать при помощи функции `mosaic()` из пакета `vcd` (в базовой версии R есть функция `mosaicplot()`, но я рекомендую использовать пакет `vcd`, так как он предлагает больше возможностей). В качестве примера рассмотрим набор данных `Titanic`, входящий в состав дистрибутива R. Он содержит сведения о пассажирах, выживших и погибших, а также о классе, которым они путешествовали (первый, второй, третий, экипаж

[Crew]), поле (мужской – Male, женский – Female) и возрасте (ребенок – Child, взрослый – Adult). Это хорошо изученный набор данных. Число сочетаний разных признаков можно подсчитать так:

```
> ftable(Titanic)
                  Survived  No Yes
Class Sex    Age
1st   Male   Child            0   5
             Adult          118  57
      Female Child            0   1
             Adult            4 140
2nd   Male   Child            0  11
             Adult          154  14
      Female Child            0  13
             Adult           13  80
3rd   Male   Child           35  13
             Adult          387  75
      Female Child           17  14
             Adult           89  76
Crew  Male   Child            0   0
             Adult          670 192
      Female Child            0   0
             Adult            3  20
```

Функция mosaic() вызывается так:

```
mosaic(table)
```

где *table* – это таблица сопряженности в виде массива, или так:

```
mosaic(formula, data=),
```

где *formula* – это формула в формате R, а параметр *data* задает или набор данных, или таблицу. Дополнительный параметр shade=TRUE позволит раскрасить диаграмму в соответствии со значениями пирсоновских остатков от подобранной модели (по умолчанию подразумевается независимость переменных), параметр legend=TRUE создает легенду, расшифровывающую значения использованных цветов.

К примеру, оба следующих примера:

```
library(vcd)
mosaic(Titanic, shade=TRUE, legend=TRUE)
```

и:

```
library(vcd)
mosaic(~Class+Sex+Age+Survived, data=Titanic, shade=TRUE, legend=TRUE)
```

– создадут диаграмму, изображенную на рис. 11.21. Версия с формулой предоставляет больше возможностей по размещению переменных на диаграмме.

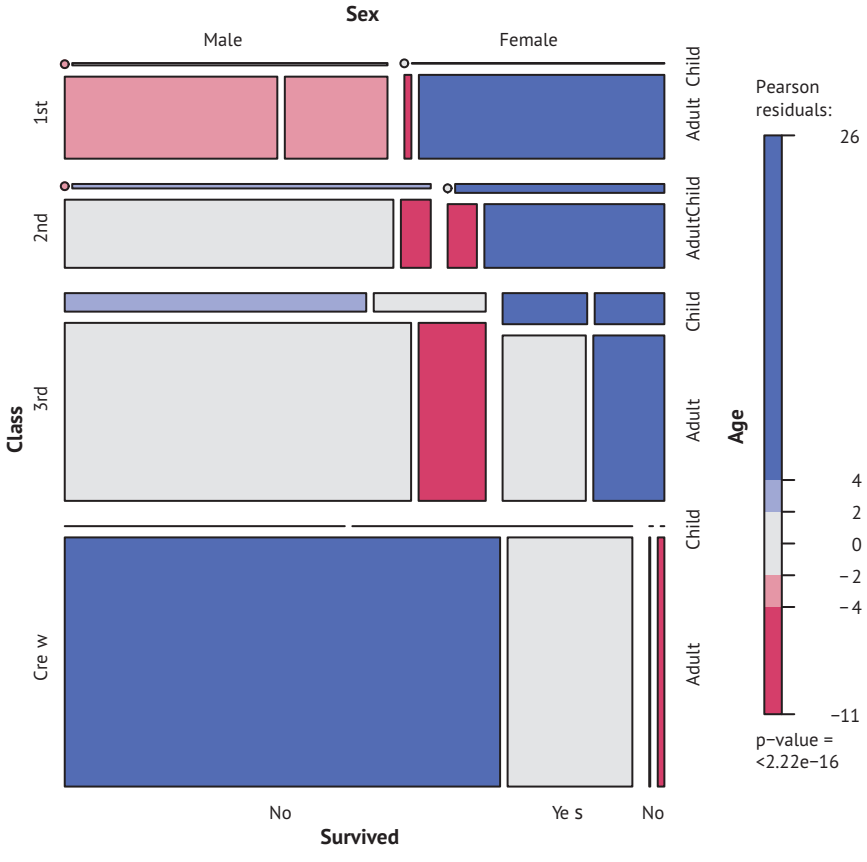


Рис. 11.21. Мозаичная диаграмма, характеризующая пол и возраст людей, выживших при крушении Титаника, а также класс, в котором они путешествовали

В этой диаграмме содержится уйма информации. Например, вероятность выжить у пассажиров, путешествовавших первым классом, намного выше, чем у экипажа. Большинство детей путешествовали вторым и третьим классами. Большинство женщин, путешествовавших первым классом, выжили, а из женщин, путешествовавших третьим классом, выжило только около половины. В составе экипажа было очень мало женщин, поэтому на диаграмме индикаторы выживания (нет – `No` и да – `Yes` внизу диаграммы) наложились друг на друга. Продолжайте рассматривать диаграмму, и вы обнаружите много интересных фактов. Не забывайте сопоставлять относительную ширину и высоту прямоугольников. Что еще вы смогли узнать о той ночи?

На усовершенствованных модификациях мозаичных диаграмм цвета и оттенки используются для отображения остатков от подобранной модели. На этой диаграмме оттенки синего соответствуют

сочетаниям значений, которые имели место чаще, чем ожидалось, в том случае, когда вероятность выжить не зависела от класса, возраста и пола. Оттенки красного соответствуют сочетаниям значений, которые наблюдались реже, чем ожидалось, исходя из предположения о независимости переменных. Обязательно опробуйте этот пример на своем компьютере, чтобы увидеть диаграмму в цвете. На ней видно, что выжило больше женщин из первого класса и погибло больше мужчин из экипажа, чем ожидалось, исходя из предположения о независимости вероятности выживания от класса, возраста и пола. Также выжило меньше мужчин из третьего класса, по сравнению с ожидаемыми значениями. За дополнительной информацией о мозаичных диаграммах обращайтесь к справке `example(mosaic)`.

В этой главе мы рассмотрели разнообразные способы графического отображения взаимосвязей между двумя и более переменными, включая двух- и трехмерные диаграммы рассеяния, матрицы диаграмм рассеяния, пузырьковые диаграммы, линейные графики, коррелограммы и мозаичные диаграммы. Некоторые из этих методов стандартные, а некоторые – сравнительно новые.

Вместе с методами, которые позволяют изображать распределение значений одномерных переменных (глава 6), исследовать регрессионные модели (глава 8) и визуализировать межгрупповые различия (глава 9), рассмотренные в этой главе подходы дают нам необходимый инструментарий для визуализации ваших данных и извлечения информации из них.

В последующих главах вы расширите свои умения за счет дополнительных методов, включая диаграммы для моделей со скрытыми (латентными) переменными (глава 14), временных рядов (глава 15), результатов кластеризации (глава 16) и пропущенных значений (глава 18).

Итоги

- Диаграммы рассеяния и матрицы диаграмм рассеяния позволяют визуализировать отношения между парами количественных переменных. Диаграммы могут быть дополнены сглаживающими и аппроксимирующими линиями, показывающими тренды.
- При создании диаграммы рассеяния на основе большого набора данных могут пригодиться методы визуализации плотности распределения значений вместо самих точек.
- Отношения между тремя количественными переменными можно исследовать с помощью трехмерных диаграмм рассеяния или двумерных пузырьковых диаграмм.

- Изменения во времени можно эффективно описать с помощью линейных диаграмм.
- Большие корреляционные матрицы трудно осмыслить, если они представлены в табличной форме, но легко поддаются интерпретации с помощью коррелограмм – наглядных диаграмм визуального представления матриц корреляции.
- Взаимосвязи между двумя или более категориальными переменными можно визуализировать с помощью мозаичных диаграмм.

12 Статистика повторных выборок и бутстреп-анализ

В этой главе:

- понимание логики критериев перестановок;
- применение критериев перестановок к линейным моделям;
- использование бутстреп-анализа для вычисления доверительных интервалов.

В главах 7, 8 и 9 мы рассматривали статистические методы проверки гипотез и вычисления доверительных интервалов для параметров генеральной совокупности, подразумевая, что полученные данные подчиняются нормальному распределению или распределению другого типа с хорошо известными свойствами. Однако иногда это предположение не гарантировано. В случаях, когда данные имеют неизвестное или смешанное распределение, когда размеры выборок малы, когда в данных присутствует много выбросов или когда разработка подходящего критерия, основанного на теоретическом распределении, слишком сложна и затруд-

нительна с математической точки зрения, можно использовать статистические подходы, основанные на рандомизации и повторных выборках.

В этой главе мы рассмотрим два основных статистических подхода с использованием рандомизации: критерий перестановки и бутстреп-анализ. Раньше эти методы были доступны только опытным программистам и экспертам в области статистики. Однако теперь, с появлением специализированных пакетов для R, эти методы стали доступными для всех, кто занимается анализом данных.

Мы также вернемся к задачам, которые раньше анализировали при помощи традиционных методов (например, критериев Стьюдента и хи-квадрат, дисперсионного и регрессионного анализов), и посмотрим, как можно решить эти задачи, используя устойчивые к ошибкам методы, требующие большого объема вычислений. Для лучшего понимания раздела 12.2 необходимо сначала прочитать главу 7. Главы 8 и 9 содержат информацию, нужную для понимания раздела 12.3. Остальные разделы можно читать без предварительной подготовки.

12.1. Критерии перестановок

Критерии перестановок (permutation tests), также известные как *критерии с рандомизацией*, были известны в течение десятилетий, но стали доступными лишь с появлением высокопроизводительных компьютеров. Чтобы понять логику критериев перестановок, рассмотрим следующую воображаемую задачу. Десять объектов случайно подвергались одному из двух воздействий (`A` или `B`), а результаты воздействий регистрировались в результирующей переменной `score`. Результаты эксперимента представлены в табл. 12.1.

Таблица 12.1. Воображаемая задача с двумя группами

Воздействие A	Воздействие B
40	57
57	64
45	55
55	62
58	65

Результаты изображены на рис. 12.1. Достаточно ли этого, чтобы утверждать, что разные воздействия приводят к разным результатам?

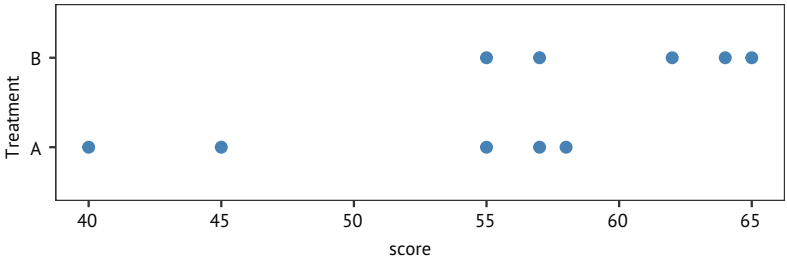


Рис. 12.1. Одномерная диаграмма рассеяния для воображаемых данных, представленных в табл. 12.1

При параметрическом подходе можно предположить, что данные получены из совокупностей, имеющих нормальное распределение с одинаковой дисперсией, и применить критерий Стьюдента для двух независимых групп. Нулевая гипотеза заключается в том, что среднее значение для выборки A равно среднему для выборки B. Для этого можно было бы вычислить критерий Стьюдента для данных и сравнить его значение с теоретическим распределением. Если бы вычисленное значение критерия достаточно сильно отличалось от теоретического, скажем находилось бы вне 95 % значений теоретического распределения, то можно было бы отвергнуть нулевую гипотезу и заявить, что выборочные средние для этих двух групп неодинаковы с вероятностью 95 %.

Критерий перестановок применяет другой подход. Если два воздействия действительно равнозначны, то названия («воздействие A» или «воздействие B») присвоены им случайным образом. Мы можем проверить достоверность различий между воздействиями при помощи следующего алгоритма.

1. Вычислить выборочную *t*-статистику, как при параметрическом подходе, назовем ее `t0`.
2. Поместить все 10 значений в одну группу.
3. Случайно поместить пять значений в группу A и пять – в группу B.
4. Снова вычислить *t*-статистику.
5. Повторить шаги 3–4 для всех возможных способов перестановки значений по пять в группе A и пять в группе B. Всего таких способов 252.
6. Расположить 252 значения *t*-статистики в порядке возрастания. Это эмпирическое распределение, основанное на выборочных данных.
7. Если `t0` не входит в центральные 95 % значений эмпирического распределения, то нулевую гипотезу о равенстве средних значений в двух группах можно отвергнуть с вероятностью 95 %.

Обратите внимание, что в обоих подходах, параметрическом и с перестановками, вычисляется одна и та же *t*-статистика. Однако вместо сравнения статистики с теоретическим распределением для выяснения значимости отличий от ожидаемых значений, чтобы можно было отклонить нулевую гипотезу, *t*-статистика сравнивается с эмпирическим распределением, полученным при перестановках данных. Эту логику можно распространить на большинство классических статистических критериев и линейных моделей.

В только что приведенном примере эмпирическое распределение было основано на всех возможных перестановках данных. В таких случаях критерий перестановок называется *точным* критерием. С увеличением размеров выборок необходимое для проведения всех возможных перестановок время может стать непомерно большим. В таких случаях можно использовать моделирование Монте-Карло, создавая выборки из всех возможных перестановок. При таком образе действий критерий получается приближенным.

Если вы не уверены в нормальном распределении данных, задумываетесь о влиянии выбросов или чувствуете, что набор данных может быть недостаточно большим для стандартных параметрических подходов, критерий перестановок послужит отличной альтернативой параметрическим методам. В R реализованы самые сложные и широко применимые методы перестановок из имеющихся в настоящее время. В этом разделе мы рассмотрим два пакета: `coin` и `lmPerm`. Пакет `coin` содержит обширный набор критериев перестановок для проверки данных на независимость, а в пакете `lmPerm` реализованы критерии перестановок для дисперсионного и регрессионного анализов. Мы рассмотрим их по очереди. Не забудьте установить эти пакеты перед использованием (`install.packages(c("coin", "lmPerm"))`).

Настройка начального значения для генератора псевдослучайных чисел

Прежде чем двигаться дальше, нужно вспомнить, что критерии перестановок используют псевдослучайные числа для создания выборки из всех возможных перестановок (для приближенных критериев). По этой причине результаты каждого отдельного прогона могут немного отличаться. Для получения одинаковых результатов нужно задать начальное число для генератора псевдослучайных чисел. Это особенно полезно, если вы предполагаете поделиться своими результатами с другими. Установка начального числа, равного 1234 (т. е. `set.seed(1234)`), позволит воспроизвести результаты, представленные в этой главе.

12.2. Критерии перестановок в пакете coin

В пакете `coin` реализованы разнообразные критерии перестановок для проверки данных на независимость. При помощи этого пакета можно ответить на такие вопросы, как:

- Зависят ли результаты от принадлежности к группе?
- Независимы ли две числовые переменные?
- Независимы ли две категориальные переменные?

Используя функции из этого пакета (табл. 12.2), можно вычислять критерии перестановок, аналогичные большинству традиционных статистических критериев, описанных в главе 7.

Таблица 12.2. Функции в пакете coin для вычисления критериев перестановок, эквивалентных традиционным статистическим критериям

Критерий	Функция в пакете `coin`*
Критерий перестановок для двух и *k* выборок	`oneway_test(y ~ A)`
Критерий перестановок для двух и *k* выборок с группирующим фактором	`oneway_test(y ~ A | C)`
Критерий ранговых сумм Вилкоксона–Манна–Уитни	`wilcox_test(y ~ A)`
Тест Краскела–Уоллиса	`kruskal_test(y ~ A)`
Критерий Пирсона хи-квадрат	`chisq_test(A ~ B)`
Критерий Кохрана–Мантеля–Гензеля	`cmh_test(A ~ B | C)`
Критерий линейной зависимости	`lbl_test(D ~ E)`
Критерий Спирмена	`spearman_test(y ~ x)`
Тест Фридмана	`friedman_test(y ~ A | C)`
Ранговый критерий Вилкоксона	`wilcoxsign_test(y1 ~ y2)`

* `y` и `x` – числовые переменные, `A` и `B` – категориальные факторы, `C` – категориальная группирующая переменная, `D` и `E` – упорядоченные факторы, `y1` и `y2` – парные числовые переменные.

Все функции, перечисленные в табл. 12.2, имеют следующий синтаксис:

```
имя_функции(formula, data, distribution= )
```

где

- *formula* описывает взаимосвязь между проверяемыми переменными. Примеры приведены в таблице;
- *data* – таблица данных;

- `distribution` определяет, как должно создаваться эмпирическое распределение для нулевой гипотезы. Возможные значения: `exact`, `asymptotic` и `approximate`.

С параметром `distribution="exact"` распределение для нулевой гипотезы рассчитывается точно (т. е. на основании всех возможных перестановок). Распределение может также вычисляться приближенно по асимптотическому распределению (`distribution="asymptotic"`) или методом перестановок Монте-Карло (`distribution="approximate(nresample=n)"`), где `n` означает число повторений для приближенного вычисления точного распределения. В настоящее время параметр `distribution="exact"` доступен только при работе с двумя выборками.

> **ПРИМЕЧАНИЕ.** Для анализа категориальные и порядковые переменные должны быть представлены факторами и упорядоченными факторами соответственно. Кроме того, данные должны передаваться в формате таблицы данных.

В оставшейся части этого раздела мы применим несколько критериев перестановок из числа перечисленных в табл. 12.2 для решения задач из предыдущих глав. Это позволит вам сравнить результаты параметрических и непараметрических критериев. А в завершение обсуждения пакета `coin` мы рассмотрим более сложные приемы анализа.

12.2.1. Проверка независимости двух и k выборок

Для начала сравним критерий Стьюдента для независимых выборок с односторонним точным критерием на примере воображаемых данных из табл. 12.2. Результаты представлены в листинге 12.1.

Листинг 12.1. Сравнение критерия Стьюдента с односторонним критерием перестановок для воображаемых данных

```
> library(coin)
> score <- c(40, 57, 45, 55, 58, 57, 64, 55, 62, 65)
> treatment <- factor(c(rep("A",5), rep("B",5)))
> mydata <- data.frame(treatment, score)
> t.test(score~treatment, data=mydata, var.equal=TRUE)

        Two Sample t-test

data:  score by treatment
t = -2.345, df = 8, p-value = 0.04705
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -19.0405455  -0.1594545
sample estimates:
mean in group A mean in group B
           51.0            60.6
```

```
> oneway_test(score~treatment, data=mydata, distribution="exact")

        Exact Two-Sample Fisher-Pitman Permutation Test

data:  score by treatment (A, B)
Z = -1.9147, p-value = 0.07143
alternative hypothesis: true mu is not equal to 0
```

Традиционный критерий Стьюдента свидетельствует о статистической значимости различий между группами (`p < 0.05`), тогда как точный непараметрический критерий – нет (`p > 0.072`). Имея только 10 наблюдений, я больше склонен верить результатам критерия перестановок и попытаться собрать больше данных, прежде чем прийти к окончательному заключению.

Далее рассмотрим критерий Вилкоксона–Манна–Уитни. В главе 7 мы исследовали различия вероятности попасть в тюрьму в южных и неюжных штатах США, используя функцию `wilcox.test()`. Применив точный критерий ранговых сумм Вилкоксона, получаем следующий результат:

```
> library(MASS)
> UScrime$So <- factor(UScrime$So)
> wilcox_test(Prob ~ So, data=UScrime, distribution="exact")

        Exact Wilcoxon Mann-Whitney Rank Sum Test

data:  Prob by So (0, 1)
Z = -3.7, p-value = 8.488e-05
alternative hypothesis: true mu is not equal to 0
```

который свидетельствует о том, что вероятность угодить за решетку выше в южных штатах. Обратите внимание, что в этом программном коде числовая переменная `So` была преобразована в фактор. Это необходимо потому, что, как уже упоминалось, для анализа с помощью функций из пакета `coin` все категориальные переменные должны быть преобразованы в факторы. Кроме того, внимательный читатель мог заметить, что эти результаты идеально согласуются с результатами функции `wilcox.test()`, приведенными в главе 7. Это объясняется тем, что по умолчанию `wilcox.test()` тоже рассчитывает точное распределение.

Наконец, рассмотрим критерий для k выборок. В главе 9 мы использовали однофакторный дисперсионный анализ и оценивали влияние пяти разных режимов приема лекарств на уровень холестерина по выборке из 50 пациентов. Теперь вычислим приближенный критерий перестановок для k выборок:

```
> library(multcomp)
> set.seed(1234)
> oneway_test(response~trt, data=cholesterol,
  distribution=approximate(nresample=9999))
```

```
        Approximative K-Sample Fisher-Pitman Permutation Test

data:  response by trt (1time, 2times, 4times, drugD, drugE)
chi-squared = 36.381, p-value < 1e-04
```

В данном случае опорное распределение рассчитано на основе 9999 перестановок данных. Мы задали начальное число для генератора псевдослучайных чисел, поэтому ваши результаты совпадут с моими. Как видите, результаты для пациентов из разных групп заметно различаются.

12.2.2. Независимость в таблицах сопряженности

С помощью критерия перестановок, реализованных в виде функций `chisq_test()` или `cmh_test()`, можно оценить независимость двух категориальных переменных. Вторая функция применяется, когда данные разделены на группы согласно значениям третьей категориальной переменной. Если обе переменные порядковые, то с помощью функции `cmh_test()` можно проверить наличие линейного тренда.

В главе 7 мы использовали критерий хи-квадрат для проверки связи между способом лечения артрита и улучшением состояния больных. Переменная `Treatment` (лечение) имела два уровня (`Placebo` и `Treated`), а переменная `Improved` (улучшение) – три уровня (`None`, `Some` и `Marked`) – она была представлена как упорядоченный фактор.

Вычислить критерий перестановок, аналогичный критерию хи-квадрат, можно следующим образом:

```
> library(coin)
> library(vcd)
> Arthritis <- transform(Arthritis,
  Improved=as.factor(as.numeric(Improved)))
> set.seed(1234)
> chisq_test(Treatment~Improved, data=Arthritis,
             distribution=approximate(nresample=9999))

                Approximative Pearson Chi-Squared Test

data:  Treatment by Improved (1, 2, 3)
chi-squared = 13.055, p-value = 0.0018
```

Этот код вычислит приближенный критерий хи-квадрат, основанный на 9999 перестановках. Возможно, у кого-то из вас возник вопрос: зачем переменная `Improved` была преобразована из упорядоченного фактора в категориальную переменную? Хороший вопрос! Если оставить ее в виде упорядоченного фактора, то пакет `coin` вычислит критерий для проверки линейного тренда вместо хи-квадрат. Проверка критерия наличия линейного тренда могла бы быть хорошим решением в данной ситуации, но нам нужен именно критерий хи-квадрат, чтобы сравнить результаты с приведенными в главе 7.

12.2.3. Независимость между числовыми переменными

Функция `spearman_test()` позволяет вычислить критерий перестановок для оценки независимости двух числовых переменных. В главе 7 мы исследовали корреляцию между уровнем неграмотности и уровнем преступности в разных штатах США. Эту связь можно проверить при помощи критерия перестановок, как показано ниже:

```
> states <- as.data.frame(state.x77)
> set.seed(1234)
> spearman_test(Illiteracy~Murder, data=states,
                distribution=approximate(B=9999))

            Approximative Spearman Correlation Test

data:  Illiteracy by Murder
Z = 4.7065, p-value < 1e-04
alternative hypothesis: true rho is not equal to 0
```

На основании приближенного критерия с 9999 перестановками нулевая гипотеза может быть отвергнута. Обратите внимание, что набор данных `state.x77` – это матрица. Перед использованием пакета `coin` ее нужно преобразовать в таблицу данных.

12.2.4. Критерии перестановок для двух и k зависимых выборок

Критерии перестановок для зависимых выборок используются, когда пары наблюдений в разных группах соответствуют друг другу или в случаях повторных измерений. Вычислить критерий перестановок для двух зависимых выборок можно с помощью функции `wilcoxsign_test()`, а для большего числа групп – с помощью функции `friedman_test()`.

В главе 7 мы сравнивали уровень безработицы для горожан мужского пола в возрасте 14–24 лет (переменная `U1`) и в возрасте 35–39 лет (`U2`). Поскольку значения обеих переменных известны для 50 штатов Америки, мы имеем две зависимые группы (`state` (штат) – переменная, задающая пары значений). Для сравнения уровня безработицы в этих двух возрастных группах можно использовать ранговый тест Вилкоксона:

```
> library(coin)
> library(MASS)
> wilcoxsign_test(U1~U2, data=UScrime, distribution="exact")

        Exact Wilcoxon-Signed-Rank Test

data:  y by x (neg, pos)
         stratified by block  Z = 5.9691, p-value = 1.421e-14
alternative hypothesis: true mu is not equal to 0
```

На основании полученных результатов можно заключить, что уровни безработицы различаются.

12.2.5. Дополнительная информация

В пакете coin реализован основной набор методов проверки независимости одной группы переменных от другой (с возможным разделением на группы согласно значениям еще одной сторонней переменной) при помощи критериев перестановок. В частности, функция independence_test() позволяет пользователю вычислить большинство обычных критериев перестановок, а также создать новые и нестандартные статистические критерии для ситуаций, в которых традиционные подходы не работают. Такая гибкость имеет свою цену: для правильного использования этой функции требуется отличное значение статистики. За дальнейшими деталями обращайтесь к сопроводительной документации с описанием пакета (vignette("coin")).

В следующем разделе вы познакомитесь с пакетом lmPerm. В нем реализованы критерии перестановок для линейных моделей, в том числе регрессионного и дисперсионного анализов.

12.3. Критерии перестановок в пакете lmPerm

В пакете lmPerm реализован подход на основе перестановок к анализу линейных моделей. В частности, функции lmp() и aovp() – это аналоги функций lm() и aov(), модифицированные для вычисления критериев перестановок.

Функции lmp() и aovp() принимают те же параметры, что и функции lm() и aov(), плюс один дополнительный параметр perm=, в котором могут передаваться значения "Exact" (точный критерий, основанный на всех возможных перестановках), "Prob" и "SPR". Значение "Prob" позволяет вычислить критерий по выборке из возможных перестановок. Создание выборки прекращается, когда оценка стандартного отклонения значения p становится меньше 0,1. Условие остановки определяется дополнительным параметром Ca. Наконец, параметр SPR позволяет использовать критерий отношений последовательных вероятностей в качестве условия остановки. Имейте в виду, что если число больше 10, то параметр perm="Exact" автоматически заменяется параметром perm="Prob"; точные критерии доступны только для малых выборок.

Чтобы посмотреть, как это работает, применим подход на основе перестановок к простой, полиномиальной и множественной регрессиям, одно- и двухфакторному дисперсионному анализу и однофакторному ковариационному анализу.

12.3.1. Простая и полиномиальная регрессия

В главе 8 мы использовали линейную регрессию для изучения связи между весом и ростом в группе из 15 женщин. Применив функцию lmp() вместо lm(), можно получить результат критерия перестановок, как показано в листинге 12.2.

Листинг 12.2. Критерий перестановок для простой линейной регрессии

```
> library(lmPerm)
> set.seed(1234)
> fit <- lmp(weight~height, data=women, perm="Prob")
[1] "Settings:  unique SS : numeric variables centered"
> summary(fit)

Call:
lmp(formula = weight ~ height, data = women, perm = "Prob")

Residuals:
   Min     1Q Median     3Q    Max
-1.733 -1.133 -0.383  0.742  3.117

Coefficients:
       Estimate Iter Pr(Prob)
height     3.45 5000   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.5 on 13 degrees of freedom
Multiple R-Squared: 0.991,      Adjusted R-squared: 0.99
F-statistic: 1.43e+03 on 1 and 13 DF,  p-value: 1.09e-14
```

В листинге 12.3 показано, как реализовать подгонку квадратичной модели.

Листинг 12.3. Критерий перестановок для полиномиальной регрессии

```
> library(lmPerm)
> set.seed(1234)
> fit <- lmp(weight~height + I(height^2), data=women, perm="Prob")
[1] "Settings:  unique SS : numeric variables centered"
> summary(fit)

Call:
lmp(formula = weight ~ height + I(height^2), data = women, perm = "Prob")

Residuals:
   Min     1Q  Median     3Q    Max  -0.5094 -0.2961 -0.0094  0.2862  0.5971

Coefficients:
            Estimate Iter Pr(Prob)
height       -7.3483 5000   <2e-16 ***
I(height^2)   0.0831 5000   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.38 on 12 degrees of freedom
Multiple R-Squared: 0.999,      Adjusted R-squared: 0.999
F-statistic: 1.14e+04 on 2 and 12 DF,  p-value: <2e-16
```

Как видите, вычислить критерий перестановок просто – нужно лишь немного изменить программный код. Кроме того, `lmp()` выводит результаты в том же формате, что и `lm()`. Обратите внимание на дополнительный столбец `Iter`, в котором указано число итераций, потребовавшихся для достижения условия остановки.

12.3.2. Множественная регрессия

В главе 8 мы использовали множественную регрессию для предсказания уровня преступности по значениям численности населения, уровня неграмотности, дохода и морозности для 50 штатов США. В листинге 12.4 показано, какие результаты дает применение функции `lmp()` для решения этой задачи.

Листинг 12.4. Критерий перестановок для множественной регрессии

```
> library(lmPerm)
> set.seed(1234)
> states <- as.data.frame(state.x77)
> fit <- lmp(Murder~Population + Illiteracy+Income+Frost,
             data=states, perm="Prob")
[1] "Settings:  unique SS : numeric variables centered"
> summary(fit)

Call:
lmp(formula = Murder ~ Population + Illiteracy + Income + Frost,
    data = states, perm = "Prob")

Residuals:
     Min       1Q   Median       3Q      Max
-4.79597 -1.64946 -0.08112  1.48150  7.62104

Coefficients:
            Estimate Iter Pr(Prob)
Population 2.237e-04   51   1.0000
Illiteracy 4.143e+00 5000   0.0004 ***
Income     6.442e-05   51   1.0000
Frost      5.813e-04   51   0.8627
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '. ' 0.1 ' ' 1

Residual standard error: 2.535 on 45 degrees of freedom
Multiple R-Squared: 0.567,      Adjusted R-squared: 0.5285
F-statistic: 14.73 on 4 and 45 DF,  p-value: 9.133e-08
```

Как мы выяснили в главе 8, согласно классическим критериям, численность населения (`Population`) и уровень неграмотности (`Illiteracy`) статистически значимо (`p < 0.05`) влияют на уровень преступности. Но критерий перестановок сообщает, что численность населения не оказывает существенного эффекта на уровень преступности.

Когда результаты двух подходов не совпадают, нужно более внимательно посмотреть на данные. Возможно, не соблюдается предположение о нормальности распределения или присутствуют выбросы.

12.3.3. Однофакторные дисперсионный и ковариационный анализы

Все виды дисперсионного анализа, описанные в главе 9, можно провести при помощи критериев перестановок. Для начала рассмотрим задачу с однофакторным дисперсионным анализом, описанную в разделе 9.1, о влиянии разных способов лечения на снижение уровня холестерина. Решение и результаты представлены в листинге 12.5.

Листинг 12.5. Критерий перестановок для однофакторного дисперсионного анализа

```
> library(lmPerm)
> library(multcomp)
> set.seed(1234)
> fit <- aovp(response~trt, data=cholesterol, perm="Prob")
[1] "Settings:  unique SS "
> anova(fit)
Component 1 :
           Df R Sum Sq R Mean Sq Iter  Pr(Prob)
trt         4  1351.37    337.84 5000 < 2.2e-16 ***
Residuals  45   468.75     10.42
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '. ' 0.1 ' ' 1
```

Результаты свидетельствуют о том, что разные способы лечения имеют неодинаковый эффект.

Второй пример в этом разделе: применение критерия перестановок для проведения однофакторного ковариационного анализа. Задача, описанная в главе 9, заключается в исследовании влияния четырех доз препаратов на вес мышат при разной продолжительности беременности. Решение и результаты представлены в листинге 12.6.

Листинг 12.6. Критерий перестановок для однофакторного ковариационного анализа

```
> library(lmPerm)
> set.seed(1234)
> fit <- aovp(weight ~ gesttime + dose, data=litter, perm="Prob")
[1] "Settings:  unique SS : numeric variables centered"
> anova(fit)
Component 1 :
           Df R Sum Sq R Mean Sq Iter Pr(Prob)
gesttime    1   161.49   161.493 5000   0.0006 ***
dose        3   137.12    45.708 5000   0.0392 *
Residuals  69  1151.27    16.685
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Из полученных значений статистической ошибки первого рода следует, что разные дозы препарата неодинаково влияют на вес мышат при разных сроках беременности.

12.3.4. Двухфакторный дисперсионный анализ

Мы закончим этот раздел примером применения критерия перестановок для двухфакторного дисперсионного анализа. В главе 9 мы исследовали влияние витамина C на рост зубов у морских свинок. Двумя изменяемыми факторами были доза (три уровня) и способ получения витамина (два уровня). Каждой комбинации воздействий подвергалось по десять свинок, в результате чего был получен сбалансированный 3×2 факторный план эксперимента. Порядок вычисления критерия перестановок представлен в листинге 12.7.

Листинг 12.7. Критерий перестановок для двухфакторного дисперсионного анализа

```
> library(lmPerm)
> set.seed(1234)
> fit <- aovp(len~supp*dose, data=ToothGrowth, perm="Prob")
[1] "Settings:  unique SS : numeric variables centered"
> anova(fit)
Component 1 :
           Df R Sum Sq R Mean Sq Iter Pr(Prob)
supp        1   205.35    205.35 5000  < 2e-16 ***
dose        1  2224.30   2224.30 5000  < 2e-16 ***
supp:dose   1    88.92     88.92 2032  0.04724 *
Residuals  56   933.63     16.67
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

На уровне значимости 0,05 все три эффекта отличаются от нуля. На уровне значимости 0,01 от нуля отличается только главный эффект.

Важно отметить, что когда функция `aovp()` применяется к разным типам дисперсионного анализа, по умолчанию каждый эффект корректируется по всем остальным эффектам (используется так называемая *уникальная сумма квадратов*, или *сумма квадратов SAS III типа*). По умолчанию для параметрического дисперсионного анализа в R применяется последовательная сумма квадратов (*SAS I типа*). Каждый эффект корректируется по эффектам, которые появляются в модели *раньше*. Для сбалансированного плана эксперимента оба подхода дадут одинаковый результат, а для несбалансированного, с разным числом наблюдений для разных комбинаций значений факторов, разные методы дадут разные результаты. Чем больше несбалансированность, тем больше будут различаться результаты. При желании можно вычислить последовательную сумму квадратов, добавив в вызов функции `aovp()` параметр `seqs=TRUE`. За более подробной информацией о суммах квадратов I и III типов обращайтесь к разделу 9.2.

12.4. Дополнительные замечания о критериях перестановок

Критерии перестановок – достойная альтернатива критериям, основанным на предположениях о распределении наблюдений в выборках. При помощи всех описанных выше критериев перестановок можно проверять статистические гипотезы, не опираясь на предположение, что данные в выборке подчиняются нормальному распределению или распределениям Стьюдента, Фишера либо хи-квадрат.

Вы могли заметить, насколько хорошо результаты вычисления классических критериев соответствуют результатам критериев перестановок. Данные в этих задачах были «правильными», и степень согласия между методами ясно показывает, насколько хорошо в таких ситуациях работают классические методы.

Критерии перестановок блистательно показывают себя в случаях, когда распределение данных значительно отличается от нормального (например, сильно асимметричное), когда присутствуют выбросы, когда размеры выборок малы или когда не существует параметрических критериев нужного типа. Однако если имеющаяся выборка нерепрезентативна, то никакой критерий, включая критерий перестановок, не сможет сделать выводы более корректными.

Критерии перестановок в основном используются для вычисления статистической ошибки первого рода (p-value). Они могут помочь ответить на вопрос «Существует ли закономерность?». Сложнее использовать такие критерии для вычисления доверительных интервалов и оценки точности измерения. К счастью, это область, в которой превосходно работает бутстреп-анализ.

12.5. Бутстреп-анализ

Бутстреп-анализ (bootstrapping) создает эмпирическое распределение статистики или набора статистик путем создания многих случайных выборок с возвратом из исходной выборки. Это позволяет вычислять доверительные интервалы и проверять статистические гипотезы, не выдвигая предположений о теоретическом распределении данных.

Логику бутстреп-анализа проще объяснить на примере. Допустим, вы хотите рассчитать 95%-ный доверительный интервал для выборочного среднего. У вас есть 10 наблюдений, по которым вы определили выборочное среднее, равное 40, и выборочное стандартное отклонение, равное 5. Если вы готовы допустить нормальное распределение выборочного среднего, то $(1 - \alpha/2)$%-ный доверительный интервал можно вычислить так:

$$\overline{X} - t\frac{s}{\sqrt{n}} < \mu < \overline{X} + t\frac{s}{\sqrt{n}},$$

где t – это верхнее $(1 - \alpha/2)$ критическое значение для распределения t с $n - 1$ степенями свободы. В случае 95%-ного доверительного интервала получаем $40 - 2{,}262(5/3{,}163) < \mu < 40 + 2{,}262(5/3{,}162)$, или $36{,}424 < \mu < 43{,}577$. То есть с уверенностью 95 % можно ожидать, что вычисленный интервал будет включать истинное среднее значение генеральной совокупности.

Но как быть, если нет оснований принять допущение о нормальном распределении выборочных средних? Тогда вместо приведенных выше вычислений можно использовать бутстреп-анализ:

1. Случайно выбрать 10 наблюдений из выборки с возвратом значений после каждого выбора. Некоторые наблюдения могут быть выбраны больше одного раза, а некоторые – могут остаться вовсе не выбранными.
2. Вычислить среднее для полученной выборки из 10 значений.
3. Повторить шаги 1 и 2 тысячу раз.
4. Отсортировать тысячу выборочных средних по возрастанию.
5. Найти выборочные средние, которые представляют собой 2,5 и 97,5 процентили. В данном случае 25-е число с начала и с конца. Это и будут границы 95%-ного доверительного интервала.

Похоже, что в данном случае выборочные средние распределены нормально, поэтому применение бутстреп-анализа не даст особого выигрыша. Однако есть много ситуаций, когда такой подход дает преимущество. Например, может понадобиться вычислить доверительные интервалы для выборочной медианы или разницы между двумя выборочными медианами. Для данного случая нет простой «классической» формулы, и бутстреп-анализ может оказаться как нельзя кстати. Если распределение данных неизвестно, если выборка содержит слишком много выбросов, если размеры выборок малы или если не существует параметрического метода, то бустреп-анализ часто оказывается полезным способом вычисления доверительных интервалов и проверки гипотез.

12.6. Проведение бутстреп-анализа при помощи пакета boot

Пакет `boot` предоставляет обширные возможности для бутстреп-анализа и связанных с ним методов создания выборок с возвратом. Бутстреп-анализ можно применить к одной статистике (например, к медиане) или к вектору статистик (например, к набору коэффициентов регрессии). Не забудьте скачать и установить пакет `boot` перед его первым использованием (`install.packages("boot")`).

Бутстреп-анализ может показаться сложным, но после изучения нескольких примеров все должно стать понятным.

В общем случае бутстреп-анализ состоит из трех этапов:

1. создание функции, вычисляющей нужную статистику или статистики. Если имеется одна статистика (например, медиана), то функция должна возвращать число. Если есть набор статистик (например, набор коэффициентов регрессии), то функция должна возвращать вектор;
2. применение функции `boot()` к вашей функции, чтобы создать бутстреп-реплики вашей статистики или статистик;
3. применение функции `boot.ci()` для вычисления доверительных интервалов.

Теперь перейдем к частностям. Основная функция, выполняющая бустреп-анализ, – это `boot()`. Она имеет следующий синтаксис:

```
bootobject <- boot(data=, statistic=, R=, ...)
```

Все параметры функции описаны в табл. 12.3.

Таблица 12.3. Параметры функции boot()

Параметр	Описание
`data`	Вектор, матрица или таблица данных
`statistic`	Функция, вычисляющая *k* статистик, которые будут подвергнуты бутстреп-анализу (*k* = 1, если есть только одна статистика)
`R`	Число бутстреп-выборок
`...`	Дополнительные параметры функции, которая создает нужную статистику или нужные статистики

Функция `boot()` вызывает функцию `statistic()` *R* раз. Каждый раз она генерирует набор целых чисел, выбранных случайно с возвратом из диапазона `1:nrow(data)`. Эти числа используются для создания новой выборки. Статистики вычисляются для новой выборки, а результаты записываются в *`bootobject`*. Структура этого объекта описана в табл. 12.4.

Таблица 12.4. Элементы объекта, создаваемого функцией boot()

Элемент	Описание
`t0`	Наблюдаемые значения *k* статистик для исходных данных
`t`	Матрица *R k*, где каждая строка – это одна из бутстреп-реализаций *k* статистик

Эти элементы доступны как `bootobject$t0` и `bootobject$t`.

После создания новых выборок результаты бутстреп-анализа можно получить при помощи функций `print()` и `plot()`. Если результаты выглядят разумно, можно вычислить доверительные ин-

тервалы для статистик(и) при помощи функции boot.ci(), которая имеет следующий синтаксис:

```
boot.ci(bootobject, conf=, type= )
```

Параметры описаны в табл. 12.5.

Таблица 12.5. Параметры функции boot.ci()

Параметр	Описание
bootobject	Объект, создаваемый функцией boot()
conf	Требуемый доверительный интервал (по умолчанию conf=0.95)
type	Тип вычисляемого доверительного интервала. Возможны следующие значения: "norm", "basic", "stud", "perc", "bca" и "all" (по умолчанию type="all")

Параметр type определяет способ вычисления границ доверительного интервала. Метод процентилей (perc) был разобран в примере с выборочным средним. Метод bca вычисляет интервал, который делает простые поправки на смещение. Я считаю, что bca – самый удачный метод в большинстве ситуаций. Обсуждение этих методов можно найти в публикации Mooney & Duval (1993).

В оставшихся разделах мы сначала рассмотрим бутстреп-анализ для одной статистики, а затем для вектора статистик.

12.6.1. Бутстреп-анализ для одной статистики

Набор данных mtcars содержит информацию о 32 автомобилях, опубликованную в журнале *Motor Trend* за 1974 год. Предположим, что мы использовали множественную регрессию для предсказания пробега машины на одном галлоне топлива по ее весу и рабочему объему цилиндров двигателя. В дополнение к стандартным регрессионным статистикам мы хотели бы получить 95%-ный доверительный интервал для коэффициента детерминации (доли дисперсии зависимой переменной, которая объясняется независимыми переменными). Доверительный интервал можно получить при помощи непараметрического бутстреп-анализа.

Первая задача – написать функцию для вычисления коэффициента детерминации:

```
rsq <- function(formula, data, indices) {
         d <- data[indices,]
         fit <- lm(formula, data=d)
         return(summary(fit)$r.square)
}
```

Выражение d <- data[indices,] нужно, чтобы функция boot() могла создавать выборки.

Затем можно получить большое число повторных бутстреп-выборок (скажем, 1000):

```
library(boot)
set.seed(1234)
results <- boot(data=mtcars, statistic=rsq,
                R=1000, formula=mpg~wt+disp)
```

и вывести результаты на экран:

```
> print(results)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = mtcars, statistic = rsq, R = 1000, formula = mpg ~
    wt + disp)


Bootstrap Statistics :
     original      bias     std. error
t1* 0.7809306  0.01333670   0.05068926
```

или построить диаграмму вызовом функции `plot(results)`. Полученная диаграмма представлена на рис. 12.2.

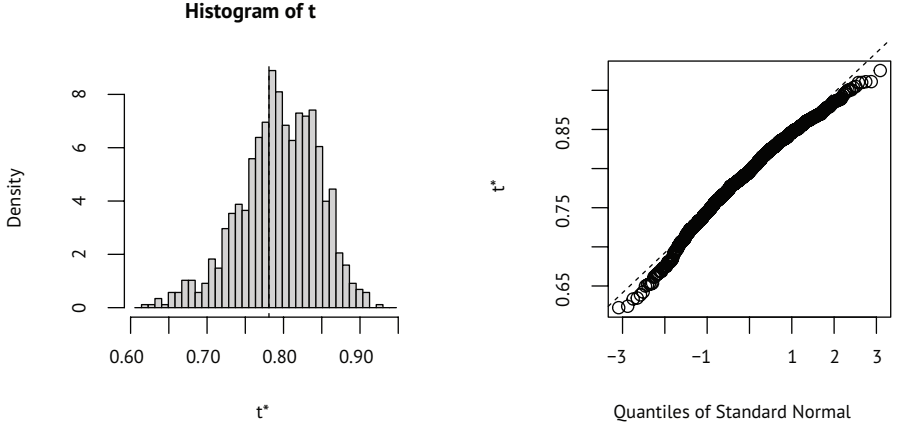


Рис. 12.2. Распределение значений коэффициентов детерминации, полученных при бутстреп-анализе

На рис. 12.2 можно видеть, что распределение значений коэффициентов детерминации, полученных при бутстреп-анализе, отличается от нормального. Теперь вычислим 95%-ный доверительный интервал для коэффициента детерминации:

```
> boot.ci(results, type=c("perc", "bca"))
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = c("perc", "bca"))
```

```
Intervals :
Level     Percentile            BCa
95%   ( 0.6753,  0.8835 )   ( 0.6344,  0.8561 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
```

Из этого примера видно, что разные способы вычисления доверительного интервала могут давать разные результаты. В этом случае интервал с поправкой на систематическую ошибку (`bca`) несколько отличается от интервала, вычисленного методом процентилей (`proc`). В любом случае нулевая гипотеза H_0 о равенстве коэффициента детерминации нулю будет отвергнута, потому что ноль находится за пределами доверительного интервала.

В этом разделе мы вычислили границы доверительного интервала для одной статистики. В следующем разделе мы оценим доверительные интервалы сразу для нескольких статистик.

12.6.2. Бутстреп-анализ для нескольких статистик

В предыдущем примере бутстреп-анализ использовался для оценки доверительного интервала для одной статистики (коэффициента детерминации). Продолжая этот пример, найдем 95%-ные доверительные интервалы для вектора статистик, а именно рассчитаем доверительные интервалы для трех коэффициентов регрессионной модели (свободный член, вес автомобиля и объем цилиндров).

Сначала напишем функцию, возвращающую вектор коэффициентов регрессии:

```
bs <- function(formula, data, indices) {
        d <- data[indices,]
        fit <- lm(formula, data=d)
        return(coef(fit))
}
```

Затем применим эту функцию для получения 1000 бутстреп-выборок:

```
library(boot)
set.seed(1234)
results <- boot(data=mtcars, statistic=bs,
                R=1000, formula=mpg~wt+disp)
> print(results)
ORDINARY NONPARAMETRIC BOOTSTRAP
Call:
boot(data = mtcars, statistic = bs, R = 1000, formula = mpg ~
    wt + disp)

Bootstrap Statistics :
    original   bias    std. error
t1*  34.9606  0.137873     2.48576
t2*  -3.3508 -0.053904     1.17043
t3*  -0.0177 -0.000121     0.00879
```

При бутстреп-анализе нескольких статистик в вызов функций `plot()` и `boot.ci()` нужно добавлять параметр индекса, чтобы указать, какую колонку `bootobject$t` требуется проанализировать. В этом примере `1` соответствует свободному члену, `2` – весу машины, а `3` – объему цилиндров. Чтобы построить диаграмму с результатами для веса машины, вызовем функцию

```
plot(results, index=2)
```

Полученная диаграмма показана на рис. 12.3.

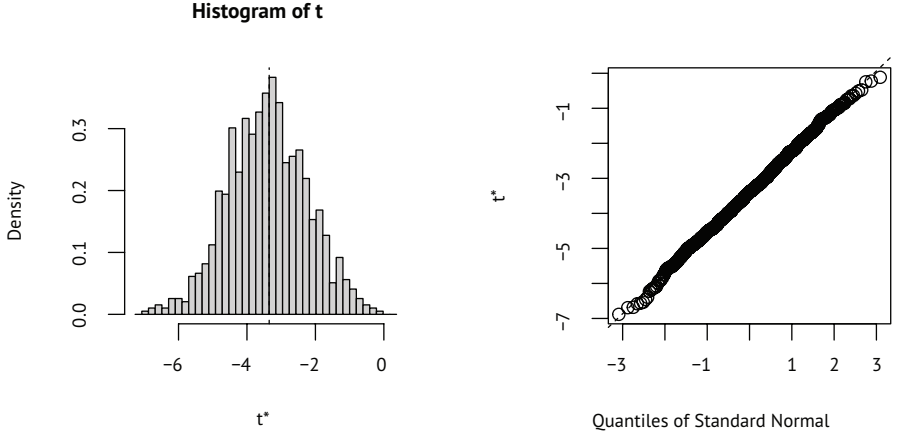


Рис. 12.3. Распределение коэффициентов регрессии для веса автомобиля, полученных при бутстреп-анализе

Получить 95%-ные доверительные интервалы коэффициентов регрессии для веса машины и объема двигателя можно так:

```
> boot.ci(results, type="bca", index=2)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 2)

Intervals :
Level       Bca
95%   (-5.477, -0.937 )
Calculations and Intervals on Original Scale

> boot.ci(results, type="bca", index=3)

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = results, type = "bca", index = 3)
```

```
Intervals :
Level       BCa
95%   (-0.0334, -0.0011 )
Calculations and Intervals on Original Scale
```

ПРИМЕЧАНИЕ. В предыдущем примере мы извлекали повторные выборки из всей исходной выборки. Если допустить, что независимые переменные имеют фиксированные значения (обычное дело для спланированных экспериментов), то повторные выборки лучше получать только для остатков. См. публикацию Mooney & Duval (1993, стр. 16–17), где приводятся простое объяснение и алгоритм.

Прежде чем закончить обсуждение бутстреп-анализа, стоит ответить на два часто возникающих вопроса:

- Насколько велика должна быть исходная выборка?
- Сколько выборок-реплик следует получить?

На эти вопросы не существует простого ответа. Некоторые говорят, что размер исходной выборки в 20–30 наблюдениях достаточен для получения хороших результатов, если выборка репрезентативна. Случайный выбор объектов из генеральной совокупности – наиболее надежный способ получения репрезентативной выборки. Что касается второго вопроса, то, по моему мнению, 1000 реплик более чем достаточно в большинстве случаев. Компьютерное время дешево, и при необходимости всегда можно увеличить число реплик.

Существует много полезных источников информации по критериям перестановок и бутстреп-анализу. Прекрасный выбор для начала – статья Yu (2003), имеющаяся в открытом доступе в интернете. В публикации Good (2006) представлен обширный общий обзор методов повторных выборок, включая примеры программного кода на R. Хорошее и доступное введение в бутстреп-анализ можно найти в публикации Mooney & Duval (1993). Наиболее полный источник информации по бутстреп-анализу – это работа Efron & Tibshirani (1998). Наконец, существует множество замечательных онлайн-ресурсов, включая Simon (1997), Canty (2002), Shah (2005) и Fox (2002).

Итоги

- Статистика повторных выборок и бутстреп-анализ – вычислительно сложные методы, позволяющие проверять гипотезы и оценивать доверительные интервалы без предположений о теоретическом распределении.
- Они особенно ценны, когда данные получены из генеральной совокупности с неизвестным распределением, когда есть се-

рьезные выбросы, когда размеры выборки малы и когда нет существующих параметрических методов, которые помогли бы проверить интересующие гипотезы.

- Они особенно интересны, потому что дают возможность ответить на вопросы, когда стандартные предположения о данных явно несостоятельны или когда нет никакого представления о том, как подойти к проблеме.
- Однако они не панацея. Они не могут превратить плохие данные в хорошие. Если исходные выборки не являются репрезентативными или слишком малы, то эти методы не помогут.

Часть IV

Методы повышенной сложности

В этой, последней части мы рассмотрим продвинутые методы статистического и графического анализа, дополняющие комплект инструментов анализа данных. Описываемые здесь методы играют важную роль в быстро развивающейся области интеллектуального анализа данных и прогнозной аналитики.

Глава 13 дополняет новыми инструментами методы регрессионного анализа, представленные в главе 8, и знакомит с параметрическими подходами к исследованию данных, распределение которых отлично от нормального. Глава начинается с описания обобщенной линейной модели, а затем смещает акцент на случаи, когда требуется предсказать зависимую переменную, являющуюся либо категориальной (логистическая регрессия), либо счетной (пуассоновская регрессия).

Обработка большого числа переменных – непростая задача из-за сложностей, свойственных многомерным данным. Глава 14 описывает два распространенных метода исследования и упрощения многомерных данных. Метод главных компонент можно использовать для преобразования большого числа коррелированных перемен-

ных в меньший набор комбинированных признаков. Факторный анализ – это совокупность методов обнаружения скрытой структуры, лежащей в основе данного набора переменных. В главе 14 даны пошаговые инструкции для выполнения этих методов анализа.

В главе 15 рассматриваются приемы анализа данных, зависящих от времени. Аналитики часто сталкиваются с необходимостью изучения тенденций и предсказания будущих событий. Сначала в этой главе будет дано подробное введение в анализ временных рядов и прогнозирование, а затем, после описания общих характеристик временных рядов, будут представлены два наиболее популярных подхода к прогнозированию (экспоненциальный и авторегрессионный).

Глава 16 посвящена кластерному анализу. Метод главных компонент и факторный анализ упрощают многомерные данные, объединяя отдельные переменные в комбинированные признаки, тогда как кластерный анализ стремится упростить многомерные данные, объединяя отдельные наблюдения в подгруппы, называемые *кластерами*. Кластеры содержат похожие наблюдения и отличающиеся от наблюдений в других кластерах. Здесь рассматриваются методы определения количества кластеров, присутствующих в наборе данных, и объединения наблюдений в эти кластеры.

Глава 17 посвящена важной теме классификации. Решая задачу классификации, аналитик стремится разработать модель для предсказания принадлежности новых наблюдений к той или иной группе (например, хороший/плохой кредитный рейтинг, доброкачественный/злокачественный, годен/не годен) на основе (потенциально большого) набора независимых переменных. Здесь рассматривается множество методов, включая логистическую регрессию, деревья решений, случайные леса и метод опорных векторов, а также несколько методов оценки эффективности полученных классификационных моделей.

На практике исследователям часто приходится иметь дело с неполными наборами данных. Глава 18 рассматривает современные подходы к распространенной проблеме пропущенных данных. В R реализован ряд изящных методов для анализа наборов данных, в которых есть значения, пропущенные по разным причинам. Здесь описаны несколько лучших подходов, а также даны рекомендации по выбору лучших методов.

К концу части IV вы будете обладать достаточными умениями, чтобы справиться со всем разнообразием сложных задач анализа данных. К ним относятся моделирование зависимых переменных с отличным от нормального распределением, работа с большим числом коррелированных переменных, уплотнение большого количества наблюдений в компактные однородные кластеры, разработка моделей для предсказания числовых или категориальных значений в будущем и приемы анализа неупорядоченных и неполных данных.

13 Обобщенные линейные модели

В этой главе:

- формулирование обобщенной линейной модели;
- предсказание значений категориальных зависимых переменных;
- моделирование счетных данных.

В главах 8 (регрессия) и 9 (дисперсионный анализ) мы исследовали линейные модели, которые можно использовать для предсказания значений зависимых переменных с нормальным распределением по значениям набора непрерывных и/или категориальных независимых переменных. Однако часто нет никаких оснований предполагать, что зависимые переменные распределены нормально (или даже непрерывны). Например:

- зависимая переменная может быть категориальной. Бинарные переменные (например, да/нет, сдал/провалился, живой/мертвый) и политомические переменные (например, плохой/хороший/превосходный, республиканец/демократ/независимый) точно не характеризуются нормальным распределением;

- зависимая переменная может быть счетной (например, число дорожно-транспортных происшествий за неделю, объем жидкости, выпиваемой за день). Такие переменные имеют ограниченное число значений и никогда не бывают отрицательными. Кроме того, их среднее и дисперсия часто связаны (это не выполняется для переменных с нормальным распределением).

Обобщенные линейные модели (generalized linear models) расширяют область применения линейных моделей, делая возможным анализ зависимых переменных с распределением, отличным от нормального.

Эту главу мы начнем с краткого обзора обобщенных линейных моделей и используемой для их построения функции `glm()`. Затем сосредоточимся на двух распространенных частных случаях – *логистической регрессии* (зависимая переменная категориальная) и *пуассоновской регрессии* (зависимая переменная счетная).

Чтобы обсуждение было более интересным, применим обобщенные линейные модели для решения двух исследовательских задач, которые непросто решить при помощи стандартных линейных моделей:

- Какие личные, демографические и психологические характеристики позволяют предсказать супружескую неверность? В этом случае зависимая переменная будет бинарной (был роман на стороне или нет).
- Какое воздействие на число припадков, произошедших за 8 недель, оказывает прием лекарств? В данном случае зависимая переменная – счетная (число припадков).

Для ответа на первый вопрос мы используем логистическую регрессию, а для ответа на второй вопрос – пуассоновскую регрессию. В процессе мы обсудим дополнительные приложения этих методов.

13.1. Обобщенные линейные модели и функция glm()

К обобщенным линейным моделям относится множество распространенных методов анализа данных. В этом разделе мы кратко рассмотрим некоторые теоретические аспекты, лежащие в основе этого метода. Если хотите, можете спокойно пропустить данный раздел и вернуться к нему впоследствии.

Представим, что нам нужно смоделировать связь между зависимой переменной Y и набором из p независимых переменных $X_1 \ldots X_p$. В стандартной линейной модели мы бы предположили, что Y имеет

нормальное распределение, а ее связь с независимыми переменными описывается таким уравнением:

$$\mu_y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j.$$

Это значит, что условное среднее зависимой переменной определяется как линейная комбинация значений независимых переменных. Параметры, определяющие ожидаемое изменение переменной Y на единицу изменения X_j, обозначены как β_j, а β_0 – это ожидаемое значение переменной Y, когда все независимые переменные равны нулю. Мы можем предсказать среднее значение распределения значений Y для наблюдений с заданным набором значений X, взвешивая и суммируя переменные X.

Обратите внимание, что здесь не делается никаких предположений о распределении значений независимых переменных X_j. Они, в отличие от Y, не обязательно должны иметь нормальное распределение. На самом деле они часто бывают категориальными (например, в дисперсионном анализе). Кроме того, допускаются нелинейные комбинации независимых переменных. Нередко в уравнение входят такие независимые переменные, как X^2 или $X_1 \times X_2$. Здесь важна линейность комбинаций параметров (β_0, β_1, ..., β_p).

При создании обобщенных линейных моделей подбираются модели вида:

$$g\left(\mu_y\right) = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$$

где $g(\mu_y)$ – это функция условного среднего (называемая *связующей функцией*). Кроме того, мы отказываемся от предположения о нормальном распределении значений Y и вместо этого подразумеваем, что Y подчиняется распределению из семейства экспоненциальных распределений. Мы задаем связующую функцию и вероятностное распределение, а параметры оцениваются при помощи итеративного алгоритма максимального правдоподобия.

13.1.1. Функция glm()

Обобщенные линейные модели обычно подбираются в R при помощи функции `glm()` (хотя доступны и другие функции). Эта функция применяется практически так же, как функция `lm()`, но имеет дополнительные параметры. Она имеет следующий синтаксис:

```
glm(formula, family=family(link=function), data=)
```

Семейства распределений вероятностей (`family`) и соответствующие связующие функции (`function`), используемые по умолчанию, перечислены в табл. 13.1.

Таблица 13.1. Параметры функции glm()

Семейства распределений	Связующая функция по умолчанию
`binomial`	`(link = "logit")`
`gaussian`	`(link = "identity")`
`gamma`	`(link = "inverse")`
`inverse.gaussian`	`(link = "1/mu^2")`
`poisson`	`(link = "log")`
`quasi`	`(link = "identity", variance = "constant")`
`quasibinomial`	`(link = "logit")`
`quasipoisson`	`(link = "log")`

Функция `glm()` позволяет подгонять разные распространенные модели, включая логистическую и пуассоновскую регрессии, а также модели для анализа выживания (здесь не рассматривается). Это можно продемонстрировать на двух первых моделях следующим образом. Предположим, что у нас есть таблица данных (`mydata`) с одной зависимой переменной (`Y`) и тремя независимыми (`X1`, `X2`, `X3`).

Логистическая регрессия подходит для дихотомических зависимых переменных (`0`, `1`). Модель предполагает, что `Y` имеет биномиальное распределение и можно подогнать линейную модель следующего вида:

$$\log_e\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \sum_{j=1}^{p}\beta_j X_j$$

где $\varpi = \mu_Y$ – это условное среднее `Y` (то есть вероятность того, что `Y = 1` при данном наборе значений X), $(\varpi/1-\varpi)$ – это отношение шансов того, что `Y = 1`, а $\log(\varpi/1-\varpi)$ – логарифм отношения шансов, или *логит* (logit; модель вероятности с логистическим распределением). В данном случае $\log(\varpi/1-\varpi)$ – это связующая функция, вероятностное распределение – биномиальное, соответственно, логистическая регрессионная модель может быть подобрана вызовом функции:

```
glm(Y~X1+X2+X3, family=binomial(link="logit"), data=mydata)
```

Логистическая регрессия более подробно описывается в разделе 13.2.

Пуассоновская регрессия применяется, если зависимая переменная – счетная. Пуассоновская регрессионная модель подразумевает, что Y имеет пуассоновское распределение, а линейная модель имеет вид:

$$\log_e(\lambda) = \beta_0 + \sum_{j=1}^{p}\beta_j X_j$$

где λ – это среднее (и дисперсия) переменной *Y*. В данном случае связующая функция имеет вид log(λ), а вероятностная функция – пуассоновская, соответственно, пуассоновская регрессионная модель может быть подобрана вызовом функции

```
glm(Y~X1+X2+X3, family=poisson(link="log"), data=mydata)
```

Более подробно пуассоновская регрессия описывается в разделе 13.3.

Стоит отметить, что стандартная линейная модель – это частный случай обобщенной линейной модели. Если связующая функция будет иметь вид $g(\mu_Y) = \mu_Y$ (тождественная функция), а распределение будет нормальным (гауссовым), то вызов

```
glm(Y~X1+X2+X3, family=gaussian(link="identity"), data=mydata)
```

даст те же результаты, что и вызов

```
lm(Y~X1+X2+X3, data=mydata)
```

Подводя итоги, можно отметить, что обобщенные линейные модели расширяют применимость стандартных линейных моделей, предлагая возможность задать *функцию* для условного среднего зависимой переменной (а не само условное среднее) и предполагая, что распределение зависимой переменной относится к семейству *экспоненциальных* распределений (а не ограничено нормальным распределением). Оценка параметров происходит методом максимального правдоподобия, а не методом наименьших квадратов.

13.1.2. Вспомогательные функции

Многие функции, которые мы использовали совместно с функцией `lm()`, когда анализировали стандартные линейные модели, имеют свои аналоги для `glm()`. Некоторые из них перечислены в табл. 13.2.

Таблица 13.2. Функции, используемые вместе с функцией glm()

Функция	Описание
`summary()`	Выводит подробные результаты для подобранной модели
`coefficients(), coef()`	Выводит параметры модели (свободный член и регрессионные коэффициенты)
`confint()`	Доверительные интервалы для параметров модели (по умолчанию 95 %)
`residuals()`	Выводит остатки подобранной модели
`anova()`	Создает таблицу дисперсионного анализа для сравнения двух моделей
`plot()`	Создает диагностические диаграммы для оценки соответствия модели данным
`predict()`	Использует подобранную модель для предсказания значений зависимой переменной в новом наборе данных

Окончание табл. 13.2

Функция	Описание
`deviance()`	Возвращает отклонение от подобранной модели
`df.residual()`	Возвращает остаточные степени свободы для подобранной модели

Примеры применения этих функций будут представлены ниже. В следующем разделе мы кратко рассмотрим способы оценки адекватности модели.

13.1.3. Соответствие модели фактическим данным и регрессионная диагностика

Проверка адекватности обобщенных линейных моделей так же важна, как и для стандартных (МНК) линейных моделей. К сожалению, среди статистиков нет единодушия в выборе способов проверки адекватности обобщенных линейных моделей. В целом можно использовать приемы, описанные в главе 8, учитывая следующие предостережения.

При оценке адекватности модели, как правило, требуется изобразить зависимость предсказанных значений зависимой переменной в исходных единицах измерения от остатков в единицах стандартного отклонения. Например, обычную диагностическую диаграмму можно создать вызовом:

```
plot(predict(model, type="response"),
    residuals(model, type= "deviance"))
```

где *model* – это объект, созданный функцией `glm()`.

Вычисляемые в R стандартизованные по Стьюденту остатки, значения показателя влияния наблюдения и D-статистика Кука будут лишь примерными. Кроме того, не существует общего согласия по выбору пороговых значений для выявления проблемных наблюдений. Решения по выбору значений этих статистик должны приниматься методом сравнения. Один из подходов заключается в том, чтобы графически изобразить значения статистики для всех наблюдений и поискать необычно большие значения. Например, вот как можно создать три диагностические диаграммы:

```
plot(hatvalues(model))
plot(rstudent(model))
plot(cooks.distance(model))
```

Или, как вариант, можно создать одну общую диаграмму:

```
library(car)
influencePlot(model)
```

Горизонтальная ось в этой диаграмме – напряженность (leverage), вертикальная ось – стьюдентизированные остатки, а размер отображаемых символов пропорционален расстоянию Кука.

Диагностические диаграммы особенно полезны, когда зависимая переменная имеет много значений. Если зависимая переменная принимает ограниченное число значений (как в случае логистической регрессии), то польза от диагностических диаграмм не особенно велика.

За более подробной информацией о регрессионной диагностике обобщенных линейных моделей обращайтесь к публикациям Fox (2008) и Faraway (2006). В оставшейся части этой главы мы подробно рассмотрим две наиболее популярные разновидности обобщенных линейных моделей: логистическую и пуассоновскую регрессии.

13.2. Логистическая регрессия

Логистическая регрессия широко используется для предсказания значений бинарной зависимой переменной по набору непрерывных и/или категориальных зависимых переменных. Для демонстрации этого подхода мы исследуем данные по супружеской неверности, содержащиеся в таблице данных `Affairs`, входящей в состав пакета `AER`. Не забудьте скачать и установить этот пакет (`install.packages("AER")`) перед первым использованием.

Данные по супружеским изменам, известные под названием «Измены Фейра»[1], основаны на перекрестном анализе, проведенном журналом *Psychology Today* в 1969 году и описанном в публикациях Greene (2003) и Fair (1978). Он содержит значения 9 переменных, зарегистрированных для 601 человека, описывающих, как часто респондент вступал во внебрачные сексуальные связи за последний год, пол и возраст опрашиваемого, сколько лет находится в браке, есть ли дети, степень религиозности (по пятибалльной шкале от 1 – атеист до 5 – очень религиозен), образование, род занятий (по семибальной классификации Холлингсхеда (Hollingshead) с обратной нумерацией) и числовую самооценку счастья в браке (от 1 – очень несчастлив до 5 – очень счастлив).

Давайте посмотрим на описательные статистики:

```
> data(Affairs, package="AER")
> summary(Affairs)
    affairs          gender         age          yearsmarried    children
 Min.   : 0.000   female:315   Min.   :17.50   Min.   : 0.125   no :171
 1st Qu.: 0.000   male  :286   1st Qu.:27.00   1st Qu.: 4.000   yes:430
 Median : 0.000                Median :32.00   Median : 7.000
 Mean   : 1.456                Mean   :32.49   Mean   : 8.178
 3rd Qu.: 0.000                3rd Qu.:37.00   3rd Qu.:15.000
 Max.   :12.000                Max.   :57.00   Max.   :15.000
 religiousness     education       occupation        rating
```

[1] Фейр (Fair) – ученый, впервые проанализировавший этот набор данных, см. ниже. – *Прим. перев.*

```
 Min.   :1.000   Min.   : 9.00   Min.   :1.000   Min.   :1.000
 1st Qu.:2.000   1st Qu.:14.00   1st Qu.:3.000   1st Qu.:3.000
 Median :3.000   Median :16.00   Median :5.000   Median :4.000
 Mean   :3.116   Mean   :16.17   Mean   :4.195   Mean   :3.932
 3rd Qu.:4.000   3rd Qu.:18.00   3rd Qu.:6.000   3rd Qu.:5.000
 Max.   :5.000   Max.   :20.00   Max.   :7.000   Max.   :5.000

> table(Affairs$affairs)
  0   1   2   3   7  12
451  34  17  19  42  38
```

Из приведенных значений видно, что 52 % респондентов – это женщины, у 72 % были дети, а медианный возраст равен 32 годам. Если говорить о зависимой переменной, то 75 % респондентов сказали, что они не имели связей на стороне за последний год (451 человек из 601). Наибольшее число супружеских измен за год составило 12 (у 6 % испытуемых).

Хотя учитывалось *абсолютное число* опрометчивых поступков, в данном случае нам интересна бинарная переменная (была измена или нет). Переменную `affairs` можно преобразовать в дихотомическую переменную `ynaffair`:

```
> Affairs$ynaffair <- ifelse(Affairs$affairs > 0, 1, 0)
> Affairs$ynaffair <- factor(Affairs$ynaffair,
                             levels=c(0,1),
                             labels=c("No","Yes"))
> table(Affairs$ynaffair)
No Yes
451 150
```

и полученный дихотомический фактор использовать как зависимую переменную в модели логистической регрессии:

```
> fit.full <- glm(ynaffair ~ gender + age + yearsmarried + children +
                  religiousness + education + occupation +rating,
                  data=Affairs, family=binomial())
> summary(fit.full)

Call:
glm(formula = ynaffair ~ gender + age + yearsmarried + children +
    religiousness + education + occupation + rating, family = binomial(),
    data = Affairs)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.571  -0.750  -0.569  -0.254   2.519

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.3773     0.8878    1.55  0.12081
gendermale     0.2803     0.2391    1.17  0.24108
age           -0.0443     0.0182   -2.43  0.01530 *
```

```
yearsmarried    0.0948     0.0322    2.94  0.00326 **
childrenyes     0.3977     0.2915    1.36  0.17251
religiousness  -0.3247     0.0898   -3.62  0.00030 ***
education       0.0211     0.0505    0.42  0.67685
occupation      0.0309     0.0718    0.43  0.66663
rating         -0.4685     0.0909   -5.15  2.6e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 675.38  on 600  degrees of freedom
Residual deviance: 609.51  on 592  degrees of freedom
AIC: 627.5

Number of Fisher Scoring iterations: 4
```

По значениям статистической ошибки первого рода (p) для коэффициентов регрессии (последний столбец) можно видеть, что пол, наличие детей, образование и род занятий не вносят значимого вклада в модель (мы не можем отвергнуть гипотезу о равенстве этих параметров нулю). Давайте подберем вторую модель, исключив эти параметры, и проверим, соответствует ли эта урезанная модель фактическим данным настолько же хорошо:

```
> fit.reduced <- glm(ynaffair ~ age + yearsmarried + religiousness +
                       rating, data=Affairs, family=binomial())
> summary(fit.reduced)
Call:
glm(formula = ynaffair ~ age + yearsmarried + religiousness + rating,
    family = binomial(), data = Affairs)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.628  -0.755  -0.570  -0.262   2.400

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)     1.9308     0.6103    3.16  0.00156 **
age            -0.0353     0.0174   -2.03  0.04213 *
yearsmarried    0.1006     0.0292    3.44  0.00057 ***
religiousness  -0.3290     0.0895   -3.68  0.00023 ***
rating         -0.4614     0.0888   -5.19  2.1e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 675.38  on 600  degrees of freedom
Residual deviance: 615.36  on 596  degrees of freedom
AIC: 625.4

Number of Fisher Scoring iterations: 4
```

Все регрессионные коэффициенты в урезанной модели статистически значимы (`p < 0.05`). Поскольку это две вложенные модели (`fit.reduced` – это часть `fit.full`), то для их сравнения можно использовать функцию `anova()`. Для обобщенных линейных моделей понадобится хи-квадрат-версия этого критерия.

```
> anova(fit.reduced, fit.full, test="Chisq")
Analysis of Deviance Table

Model 1: ynaffair ~ age + yearsmarried + religiousness + rating
Model 2: ynaffair ~ gender + age + yearsmarried + children +
    religiousness + education + occupation + rating
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1       596        615
2       592        610  4     5.85      0.21
```

Незначимая величина критерия хи-квадрат (`p = 0.21`) свидетельствует о том, что урезанная модель с четырьмя независимыми переменными соответствует данным так же хорошо, как и модель с восемью независимыми переменными. Это укрепляет нашу уверенность в том, что пол, наличие детей, образование и род занятий не оказывают существенного влияния на качество предсказания зависимой переменной. Так что можно интерпретировать данные на основании более простой модели.

13.2.1. Интерпретация параметров модели

Давайте посмотрим на регрессионные коэффициенты:

```
> coef(fit.reduced)
  (Intercept)           age  yearsmarried religiousness        rating
        1.931        -0.035         0.101        -0.329        -0.461
```

В логистической регрессии в качестве моделируемых значений зависимой переменной используется логарифм отношения шансов (odds) того, что $Y = 1$[1]. Регрессионный коэффициент – это изменение логарифма отношения шансов данной зависимой переменной на единицу изменения независимой переменной при постоянных значениях всех остальных независимых переменных.

Поскольку логарифм отношения шансов сложно интерпретировать, можно потенцировать коэффициенты, чтобы представить их в виде отношения шансов:

```
> exp(coef(fit.reduced))
  (Intercept)           age  yearsmarried religiousness        rating
        6.895         0.965         1.106         0.720         0.630
```

Теперь можно видеть, что отношение шансов внебрачных связей повышается в 1,106 раза при увеличении продолжительности

[1] То есть $\ln(P(Y = 1)/P(Y = 0))$, где P – вероятность. – *Прим. перев.*

брака на один год (при постоянных значениях возраста, религиозности и оценки счастья в браке). Отношение шансов супружеской измены нужно умножать на 0,965 при увеличении возраста испытуемых на год. Отношение шансов внебрачных связей повышается с увеличением продолжительности брака и снижается с увеличением возраста, религиозности и самооценки счастья в браке. Поскольку независимые переменные не могут быть равны нулю, то свободный член в данном случае лишен смысла.

При необходимости можно вычислить доверительные интервалы для коэффициентов при помощи функции `confint()`. Например, вызов `exp(confint(fit.reduced))` рассчитает доверительные интервалы для всех коэффициентов модели в единицах отношения шансов.

Наконец, изменение независимой переменной на одну единицу может не представлять интереса. Для бинарной логистической регрессии изменение отношения шансов того, что зависимая переменная примет более высокие значения при изменении независимой переменной на n единиц, равна $\exp(\beta_j)^n$. Если увеличение продолжительности брака на один год повышает отношение шансов супружеской измены в 1,106 раза, то увеличение продолжительности брака на 10 лет повысит отношение шансов измены в 1,106^10 = 2,7 раза при постоянных значениях остальных переменных.

13.2.2. Оценка влияния независимых переменных на вероятность исхода

Многим из нас проще размышлять в терминах вероятностей, а не отношений шансов. Чтобы понять, как изменение значений независимых переменных влияет на вероятность успешного исхода, можно использовать функцию `predict()`. Первый шаг – создание искусственного набора данных, содержащего значения интересующих независимых переменных. Затем этот набор данных можно использовать вместе с функцией `predict()` для предсказания вероятностей исхода.

Давайте используем данный подход и оценим, как влияет самооценка качества брака на вероятность супружеской измены. Сначала создадим искусственный набор данных, где возраст, продолжительность брака и религиозность представлены их средними выборочными значениями, а самооценка качества брака колеблется от 1 до 5.

```
> testdata <- data.frame(rating=c(1, 2, 3, 4, 5), age=mean(Affairs$age),
                         yearsmarried=mean(Affairs$yearsmarried),
                         religiousness=mean(Affairs$religiousness))
> testdata
  rating  age yearsmarried religiousness
1      1 32.5         8.18          3.12
2      2 32.5         8.18          3.12
```

```
3      3 32.5          8.18           3.12
4      4 32.5          8.18           3.12
5      5 32.5          8.18           3.12
```

Затем используем этот набор данных и уравнение регрессии для вычисления вероятностей:

```
> testdata$prob <- predict(fit.reduced, newdata=testdata, type="response")
  testdata
  rating  age yearsmarried religiousness  prob
1      1 32.5          8.18           3.12 0.530
2      2 32.5          8.18           3.12 0.416
3      3 32.5          8.18           3.12 0.310
4      4 32.5          8.18           3.12 0.220
5      5 32.5          8.18           3.12 0.151
```

Из полученных результатов ясно, что вероятность внебрачной связи уменьшается от 0,53, если человек считает свой брак очень несчастливым (1), до 0,15, когда брак считается очень счастливым (5) – при постоянных значениях возраста. Теперь посмотрим, как на вероятность измены влияет возраст:

```
> testdata <- data.frame(rating=mean(Affairs$rating),
                         age=seq(17, 57, 10),
                         yearsmarried=mean(Affairs$yearsmarried),
                         religiousness=mean(Affairs$religiousness))
> testdata
  rating age yearsmarried religiousness
1   3.93  17         8.18          3.12
2   3.93  27         8.18          3.12
3   3.93  37         8.18          3.12
4   3.93  47         8.18          3.12
5   3.93  57         8.18          3.12

> testdata$prob <- predict(fit.reduced, newdata=testdata, type="response")
> testdata
  rating age yearsmarried religiousness    prob
1   3.93  17         8.18          3.12   0.335
2   3.93  27         8.18          3.12   0.262
3   3.93  37         8.18          3.12   0.199
4   3.93  47         8.18          3.12   0.149
5   3.93  57         8.18          3.12   0.109
```

Как видите, с увеличением возраста от 17 до 57 лет вероятность внебрачных связей снижается от 0,34 до 0,11 при постоянных значениях остальных переменных. Используя этот подход, можно исследовать влияние каждой независимой переменной на вероятность исхода.

13.2.3. Избыточная дисперсия

Ожидаемая дисперсия для данных с биномиальным распределением рассчитывается как $\sigma^2 = n\varpi(1 - \varpi)$, где n – число наблюдений,

а ϖ – вероятность принадлежности к группе $Y = 1$[1]. *Избыточная дисперсия* (overdispersion) отмечается, когда наблюдаемая дисперсия зависимой переменной превышает ожидаемую. Избыточная дисперсия может привести к искажению оценки среднеквадратичных ошибок и некорректным значениям критериев значимости.

В случае избыточной дисперсии тоже можно проводить логистическую регрессию при помощи функции `glm()`, однако в этом случае нужно использовать квазибиномиальное распределение, а не биномиальное.

Один из способов выявить избыточную дисперсию – сравнить остаточную девиату (residual deviance)[2] с остаточными степенями свободы биномиальной модели. Если отношение

$$\phi = \frac{\text{Остаточная девиата}}{\text{Остаточные степени свободы}}$$

заметно больше единицы – это признак избыточной дисперсии. Применив этот критерий к примеру с супружескими изменами, получаем:

```
> deviance(fit.reduced)/df.residual(fit.reduced)
[1] 1.032
```

– значение, близкое к единице, что свидетельствует об отсутствии избыточной дисперсии.

Существует также критерий избыточности дисперсии. Чтобы проверить его, нужно подобрать модель дважды. Первый раз с параметром `family="binomial"`, а второй – с параметром `family="quasibinomial"`. Если предположить, что объект, возвращаемый функцией `glm()`, в первом случае сохранен в переменной `fit`, а во втором – `fit.od`, то вызов

```
pchisq(summary(fit.od)$dispersion * fit$df.residual,
       fit$df.residual, lower = F)
```

вернет значение статистической ошибки первого рода (`p`) для проверки нулевой гипотезы H_0: $\varphi = 1$, в противоположность альтернативной гипотезе H_1: $\varphi \neq 1$. Если значение `p` мало (скажем, меньше 0,05), то нулевая гипотеза отвергается.

Применив эти соображения к данным о супружеских изменах, получаем:

```
> fit <- glm(ynaffair ~ age + yearsmarried + religiousness +
             rating, family = binomial(), data = Affairs)
> fit.od <- glm(ynaffair ~ age + yearsmarried + religiousness +
                rating, family = quasibinomial(), data = Affairs)
```

[1] Вероятность «благоприятного исхода». – *Прим. перев.*

[2] Общепринятый перевод термина deviance (обозначающего величину, производную от оценки максимального правдоподобия) на русский язык отсутствует. – *Прим. перев.*

```
> pchisq(summary(fit.od)$dispersion * fit$df.residual,
         fit$df.residual, lower = F)

[1] 0.34
```

Полученное значение p (0.34) совершенно точно незначимо (p > 0.05), утверждая нас во мнении, что избыточная дисперсия в данном случае не представляет проблемы. Мы вернемся к вопросу об избыточной дисперсии, когда будем обсуждать пуассоновскую регрессию.

13.2.4. Дополнительные методы

В R реализовано несколько дополнительных методов и разновидностей логистической регрессии:

- *устойчивая (robust) логистическая регрессия.* Функцию glmRob() из пакета robust можно использовать для подгонки устойчивых обобщенных регрессионных моделей, в том числе и устойчивой логистической регрессии. Этот метод может пригодиться при подборе логистических регрессионных моделей для данных с выбросами и влиятельными наблюдениями;
- *мультиномиальная (multinomial) логистическая регрессия.* Если зависимая переменная имеет больше двух неупорядоченных значений (например, женат/вдов/разведен), можно рассчитать мультиномиальную логистическую регрессию при помощи функции mlogit() из пакета mlogit или ultinom() из пакета nnet;
- *порядковая (ordinal) логистическая регрессия.* Если зависимая переменная представлена упорядоченным фактором (например, кредитный рейтинг: низкий/средний/высокий), то можно использовать порядковую логистическую регрессию (функция polyr() из пакета MASS).

Возможность моделировать категориальную зависимую переменную со многими значениями (и упорядоченными, и неупорядоченными) – важное расширение применимости метода, но оно имеет свою цену – бóльшую сложность интерпретации. Оценка соответствия модели данным и регрессионная диагностика в этих случаях тоже будут более сложными.

В примере с супружеской неверностью число внебрачных связей было преобразовано в дихотомическую (да/нет) переменную, поскольку нас интересовало наличие измен за текущий год. Если бы мы больше интересовались величиной эффекта – числом внебрачных связей за год, мы бы напрямую анализировали счетные данные. Один из распространенных подходов к анализу счетных данных – пуассоновская регрессия – следующая тема, к которой мы обратимся.

13.3. Пуассоновская регрессия

Пуассоновская регрессия используется, когда нужно предсказать значение счетной зависимой переменной по набору непрерывных и/или категориальных независимых переменных. Всеобъемлющее и при этом доступное введение в пуассоновскую регрессию можно найти в работе Coxe, West и Aiken (2009).

Для демонстрации процесса подгонки пуассоновской регрессионной модели, наряду с некоторыми проблемами, которые могут возникнуть при таком анализе данных, используем опубликованные данные о припадках (Breslow, 1993), входящие в состав пакета `robustbase`. А именно рассмотрим влияние приема лекарств от эпилептических припадков на число припадков, которые случились за 8 недель, прошедших со времени начала лечения. Не забудьте установить пакет `robustbase`, перед тем как продолжить.

Для пациентов, которые страдали от простых или сложных припадков, были получены данные о возрасте и о числе припадков за 8 недель до и после начала лечения. Больные были случайным образом распределены на две группы, которые получали плацебо (placebo) или настоящее лекарство (progabide). Переменная `Ysum` (число припадков за 8 недель после начала лечения) – это зависимая переменная. Тип лечения (`Trt`), возраст в годах (`Age`) и число припадков за 8 недель до начала лечения (`Base`) – независимые переменные. Возраст и число припадков до начала лечения включены в анализ, потому что они могут оказывать влияние на зависимую переменную. Нам интересно узнать, можно ли утверждать, что применение лекарства снижает число припадков с учетом влияния этих ковариат.

Сначала посмотрим на описательные статистики для этого набора данных:

```
> data(epilepsy, package="robustbase")
> names(epilepsy)
 [1] "ID"    "Y1"    "Y2"    "Y3"    "Y4"    "Base"  "Age"   "Trt"   "Ysum"
[10] "Age10" "Base4"

> summary(breslow.dat[6:9])
      Base            Age              Trt          Ysum
 Min.   :  6.0   Min.   :18.0   placebo  :28   Min.   :  0.0
 1st Qu.: 12.0   1st Qu.:23.0   progabide:31   1st Qu.: 11.5
 Median : 22.0   Median :28.0                  Median : 16.0
 Mean   : 31.2   Mean   :28.3                  Mean   : 33.1
 3rd Qu.: 41.0   3rd Qu.:32.0                  3rd Qu.: 36.0
 Max.   :151.0   Max.   :42.0                  Max.   :302.0
```

Обратите внимание, что хотя набор данных включает 12 переменных, мы сосредоточили свое внимание только на четырех упомянутых выше. Распределение значений числа припадков до и пос-

ле лечения характеризуется заметной асимметрией. Давайте проанализируем зависимую переменную подробнее. Следующий код выводит диаграмму, представленную на рис. 13.1.

```
library(ggplot2)
   ggplot(epilepsy, aes(x=Ysum)) +
  geom_histogram(color="black", fill="white") +
  labs(title="Distribution of seizures",
       x="Seizure Count",
       y="Frequency") +
  theme_bw()
ggplot(epilepsy, aes(x=Trt, y=Ysum)) +
  geom_boxplot() +
  labs(title="Group comparisons", x="", y="") +
  theme_bw()
```

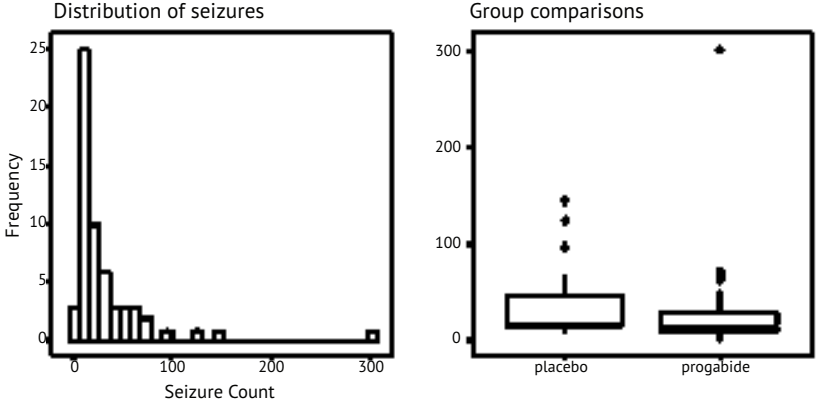


Рис. 13.1. Распределение значений числа припадков после лечения (источник данных: опубликованные данные о припадках (Breslow, 1993))

Хорошо заметно асимметричное распределение значений зависимой переменной, а также возможное наличие выбросов. На первый взгляд, число припадков у людей, принимающих настоящее лекарство, кажется меньше и имеет меньший разброс данных. Для данных с пуассоновским распределением можно ожидать, что меньшая дисперсия будет сопряжена с меньшим средним значением. Гетерогенность дисперсии[1] не представляет проблемы для пуассоновской регрессии, в отличие от стандартной МНК-регрессии.

Следующий шаг – подгонка пуассоновской регрессионной модели:

```
> fit <- glm(Ysum ~ Base + Age + Trt, data=epilepsy, family=poisson())
> summary(fit)

Call:
```

[1] Межгрупповые различия дисперсии. – *Прим. перев.*

```
glm(formula = Ysum ~ Base + Age + Trt, family = poisson(), data = epilepsy)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-6.057  -2.043  -0.940   0.793  11.006

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.948826   0.135619   14.37  < 2e-16 ***
Base         0.022652   0.000509   44.48  < 2e-16 ***
Age          0.022740   0.004024    5.65  1.6e-08 ***
Trtprogabide -0.152701   0.047805   -3.19   0.0014 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 2122.73  on 58  degrees of freedom
Residual deviance:  559.44  on 55  degrees of freedom
AIC: 850.7

Number of Fisher Scoring iterations: 5
```

На экран выводятся девиаты, параметры регрессионной модели, среднеквадратичные отклонения и критерии, проверяющие предположение о неравенстве параметров нулю. Обратите внимание, что влияние каждой независимой переменной значимо на уровне `p < 0.05`.

13.3.1. Интерпретация параметров модели

Коэффициенты модели можно получить вызовом функции `coef()` или увидеть в составе таблицы `Coefficients`, создаваемой функцией `summary()`:

```
> coef(fit)
 (Intercept)         Base          Age Trtprogabide
      1.9488       0.0227       0.0227      -0.1527
```

В пуассоновской регрессии зависимая переменная моделируется как логарифм условного среднего $\log_e(\lambda)$. Регрессионный коэффициент при переменной `Age` (возраст), равный 0,0227, показывает, что каждый дополнительный год возраста при постоянных значениях числа припадков до лечения и способа лечения сопряжен с увеличением логарифма среднего значения числа припадков на 0,03. Поскольку не бывает нулевого возраста и никто из участников эксперимента не имел нулевого числа припадков до лечения, свободный коэффициент не имеет смысла.

Обычно интерпретировать регрессионные коэффициенты гораздо проще, когда они выражены в исходных единицах зависимой переменной (число припадков, а не логарифм этого числа).

Для этого коэффициенты нужно потенцировать:

```
> exp(coef(fit))
 (Intercept)         Base          Age Trtprogabide
       7.020        1.023        1.023        0.858
```

Теперь видно, что увеличение возраста на один год *увеличивает* ожидаемое число припадков в 1,023 раза. Это значит, что с возрастом число припадков увеличивается. Более важно, что изменение типа лечения на единицу (то есть переход от плацебо к настоящему лекарству) уменьшает число припадков в 0,86 раза, т. е. можно ожидать снижения числа припадков на 14 % у пациентов, принимающих настоящее лекарство, по сравнению с теми, кто принимает плацебо, при постоянных значениях возраста и числа припадков до лечения.

Важно помнить, что, как и экспоненциальные параметры в логистической регрессии, экспоненциальные параметры в пуассоновской регрессии оказывают мультипликативный, а не аддитивный эффект на зависимую переменную. Так же как и в логистической регрессии, нужно проверить модель на наличие избыточной дисперсии.

13.3.2. Избыточная дисперсия

В пуассоновском распределении дисперсия и среднее равны. Избыточная дисперсия при пуассоновской регрессии отмечается, когда наблюдаемая дисперсия зависимой переменной больше, чем ожидается, исходя из свойств распределения. Поскольку избыточная дисперсия часто встречается при работе со счетными данными и оказывает неблагоприятный эффект на интерпретацию результатов, мы потратим некоторое время на обсуждение этой проблемы.

Есть несколько причин, почему может возникать избыточная дисперсия (Coxe et al., 2009):

- отсутствие важной независимой переменной;
- явление, известное под названием *«зависимость от состояния»*. События считаются независимыми друг от друга. Для числа припадков это будет значить, что для любого пациента вероятность припадка не зависит от вероятности других припадков. Однако это допущение часто не выполняется. Не похоже, что для данного пациента вероятность первого припадка равна вероятности 40-го припадка, если их уже было 39;
- при продолжительных исследованиях избыточная дисперсия может быть вызвана группировкой данных, свойственной повторным измерениям. Мы не будем обсуждать здесь пуассоновские модели для продолжительных исследований.

Если избыточная дисперсия имеет место, но она не учитывается в модели, есть риск получить слишком низкие среднеквадратичные ошибки и доверительные интервалы, а критерии достоверно-

сти будут слишком либеральными (т. е. можно найти закономерности там, где их на самом деле нет).

Как и в случае логистической регрессии, признак избыточной дисперсии – это отношение остаточной девиаты к числу степеней свободы остатков, заметно превышающее единицу. Для наших данных по припадкам это отношение составляет:

```
> deviance(fit)/df.residual(fit)
[1] 10.17
```

Очевидно, что эта величина значительно больше единицы.

Пакет `qcc` позволяет вычислить критерий избыточности дисперсии для пуассоновского распределения (не забудьте скачать и установить этот пакет перед первым использованием). Вот как можно вычислить этот критерий для данных по припадкам:

```
> library(qcc)
> qcc.overdispersion.test(breslow.dat$sumY, type="poisson")
Overdispersion test Obs.Var/Theor.Var Statistic p-value
       poisson data              62.9      3646       0
```

Неудивительно, что значение `p` оказалось меньше 0,05, ясно указывая на наличие избыточной дисперсии.

Вы по-прежнему можете подбирать модели для ваших данных при помощи функции `glm()`, заменив параметр `family="poisson"` на `family="quasipoisson"`, следуя аналогии с логической регрессией, имеющей избыточную дисперсию:

```
> fit.od <- glm(sumY ~ Base + Age + Trt, data=breslow.dat,
                family=quasipoisson())
> summary(fit.od)

Call:
glm(formula = sumY ~ Base + Age + Trt, family = quasipoisson(),
    data = breslow.dat)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-6.057  -2.043  -0.940   0.793  11.006

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.94883    0.46509    4.19  0.00010 ***
Base          0.02265    0.00175   12.97  < 2e-16 ***
Age           0.02274    0.01380    1.65  0.10509
Trtprogabide -0.15270    0.16394   -0.93  0.35570
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 11.8)

    Null deviance: 2122.73  on 58  degrees of freedom
```

```
Residual deviance:  559.44  on 55  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5
```

Обратите внимание, что оценки параметров при использовании квазипуассоновского распределения идентичны полученным для пуассоновской регрессии, но стандартные ошибки намного больше. В данном случае бóльшие стандартные ошибки привели к тому, что значения `p` для переменных `Trt` и `Age` превысили 0,05. Таким образом, если учесть избыточную дисперсию, у нас недостаточно оснований утверждать, что настоящее лекарство снижает число припадков эффективнее, чем плацебо, при постоянных значениях числа припадков до лечения и возраста.

Имейте в виду, что этот пример приведен здесь только для демонстрации метода. Представленные результаты не должны использоваться для каких-либо утверждений относительно эффективности лекарств в реальном мире. Я не доктор – по крайней мере, не доктор медицинских наук – и даже не изображал его в телепередачах.

Мы закончим исследование пуассоновской регрессии обсуждением нескольких ее важных разновидностей и дополнительных методов.

13.3.3. Дополнительные методы

В R реализовано несколько дополнительных методов и разновидностей пуассоновской регрессии, включая модели, которые позволяют изменять периоды времени, вводящие поправку на избыточное число нулей и устойчивые модели, которые полезны, когда данные содержат выбросы и влиятельные наблюдения. Я опишу каждый из них отдельно.

Пуассоновская регрессия с варьирующими временными периодами

Наше описание пуассоновской регрессии было ограничено зависимыми переменными, которые хранят число событий за фиксированный отрезок времени (например, число припадков за 8 недель, число дорожно-транспортных происшествий за прошедший год, число хороших поступков за день). Длина временно́го отрезка в этом случае остается постоянной. Однако есть возможность подгонять также регрессионные модели, допускающие различия во временны́х отрезках, за которые происходят события. В таком случае зависимая переменная будет представлена частотой событий за единицу времени.

Для анализа частот нужно добавить переменную (которая будет называться, например, `time`), хранящую продолжительность временно́го отрезка, за который произошли события, отмеченные за данное наблюдение. В таком случае модель

$$\log_e(\lambda) = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$$

должна быть заменена моделью

$$\log_e\left(\frac{\lambda}{time}\right) = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$$

или, что то же самое:

$$\log_e(\lambda) = \log_e(time) + \beta_0 + \sum_{j=1}^{p} \beta_j X_j.$$

Для подгонки этой новой модели функции `glm()` нужно передать параметр `offset`. Предположим для примера, что период времени, в течение которого пациенты получали лекарство от эпилептических припадков, варьировался от 14 до 60 суток. Мы можем использовать относительную частоту припадков как зависимую переменную (при условии что для каждого пациента фиксировалась продолжительность периода лечения в сутках) и подогнать модель:

```
fit <- glm(Ysum ~ Base + Age + Trt, data=epilepsy,
           offset= log(time), family=poisson)
```

где `Ysum` – это число припадков, случившихся за время лечения. В этом случае предполагается, что относительная частота припадков оставалась постоянной (то есть два припадка за четыре дня – это то же самое, что 10 припадков за 20 дней).

Пуассоновская регрессия для данных с избыточным числом нулей

Бывает так, что частота наблюдений с нулевым числом событий в наборе данных выше, чем предсказывает пуассоновская модель. Такое может случиться, если в одной из подгрупп генеральной совокупности регистрируемые события никогда не происходят. Например, в наборе данных по супружеским изменам, который обсуждался в разделе о логистической регрессии, исходная зависимая переменная (`affairs`) была выражена числом внебрачных связей за прошедший год. Весьма вероятно, что существует группа верных супругов, которые никогда не совершат измену, вне зависимости от продолжительности времени наблюдений. Такие наблюдения называются *структурными нулями* (в основном людьми свободных нравов из числа испытуемых).

В подобных случаях можно анализировать данные при помощи подхода, который называется *пуассоновская регрессия для данных с избыточным числом нулей* (zero-inflated Poisson regression). В рамках этого подхода одновременно подгоняются две модели – одна предсказывает, кто будет или не будет изменять, а другая – сколько раз вступят в связи на стороне те участники, для которых зарегистрирована хотя бы одна измена. Это можно интерпретировать как мо-

дель, которая объединяет логистическую регрессию (для предсказания структурных нулей) и пуассоновскую регрессию (для предсказания числа наблюдений, которые не являются структурными нулями). Такой подход можно реализовать при помощи функции `zeroinfl()` из пакета `pscl`.

Устойчивая пуассоновская регрессия

Наконец, для подбора устойчивой обобщенной линейной модели, включая устойчивую пуассоновскую регрессию, можно использовать функцию `glmRob()` из пакета `robustbase`. Как уже отмечалось выше, это может пригодиться, когда в данных имеется большое количество выбросов и влиятельных наблюдений.

Дополнительные сведения

Обобщенные линейные модели – сложный метод с замысловатым математическим аппаратом, однако существует много прекрасных источников информации для более детального знакомства с ним. Хорошее короткое введение можно найти в работе Dunteman & Ho (2006). Классическое и непростое описание обобщенных линейных моделей представлено в McCullagh & Nelder (1989). Исчерпывающее и доступное введение можно найти в Dobson & Barnett (2008) и Fox (2008). Прекрасное введение в тему в контексте использования R можно найти в Faraway (2006) и Fox (2002).

Итоги

- Обобщенные линейные модели позволяют анализировать зависимые переменные, распределение которых явно отличается от нормального, включая категориальные и дискретные величины.
- Логистическую регрессию можно использовать для анализа результатов исследований с дихотомическим результатом (да/нет).
- Пуассоновскую регрессию можно использовать для анализа результатов исследований, когда зависимые переменные имеют количественное или частотное выражение.
- Диагностика регрессии для обобщенных линейных моделей нередко оказывается сложнее, чем для линейных моделей, описанных в главе 8. В частности, модели логистической и пуассоновской регрессий обязательно следует оценивать на наличие избыточной дисперсии и при ее обнаружении попробовать применить альтернативное распределение ошибок, такое как квазибиномиальное или квазипуассоновское, при подгонке модели.

14 Метод главных компонент и факторный анализ

В этой главе:

- метод главных компонент;
- факторный анализ;
- другие модели со скрытыми переменными.

Один из наиболее затруднительных аспектов многомерных данных – очевидная сложность информации. Осмыслить взаимосвязи между сотней переменных в наборе данных практически невозможно! Даже 20 переменных имеют 190 попарных корреляций, которые нужно проанализировать, чтобы понять, как отдельные переменные связаны друг с другом. К счастью, существуют две связанные между собой, но разные методологии исследования и упрощения сложных многомерных данных – метод главных компонент и факторный анализ.

Метод главных компонент (Principal Components Analysis, PCA) позволяет уменьшить размерность данных и преобразовать большое число коррелированных переменных в гораздо меньший набор некоррелированных переменных, называемых *главными компонентами*. Напри-

мер, метод главных компонент можно использовать для преобразования 30 коррелированных (и возможно, избыточных) характеристик состояния окружающей среды в пять некоррелированных комбинированных переменных, сохраняющих максимально возможную часть информации, которая содержалась в исходном наборе переменных.

Помимо МГК, существует также группа методов, объединенных общим названием *разведочный факторный анализ* (Exploratory Factor Analysis, EFA), или просто ФА, которые помогают обнаружить скрытую структуру в имеющемся наборе переменных. Этот анализ позволяет найти меньший набор *скрытых* переменных, способных объяснить взаимосвязи между наблюдаемыми, или *явными*, переменными. Например, набор данных `Harman74.cor` содержит корреляции между результатами 24 психологических тестов, которые были предложены 145 школьникам 7-х и 8-х классов. Если применить ФА к этим данным, то в результате окажется, что имеющиеся 276 парных корреляций между результатами разных тестов можно объяснить четырьмя факторами (развитость речи, скорость обработки информации, способность к логическим умозаключениям и память). 24 психологических теста содержат наблюдаемые, или явные, переменные, а четыре фактора, или скрытые переменные, выводятся из корреляций между этими наблюдаемыми переменными.

Различия между МГК и ФА показаны на рис. 14.1. Главные компоненты (PC1 и PC2 на рис. 14.1) – это линейные комбинации наблюдаемых переменных (с X1 по X5). Веса, используемые для формирования линейных комбинаций переменных, подбираются так, чтобы максимизировать долю дисперсии, объясняемую каждой главной компонентой, при этом главные компоненты не коррелированы между собой, т. е. ортогональны.

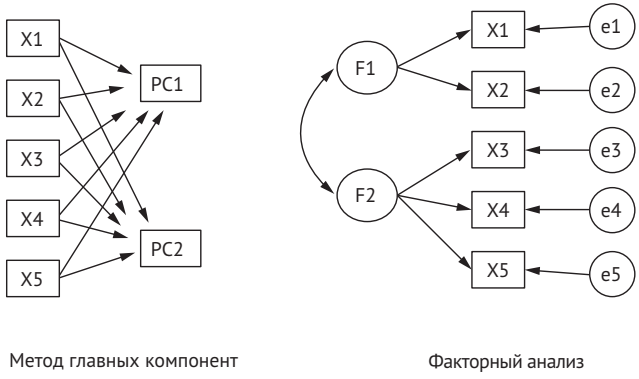


Рис. 14.1. Сравнение моделей, полученных с помощью метода главных компонент и факторного анализа. На схемах показаны наблюдаемые переменные (с X1 по X5), главные компоненты (PC1 и PC2), факторы (F1 и F2) и ошибки (с e1 по e5)

В факторном анализе, напротив, предполагается, что факторы (F1 и F2 на рис. 14.1) лежат в основе, или «обусловливают» наблюдаемые

переменные, а не являются их линейными комбинациями. Ошибки (с e1 по e5 на рис. 14.1) – это не учтенная факторами дисперсия наблюдаемых переменных[1]. Кружки означают, что факторы и ошибки не наблюдаются непосредственно, а выводятся из корреляций между переменными. В данном примере изогнутая стрелка между факторами означает, что они ортогональны. Корреляция между факторами – обычное дело для моделей ФА, но не обязательна.

Для получения устойчивых результатов при помощи описанных в этой главе методов нужны большие выборки[2]. Решить, какой размер выборки достаточен, несколько затруднительно. До последнего времени использовались эмпирические правила, например: «в факторном анализе число объектов должно превышать число переменных в 5–10 раз». Недавние исследования показали, что необходимый размер выборки зависит от числа факторов, числа сопряженных с каждым фактором переменных и от того, насколько хорошо набор факторов объясняет дисперсию переменных (Bandalos & Boehm-Kaufman, 2009). Я возьму на себя смелость сказать, что если у вас есть несколько сотен наблюдений, то вам, скорее всего, не о чем беспокоиться. В этой главе мы будем рассматривать небольшие, искусственно подобранные задачи, исключительно ради экономии места.

Для начала мы познакомимся с функциями в языке R, при помощи которых можно провести анализ методом главных компонент, или факторный анализ, и кратко рассмотрим необходимые этапы анализа. Затем тщательно разберем два примера применения МГК, за которыми последует расширенный пример применения ФА. В конце главы будут перечислены некоторые пакеты, которые можно использовать для подбора моделей со скрытыми переменными. Обсуждение коснется пакетов для конфирматорного (confirmatory) факторного анализа, моделирования структурных уравнений, анализа соответствий (correspondence analysis) и анализа скрытых (или латентных) классов.

14.1. Поддержка метода главных компонент и факторного анализа в R

В стандартном дистрибутиве R поддержка МГК и ФА реализована в виде функций `princomp()` и `factanal()` соответственно. В этой гла-

[1] Число выделяемых факторов всегда меньше числа исходных переменных, поэтому ошибки в МГК – это дисперсия, заключенная в «отброшенных» компонентах. Каким бы способом ни строилась новая ортогональная система координат, когда число ее осей станет равным или превысит число исходных переменных в выборке, никаких ошибок просто не останется, потому что данные полностью «поместятся» в построенном пространстве. – *Прим. перев.*

[2] Способы бутстреп-оценки, так хорошо описанные в данной книге, полностью применимы для оценки устойчивости полученного решения и вычисления интервальных оценок показателей полученной модели. – *Прим. перев.*

ве мы сосредоточимся на описании функций, реализованных в пакете psych. Эти функции предлагают намного более широкие возможности, чем их стандартные аналоги. Кроме того, эти функции выводят результаты в формате, более привычном для социологов и более сходном со способом представления подобных результатов в других статистических программах, таких как SAS и SPSS.

Функции, имеющиеся в пакете psych и соответствующие обсуждаемой теме, перечислены в табл. 14.1. Не забудьте установить этот пакет перед началом работы с примерами в этой главе.

Таблица 14.1. Функции поддержки факторного анализа в пакете psych

Функция	Описание
`principal()`	Анализ главных компонент с возможностью поворота осей[1]
`fa()`	Факторный анализ методом главных осей, минимальных остатков, взвешенных наименьших квадратов или наибольшего правдоподобия
`fa.parallel()`	График собственных значений (scree plot) с параллельным анализом
`factor.plot()`	Графическое представление результатов факторного анализа или метода главных компонент
`fa.diagram()`	Графическое представление матриц нагрузок в факторном анализе или методе главных компонент
`scree()`	Диаграмма собственных значений в факторном анализе или методе главных компонент

Факторный анализ (и в меньшей степени метод главных компонент) часто сбивает с толку неискушенных пользователей, потому что под этими названиями скрывается большое разнообразие методов, и чтобы достигнуть результата при помощи каждого из них, нужно преодолеть несколько этапов (и принять несколько решений). Вот эти этапы:

1. Подготовить данные. Результаты МГК и ФА выводятся из корреляций между наблюдаемыми переменными. Пользователи могут передавать функциям `principal()` и `fa()` либо исходные таблицы данных, либо корреляционные матрицы. При передаче исходных данных корреляционная матрица вычисляется автоматически. Не забудьте перед обработкой проверить данные на наличие пропущенных значений. По умолчанию пакет psych использует попарное удаление при вычислении корреляций.
2. Выбрать факторную модель, которая лучше подходит для исследовательских задач – МГК (снижение размерности данных)

[1] По своей сути это процедура ФА, проводимого с первоначальным построением осей-факторов с использованием МГК. Поскольку данный способ построения системы ортогональных осей, описывающих экспериментальные данные, наиболее обоснован математически, он вынесен в отдельную процедуру. – *Прим. перев.*

или ФА (выявление скрытой структуры). При выборе ФА нужно также выбрать определенный метод (например, наибольшего правдоподобия).

3. Решить, сколько компонент/факторов выделять.
4. Выделить компоненты/факторы.
5. Повернуть компоненты/факторы.
6. Интерпретировать результаты.
7. Вычислить оценки для компонент или факторов.

В оставшейся части этой главы мы внимательно рассмотрим каждый из этих шагов, начав с МГК. В конце главы вы найдете блок-схему с возможными этапами МГК/ФА (рис. 14.7). Вы легко разберетесь в ней после знакомства с предшествующим ей материалом.

14.2. Главные компоненты

Задача МГК – заменить большое число коррелированных переменных меньшим числом некоррелированных (ортогональных) переменных, сохраняющих как можно больше информации, содержавшейся в исходных переменных. Производные переменные, называемые *главными компонентами*, – это линейные комбинации наблюдаемых переменных. В частности, первая главная компонента

$$PC_1 = a_1X_1 + a_2X_2 + ... + a_kX_k$$

– это взвешенная комбинация k наблюдаемых переменных, которая учитывает наибольшую долю дисперсии (изменчивости) исходного набора переменных. Вторая главная компонента – это линейная комбинация, учитывающая наибольшую дисперсию исходного набора переменных, при условии что она *ортогональна* первой главной компоненте (то есть не коррелирует с ней). Каждая следующая компонента учитывает наибольшую возможную дисперсию, оставаясь ортогональной к остальным компонентам. Теоретически можно выделить столько же главных компонент, сколько имеется переменных. Однако, исходя из практических соображений, исследователи обычно стремятся охарактеризовать полный набор переменных гораздо меньшим набором компонент. Рассмотрим простой пример.

Набор данных `USJudgeRatings` содержит рейтинг судей разных штатов главного суда первой инстанции в США, составленный адвокатами. Таблица данных содержит 12 числовых переменных, характеризующих 43 судьи. Названия переменных расшифрованы в табл. 14.2.

Таблица 14.2. Переменные в наборе данных USJudgeRatings

Переменная	Описание	Переменная	Описание
CONT	Число контактов адвоката с судьей	PREP	Подготовка к судебному процессу
INTG	Непредвзятость судьи	FAMI	Знание законов
DMNR	Поведение	ORAL	Верные устные решения
DILG	Старание	WRIT	Верные письменные решения
CFMG	Качество организации процесса вынесения судебных решений	PHYS	Физические возможности
DECI	Быстрота решений	RTEN	Достоин запоминания

Возникает практический вопрос: можно ли описать оценки по этим 11 параметрам (от INTG до RTEN), используя меньший набор комбинированных переменных? И если да, то сколько таких переменных понадобится и как их определить? Поскольку наша цель – упрощение данных, мы решим эту задачу при помощи МГК. Данные представлены в виде числовых значений – баллов, пропущенные значения отсутствуют. Остается только решить, сколько главных компонент понадобится.

14.2.1. Выбор числа главных компонент

Существует несколько критериев для определения числа компонент в МГК:

- имеющийся опыт и теоретические соображения;
- порог объясняемой доли дисперсии исходных переменных (например, 80 %);
- изучение собственных значений матрицы корреляций между всеми переменными.

Наиболее распространенный подход основан на изучении собственных значений (eigenvalues). Каждая компонента связана с собственным значением корреляционной матрицы. Первой главной компоненте (Principal Component, PC) соответствует наибольшее собственное значение, второй компоненте – второе по величине собственное значение и т. д. Согласно *критерию Кайзера–Харриса* (Kaiser-Harris), следует использовать компоненты, собственные значения которых превышают единицу. Компоненты с собственными значениями меньше единицы объясняют меньше дисперсии, чем содержится в одной исходной переменной. В *критерии собственных значений* (Cattell Scree test) собственные значения отображаются на диаграмме с номерами компонент. На подобных графиках (иногда называемых графиками собственных значений,

или каменистой осыпи, scree plot) обычно виден изгиб, и используются компоненты до этого изгиба. Наконец, можно заниматься моделированием, выделяя компоненты из случайных матриц данных той же размерности, что и исходная матрица. Если собственное значение, основанное на реальных данных, выше соответствующих усредненных собственных значений для набора случайных матриц данных, тогда такая компонента считается достойной для использования. Подобный подход называется *параллельным анализом* (более подробную информацию можно найти в публикации Hayton, Allen & Scarpello, 2004).

Все три вида анализа собственных значений можно выполнить одновременно при помощи функции `fa.parallel()`. Вот пример для наших 11 переменных (без переменной `CONT`):

```
library(psych)
fa.parallel(USJudgeRatings[,-1], fa="pc", n.iter=100,
            show.legend=FALSE, main="Scree plot with parallel analysis")
abline(h=1)
```

Этот код создаст график, изображенный на рис. 14.2, отображающий значения критерия собственных значений (обозначены отрезками и крестиками), усредненных собственных значений, полученных из 100 случайных таблиц данных (пунктир) и критерия превышения единицы собственными значениями (горизонтальная линия $y = 1$).

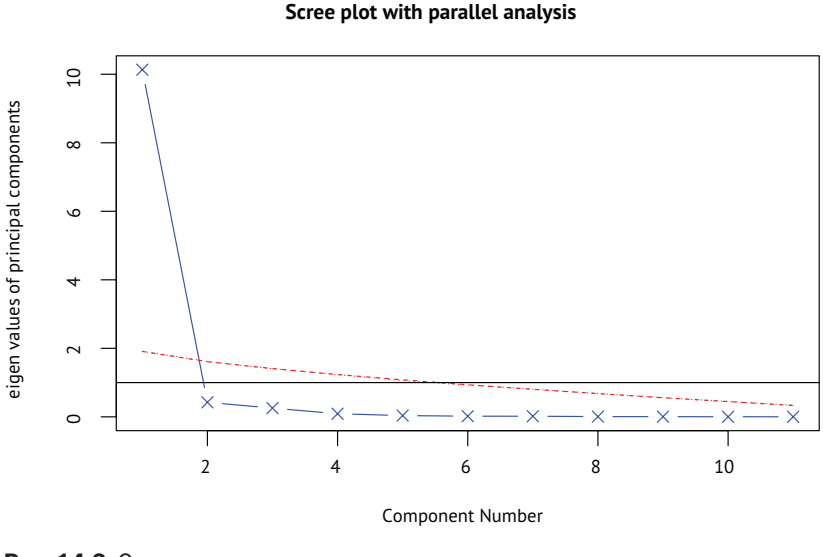


Рис. 14.2. Оценка числа главных компонент, достаточного для характеристики данных из примера про американских судей. График собственных значений (линия с крестиками), критерия превышения единицы собственными значениями (горизонтальная линия) и параллельный анализ со 100 симуляциями (пунктир) свидетельствуют о том, что достаточно использовать одну компоненту

Все три критерия свидетельствуют о том, что для характеристики этого набора данных достаточно одной компоненты. Следующий шаг – выделить эту компоненту при помощи функции `principal()`.

14.2.2. Выделение главных компонент

Как отмечалось выше, метод главных компонент можно применить к матрице с исходными данными или корреляционной матрице при помощи функции `principal()`. Она имеет следующий синтаксис:

```
principal(r, nfactors=, rotate=, scores=),
```

где

- `r` – корреляционная матрица или матрица с исходными данными;
- `nfactors` – число главных компонент, которые нужно выделить (по умолчанию одна);
- `rotate` – определяет, какой тип поворота применить (по умолчанию варимакс, см. раздел 14.2.3);
- `scores` – определяет необходимость вычисления оценок главных компонент (по умолчанию – нет).

В листинге 14.1 показано, как выделить первую главную компоненту.

Листинг 14.1. Применение метода главных компонент к данным с рейтингами американских судей

```
> library(psych)
> pc <- principal(USJudgeRatings[,-1], nfactors=1)
> pc

Principal Components Analysis
Call: principal(r = USJudgeRatings[, -1], nfactors=1)
Standardized loadings based upon correlation matrix
      PC1   h2    u2
INTG 0.92 0.84 0.157
DMNR 0.91 0.83 0.166
DILG 0.97 0.94 0.061
CFMG 0.96 0.93 0.072
DECI 0.96 0.92 0.076
PREP 0.98 0.97 0.030
FAMI 0.98 0.95 0.047
ORAL 1.00 0.99 0.009
WRIT 0.99 0.98 0.020
PHYS 0.89 0.80 0.201
RTEN 0.99 0.97 0.028

                 PC1
SS loadings    10.13
Proportion Var  0.92
[... additional output omitted ...]
```

Здесь метод главных компонент применяется к исходной таблице данных без переменной CONT. Согласно аргументам, функция должна выделить одну главную компоненту без поворота (вращение компонент обсуждается в разделе 14.2.3). Поскольку МГК применяется к корреляционной матрице, перед извлечением компонент функция автоматически вычисляет корреляции по исходным данным.

Столбец PC1 содержит *нагрузки* компоненты – корреляции наблюдаемых переменных с главной компонентой (или главными компонентами). Если бы мы извлекали несколько главных компонент, то в выводе присутствовали бы также столбцы PC2, PC3 и т. д. Нагрузки компонент используются для интерпретации их значений. Судя по результатам, все переменные сильно коррелируют с первой компонентой (PC1). Поэтому ее можно использовать как общий показатель положения судей в рейтинге.

В столбце h2 указаны *общности* компонент (component communalities) – доля дисперсии каждой переменной, объясняемая данной компонентой. В столбце u2 перечислены величины уникальности компонент (component uniquenesses) – доля дисперсии каждой переменной, которая остается не учтенной данной компонентой (1 - h2). Например, первая компонента объясняет 80 % дисперсии рейтинга физических возможностей судей (PHYS), а 20 % – нет. Эта переменная хуже всего объясняется нашей единственной главной компонентой.

В строке SS Loadings приводятся собственные значения компонент. Собственные значения – это стандартизированная дисперсия, связанная с данной компонентой (в этом случае ее значение для первой компоненты составляет 10.13). Наконец, в строке Proportion Var указана доля дисперсии, объясняемая каждой компонентой. В данном случае видно, что первая главная компонента объясняет 92 % дисперсии 11 переменных.

Рассмотрим второй пример, извлекающий несколько главных компонент. Набор данных Harman23.cor содержит данные о восьми параметрах тела у 305 девочек. В нашем случае набор данных представлен в виде корреляционной матрицы (табл. 14.3).

Таблица 14.3. Корреляции между параметрами тела у 305 девочек (набор данных Harman23.cor)

	height (рост)	arm span (размах рук)	forearm (предплечье)	lower leg (голень)	weight (вес)	bitro diameter (охват бедер)	chest girth (объем груди)	chest width (охват грудной клетки)
height (рост)	1.00	0.85	0.80	0.86	0.47	0.40	0.30	0.38
arm span (размах рук)	0.85	1.00	0.88	0.83	0.38	0.33	0.28	0.41

Окончание табл. 14.3

	height (рост)	arm span (размах рук)	forearm (предплечье)	lower leg (голень)	weight (вес)	bitro diameter (охват бедер)	chest girth (объем груди)	chest width (охват грудной клетки)
forearm (предплечье)	0.80	0.88	1.00	0.80	0.38	0.32	0.24	0.34
lower leg (голень)	0.86	0.83	0.80	1.00	0.44	0.33	0.33	0.36
weight (вес)	0.47	0.38	0.38	0.44	1.00	0.76	0.73	0.63
bitro diameter (охват бедер)	0.40	0.33	0.32	0.33	0.76	1.00	0.58	0.58
chest girth (объем груди)	0.30	0.28	0.24	0.33	0.73	0.58	1.00	0.54
chest width (охват грудной клетки)	0.38	0.41	0.34	0.36	0.63	0.58	0.54	1.00

Источник: Harman, H. H. (1976) Modern Factor Analysis, Third Edition Revised, University of Chicago Press, Table 2.3.

Нам вновь нужно заменить исходные переменные меньшим числом главных компонент. Определить число компонент, которые нужно извлечь, можно с помощью следующего программного кода. В данном случае нам нужно указать, что в качестве исходных данных используется корреляционная матрица (компонент `cov` объекта `Harman23.cor`), и указать объем выборки (`n.obs`):

```
library(psych)
fa.parallel(Harman23.cor$cov, n.obs=302, fa="pc", n.iter=100,
            show.legend=FALSE, main="Scree plot with parallel analysis")
abline(h=1)
```

Полученный график показан на рис. 14.3.

Судя по диаграмме, достаточно выделить две компоненты. Как и в первом примере, критерий Кайзера–Харриса, критерий собственных значений и параллельный анализ дают сходные результаты. Так бывает не всегда, и в подобных случаях может понадобиться извлечь разное число компонент и выбрать то, которое кажется наиболее адекватным. Код в листинге 14.2 извлекает две первые компоненты из корреляционной матрицы.

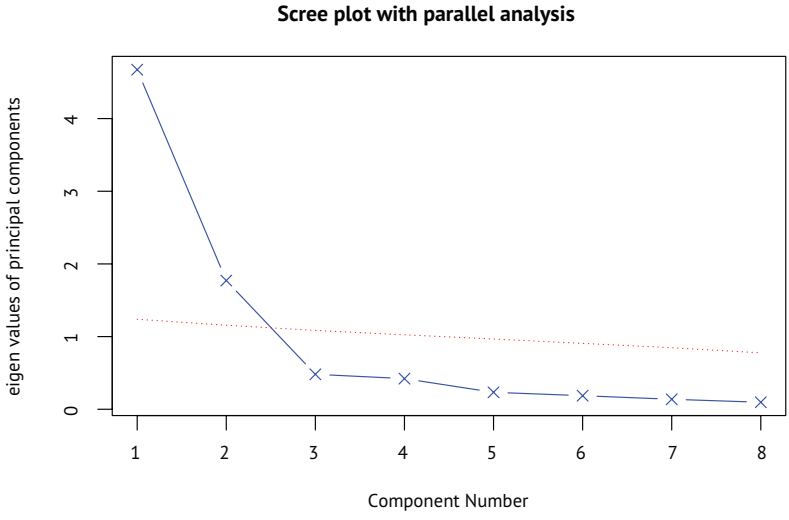


Рис. 14.3. Определение числа главных компонент, достаточного для характеристики данных из примера с параметрами тела. График собственных значений (линия с крестиками), критерия превышения единицы собственными значениями (горизонтальная линия) и параллельный анализ со 100 симуляциями (пунктир) свидетельствуют о том, что достаточно использовать две компоненты

Листинг 14.2. Применение метода главных компонент к результатам измерений параметров тела

```
> library(psych)
> pc <- principal(Harman23.cor$cov, nfactors=2, rotate="none")
> pc

Principal Components Analysis
Call: principal(r = Harman23.cor$cov, nfactors = 2, rotate = "none")
Standardized loadings based upon correlation matrix
                PC1   PC2   h2    u2
height         0.86 -0.37 0.88 0.123
arm.span       0.84 -0.44 0.90 0.097
forearm        0.81 -0.46 0.87 0.128
lower.leg      0.84 -0.40 0.86 0.139
weight         0.76  0.52 0.85 0.150
bitro.diameter 0.67  0.53 0.74 0.261
chest.girth    0.62  0.58 0.72 0.283
chest.width    0.67  0.42 0.62 0.375

                PC1  PC2
SS loadings    4.67 1.77
Proportion Var 0.58 0.22
Cumulative Var 0.58 0.81

[... additional output omitted ...]
```

Если внимательно рассмотреть столбцы `PC1` и `PC2` в листинге 14.2, то можно заметить, что первая компонента учитывает 58 % дисперсии переменных, представляющих разные параметры тела, а вторая – 22 %. Вместе эти две компоненты объясняют 81 % общей дисперсии и 88 % дисперсии значений роста.

Компоненты и факторы интерпретируются по значениям их нагрузок. Первая компонента положительно коррелирует со всеми физическими параметрами и может интерпретироваться как общий показатель размеров тела. Вторая компонента противопоставляет первые четыре переменные (`height`, `arm.span`, `forearm` и `lower.leg`) и вторые четыре (`weight`, `bitro.diameter`, `chest.girth` и `chest.width`) и, похоже, соответствует отношению длины к объему. С такой компонентой непросто работать. После извлечения двух или более компонент их можно повернуть, чтобы упростить интерпретацию. Этот вопрос мы обсудим далее.

14.2.3. Вращение главных компонент

Вращение – это набор математических приемов трансформации матрицы нагрузок компонент в другую, более легко интерпретируемую. Это делается путем максимальной «очистки» компонент. Виды вращений различаются по сохранению ортогональности получившихся в результате компонент (*ортогональное вращение*), или же допускается их корреляция (*косоугольное вращение*). Они также различаются по способу «очистки». Наиболее распространенный тип ортогонального вращения – *варимакс* (varimax), когда делается попытка очистить столбцы матрицы нагрузок так, чтобы каждая компонента объясняла дисперсию ограниченного набора переменных (то есть в каждом столбце будет лишь несколько больших нагрузок и много маленьких). В листинге 14.3 показан пример применения вращения варимакс к данным с параметрами тела. Пример косоугольного вращения будет показан в разделе 14.4.

Листинг 14.3. Применение метода главных компонент с вращением варимакс

```
> rc <- principal(Harman23.cor$cov, nfactors=2, rotate="varimax")
> rc

Principal Components Analysis
Call: principal(r = Harman23.cor$cov, nfactors = 2, rotate = "varimax")
Standardized loadings based upon correlation matrix
                RC1  RC2   h2    u2
height         0.90 0.25 0.88 0.123
arm.span       0.93 0.19 0.90 0.097
forearm        0.92 0.16 0.87 0.128
lower.leg      0.90 0.22 0.86 0.139
weight         0.26 0.88 0.85 0.150
bitro.diameter 0.19 0.84 0.74 0.261
```

```
chest.girth    0.11 0.84 0.72 0.283
chest.width    0.26 0.75 0.62 0.375

                RC1  RC2
SS loadings    3.52 2.92
Proportion Var 0.44 0.37
Cumulative Var 0.44 0.81

[... additional output omitted ...]
```

Названия столбцов изменились с `PC` на `RC` (rotated components – компоненты после поворота), чтобы обозначить вращение. Рассматривая нагрузки в столбце `RC1`, можно заметить, что первая компонента в основном объясняет первые четыре переменные (характеризующие длины). Значения нагрузки в столбце `RC2` указывают, что вторая компонента в основном объясняет переменные с 5 по 8 (характеризующие объемы). Обратите внимание, что эти две компоненты по-прежнему ортогональны, и вместе они так же хорошо объясняют дисперсию переменных, как и до вращения. Это происходит потому, что общности переменных не изменились. Общая доля объясняемой дисперсии также осталась прежней (81 %). Однако доля дисперсии, объясняемая каждой переменной, изменилась (с 58 % до 44 % для первой компоненты и с 22 % до 37 % для второй компоненты). Такое уравнивание долей дисперсии, объясняемых разными компонентами, – обычное дело, и формально их теперь следует называть компонентами, а не главными компонентами (потому что изменилась доля объясняемой дисперсии, характеризующая отдельные компоненты).

Наша конечная цель – заменить больший набор коррелированных переменных меньшим набором производных переменных. Чтобы это сделать, нужно рассчитать оценки компонент для каждого наблюдения.

14.2.4. Вычисление оценок главных компонент

В примере с рейтингом судей мы извлекли единственную главную компоненту из исходных данных, описывающих рейтинг судей по 11 признакам. Значения производной переменной для каждого участника легко получить при помощи функции `principal()` (листинг 14.4).

Листинг 14.4. Вычисление оценок компонент для исходных данных

```
> library(psych)
> pc <- principal(USJudgeRatings[,-1], nfactors=1, score=TRUE)
> head(pc$scores)
                      PC1
AARONSON,L.H. -0.1857981
ALEXANDER,J.M. 0.7469865
ARMENTANO,A.J. 0.0704772
```

```
BERDON,R.I.    1.1358765
BRACKEN,J.J. -2.1586211
BURNS,E.B.     0.7669406
```

Оценки главных компонент сохраняются как элемент `scores` в объекте, который создается функцией `principal()` при вызове с параметром `scores=TRUE`. При желании теперь можно вычислить корреляцию между числом контактов адвоката и судьи и мнением адвокатов о судье:

```
> cor(USJudgeRatings$CONT, pc$score)
              PC1
[1,] -0.008815895
```

Конечно же, не существует никакой связи между степенью знакомства адвоката с судьей и мнением адвоката[1]!

Ясно, что невозможно вычислить оценки главных компонент для каждого наблюдения, если анализ главных компонент основан на корреляционной матрице и исходные данные недоступны. Однако можно вычислить коэффициенты, которые используются для получения значений главных компонент.

Данные о параметрах тела представляли собой корреляции между этими параметрами, но у нас не было отдельных значений параметров для каждой из 305 девочек. Коэффициенты можно получить, как показано в листинге 14.5.

Листинг 14.5. Вычисление коэффициентов для главных компонент

```
> library(psych)
> rc <- principal(Harman23.cor$cov, nfactors=2, rotate="varimax")
> round(unclass(rc$weights), 2)
                 RC1   RC2
height          0.28 -0.05
arm.span        0.30 -0.08
forearm         0.30 -0.09
lower.leg       0.28 -0.06
weight         -0.06  0.33
bitro.diameter -0.08  0.32
chest.girth    -0.10  0.34
chest.width    -0.04  0.27
```

Оценки компонент вычисляются по формулам:

```
PC1 = 0.28*height + 0.30*arm.span + 0.30*forearm + 0.29*lower.leg -
      0.06*weight - 0.08*bitro.diameter - 0.10*chest.girth -
      0.04*chest.width
```

и

[1] Исследуемые объекты (в данном случае это судьи) классифицируются по значениям главных компонент. Такую классификацию изображают в виде диаграммы рассеяния. Подробнее об этом можно прочесть в книге А. Б. Шипунова с соавторами «Наглядная статистика. Используем R!» (М.: ДМК Пресс, 2012). – *Прим. перев.*

```
PC2 = -0.05*height - 0.08*arm.span - 0.09*forearm - 0.06*lower.leg +
       0.33*weight + 0.32*bitro.diameter + 0.34*chest.girth +
       0.27*chest.width
```

Подразумевается, что измеренные параметры тела были стандартизированы (среднее = 0, стандартное отклонение = 1). Обратите внимание, что коэффициенты для `PC1` примерно равны 0,3 или 0. То же верно и для `PC2`. С практической точки зрения, можно упростить дальнейший анализ, приняв оценки для первой компоненты равными среднему арифметическому стандартизированных значений четырех первых переменных. Сходным образом вторую компоненту можно определить как среднее стандартизированных значений вторых четырех переменных. Это то, что я обычно делаю на практике.

«Маленькое мгновение» завоевывает мир

Люди, анализирующие данные, частенько путают МГК и ФА. Одна из причин – историческая, она берет начало с программы, которая называлась «Little Jiffy»[1] (я не шучу). Эта одна из старых программ, широко использовавшаяся для проведения факторного анализа, но по умолчанию она применяла метод главных компонент, извлекая компоненты, собственные значения которых превышали единицу, и проводя вращение варимакс. Данная программа использовалась так широко, что многие социологи стали отождествлять ее с разведочным факторным анализом (ФА). Многие более поздние статистические программы тоже реализовали аналогичное поведение для проведения ФА[2].

Как, я надеюсь, вы поймете из следующего раздела, между МГК и ФА есть важные фундаментальные различия. Вы можете узнать больше о путанице между МГК и ФА, прочитав публикацию Hayton, Allen & Scarpello (2004).

Если цель – поиск скрытых переменных, которые объясняют наблюдаемые, тогда следует обратиться к факторному анализу. Это станет темой следующего раздела.

---

[1] «Маленькое мгновение». – *Прим. перев.*

[2] Автор заблуждается. МГК как метод выделения факторов прямо реализован в стандартной процедуре `fa()`. Метод выделения осей тут совсем ни при чем. Суть различий между МГК и ФА заключается в проведении дополнительного вращения осей после отбрасывания выделенных факторов, которые не входят в «простую структуру». Процедура дополнительного вращения системы координат, да и само понятие «простой структуры» просто не имеют смысла с точки зрения математической статистики. Ортогональные вращения дают тождественные данные, а косоугольные – фактически разрушают данные. Следует также заметить, что обоснование процедуры МГК методами математической статистики вызвало другое часто встречающееся заблуждение. Поскольку обоснование процедуры построено вокруг трансформаций многомерного нормального распределения, то и сам метод МГК ошибочно ограничивают применением только к нормально распределенным данным. На самом деле МГК не налагает никаких ограничений на вид обрабатываемых данных (в этом легко убедиться, прочитав оригинальную работу: в исходной формулировке К. Пирсона ставится задача об аппроксимации конечного множества данных и отсутствует даже гипотеза об их статистической природе, не говоря уже о распределении). – *Прим. перев.*

14.3. Разведочный факторный анализ

Цель ФА – объяснить корреляции внутри набора наблюдаемых переменных меньшим набором более фундаментальных ненаблюдаемых переменных, которые лежат в основе данных. Эти гипотетические ненаблюдаемые переменные называют *факторами*. Каждый фактор должен объяснять суммарную дисперсию двух или более наблюдаемых переменных, так что формально они называются *общими факторами*.

Модель выглядит так:

$$X_i = a_1F_1 + a_2F_2 + \ldots + a_pF_p + U_i,$$

где X_i – это i-я наблюдаемая переменная ($i = 1 \ldots k$), F_j – это общие факторы ($j = 1 \ldots p$), и $p < k$. U_i – это уникальная составляющая переменной X_i (не объясняемая общими факторами). Параметр a_i можно интерпретировать как степень вклада каждого фактора в наблюдаемую переменную. Если вернуться к примеру с набором данных `Harman74.cor`, то можно сказать, что результаты отдельных испытуемых по 24 психологическим тестам представляют собой взвешенные комбинации способностей испытуемых, оцененных по четырем психологическим параметрам, лежащим в основе этих тестов.

Хотя модели МГК и ФА различаются, в процессе анализа выполняются практически одни и те же шаги. Мы проиллюстрируем процесс, применив ФА к корреляциям между результатами шести психологических тестов для 112 человек. Тесты были следующими: невербальная оценка общего умственного развития (general), тест на завершение фигур (picture), тест блочных конструкций (blocks), тест с лабиринтом (maze), тест на понимание прочитанного (reading) и тест на словарный запас (vocab). Можно ли объяснить результаты, показанные участниками в этих тестах, при помощи небольшого числа основополагающих, или скрытых, психологических параметров?

Ковариационная матрица для наблюдаемых переменных содержится в наборе данных `ability.cov`. Ее можно преобразовать в корреляционную матрицу с помощью функции `cov2cor()`. Пропущенные данные в этом случае отсутствуют.

```
> options(digits=2)
> covariances <- ability.cov$cov
> correlations <- cov2cor(covariances)
> correlations
         general picture blocks maze reading vocab
general     1.00    0.47   0.55 0.34    0.58  0.51
picture     0.47    1.00   0.57 0.19    0.26  0.24
blocks      0.55    0.57   1.00 0.45    0.35  0.36
maze        0.34    0.19   0.45 1.00    0.18  0.22
reading     0.58    0.26   0.35 0.18    1.00  0.79
vocab       0.51    0.24   0.36 0.22    0.79  1.00
```

Поскольку мы ищем гипотетические переменные, объясняющие данные, используем ФА. Как и в случае с МГК, перед нами стоит задача определить, сколько факторов извлечь.

14.3.1. Определение числа извлекаемых факторов

Чтобы выяснить, сколько факторов извлечь, воспользуемся функцией `fa.parallel()`:

```
> library(psych)
> covariances <- ability.cov$cov
> correlations <- cov2cor(covariances)
> fa.parallel(correlations, n.obs=112, fa="both", n.iter=100,
              main="Scree plots with parallel analysis")
> abline(h=c(0, 1))
```

Полученная диаграмма показана на рис. 14.4. Обратите внимание, что функция способна выводить результаты и для метода главных компонент, и для факторного анализа, так что вы можете сравнить эти два подхода (`fa="both"`).

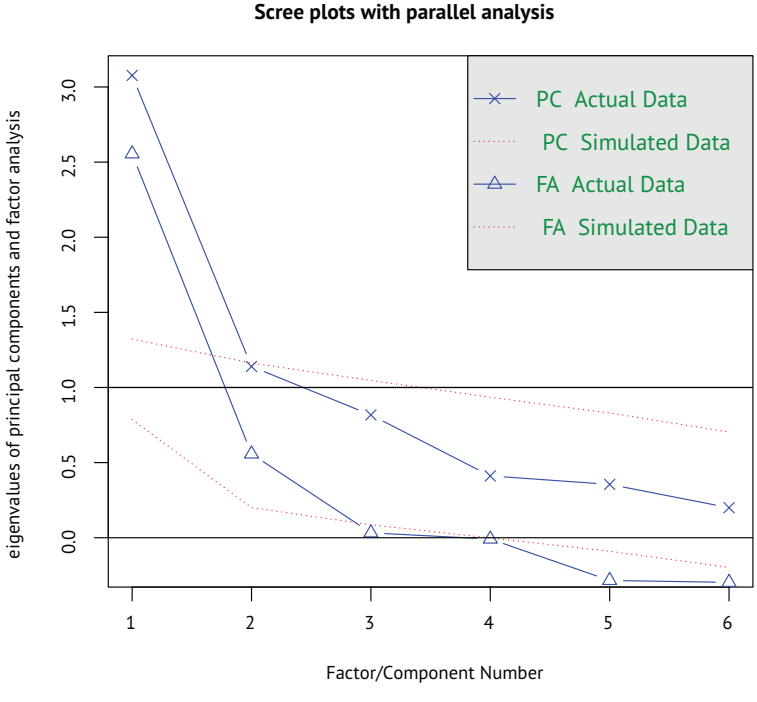


Рис. 14.4. Определение числа извлекаемых факторов для примера с психологическими тестами. Показаны результаты как для МГК (предлагается выделить две компоненты), так и для ФА (следует выделить два фактора)

Отметим несколько особенностей на этом графике. Если бы применялся МГК, нам пришлось бы выбирать между одной (критерий собственных значений, параллельный анализ) или двумя

компонентами (собственные значения больше единицы). В случае сомнения всегда лучше выделить больше факторов. Выбор большего числа факторов, чем необходимо, приводит к меньшему искажению «правильных» результатов анализа.

В случае ФА ясно видно, что нужно выделить два фактора. Первые два собственных значения (треугольники) находятся выше изгиба кривой собственных значений, а также превышают среднее собственных значений, полученных на основании 100 симулированных матриц данных. Для ФА собственные значения, согласно критерию Кайзера–Харриса, должны превышать 0, а не 1 (многие этого не знают, так что это хороший способ выигрывать пари на вечеринках). В данном случае критерий Кайзера–Харриса тоже свидетельствует о необходимости выделения двух факторов.

14.3.2. Выделение общих факторов

Теперь, решив выделить два фактора, можно воспользоваться функцией `fa()`. Она имеет следующий синтаксис:

```
fa(r, nfactors=, n.obs=, rotate=, scores=, fm=)
```

где

- `r` – корреляционная матрица или матрица с исходными данными;
- `nfactors` – число выделяемых факторов (1 по умолчанию);
- `n.obs` – число наблюдений (если анализируется корреляционная матрица);
- `rotate` – тип вращения факторов (по умолчанию облимин, `oblimin`);
- `scores` – признак необходимости вычислять оценки факторов (по умолчанию – нет);
- `fm` – метод факторного анализа (по умолчанию минрез, `minres`).

В отличие от МГК, в ФА существует много способов извлечения общих факторов: максимальное правдоподобие (`ml`), повторные главные оси (iterated principal axis, `pa`), взвешенный наименьший квадрат (`wls`), обобщенные взвешенные наименьшие квадраты (`gls`) и наименьшие остатки (`minres`). Статистики в основном предпочитают способ максимального правдоподобия, поскольку в его основе лежит проверенная статистическая модель. Иногда этот метод не сходится, и в таком случае часто хорошо работает метод повторных главных осей. Более подробное описание всех этих подходов дано в работах Mulaik (2009) и Gorsuch (1983).

В приведенном примере мы выделим факторы, не вращая их, используя метод повторных главных осей. Результаты показаны в листинге 14.6.

Листинг 14.6. Выделение факторов методом повторных главных осей без вращения

```
> fa <- fa(correlations, nfactors=2, rotate="none", fm="pa")
> fa
Factor Analysis using method =  pa
Call: fa(r = correlations, nfactors = 2, rotate = "none", fm = "pa")
Standardized loadings based upon correlation matrix
         PA1   PA2   h2   u2
general 0.75  0.07 0.57 0.43
picture 0.52  0.32 0.38 0.62
blocks  0.75  0.52 0.83 0.17
maze    0.39  0.22 0.20 0.80
reading 0.81 -0.51 0.91 0.09
vocab   0.73 -0.39 0.69 0.31
                PA1  PA2
SS loadings    2.75 0.83
Proportion Var 0.46 0.14
Cumulative Var 0.46 0.60
[... additional output deleted ...]
```

Как видите, два фактора объясняют 60 % дисперсии результатов шести психологических тестов. Однако, как следует из факторных нагрузок, полученные факторы нелегко интерпретировать. Вращение факторов должно помочь.

14.3.3. Вращение факторов

Повернуть два фактора, полученных в разделе 14.3.3, можно путем ортогонального или косоугольного вращения. Давайте попробуем оба способа и посмотрим, чем они отличаются. Сначала выполним ортогональное вращение (листинг 14.7).

Листинг 14.7. Выделение факторов с ортогональным вращением

```
> fa.varimax <- fa(correlations, nfactors=2, rotate="varimax", fm="pa")
> fa.varimax
Factor Analysis using method =  pa
Call: fa(r = correlations, nfactors = 2, rotate = "varimax", fm = "pa")
Standardized loadings based upon correlation matrix
         PA1  PA2   h2   u2
general 0.49 0.57 0.57 0.43
picture 0.16 0.59 0.38 0.62
blocks  0.18 0.89 0.83 0.17
maze    0.13 0.43 0.20 0.80
reading 0.93 0.20 0.91 0.09
vocab   0.80 0.23 0.69 0.31

                PA1  PA2
SS loadings    1.83 1.75
Proportion Var 0.30 0.29
Cumulative Var 0.30 0.60

[... additional output omitted ...]
```

Глядя на факторные нагрузки, можно сказать, что после поворота факторы легче интерпретировать. Понимание прочтенного и словарный запас соответствуют первому фактору, а фигуры, лабиринт и блоки – второму. Общий показатель невербального интеллектуального развития связан с обоими факторами. На основании этого можно решить, что у нас есть фактор вербального интеллекта и фактор невербального интеллекта.

Ортогональное вращение не влияет на ортогональность самих факторов. А что, если допустить возможность корреляции факторов? Попробуем применить косоугольное вращение, такое как промакс (`promax`; листинг 14.8).

Листинг 14.8. Выделение факторов с косоугольным вращением

```
> fa.promax <- fa(correlations, nfactors=2, rotate="promax", fm="pa")
> fa.promax
Factor Analysis using method =  pa
Call: fa(r = correlations, nfactors = 2, rotate = "promax", fm = "pa")
Standardized loadings based upon correlation matrix
          PA1   PA2   h2   u2
general  0.36  0.49 0.57 0.43
picture -0.04  0.64 0.38 0.62
blocks  -0.12  0.98 0.83 0.17
maze    -0.01  0.45 0.20 0.80
reading  1.01 -0.11 0.91 0.09
vocab    0.84 -0.02 0.69 0.31

                PA1  PA2
SS loadings    1.82 1.76
Proportion Var 0.30 0.29
Cumulative Var 0.30 0.60

 With factor correlations of
     PA1  PA2
PA1 1.00 0.57
PA2 0.57 1.00
[... additional output omitted ...]
```

Этот результат отличается от полученного ортогональным вращением. При ортогональном вращении внимание фокусируется на *матрице факторной структуры* (factor structure matrix, корреляциях переменных с факторами). При косоугольном вращении рассматриваются три матрицы: факторной структуры, модели факторов и корреляций между факторами (factor intercorrelation matrix).

Матрица модели факторов (factor pattern matrix) – это матрица стандартизированных регрессионных коэффициентов. Они определяют веса (коэффициенты) для предсказания значений переменных по значениям факторов.

В листинге 14.8 значения в столбцах `PA1` и `PA2` формируют матрицу модели факторов. Это скорее стандартизированные регресси-

онные коэффициенты, чем коэффициенты корреляции. Анализ этих столбцов также используется для интерпретации факторов (хотя не все сходятся во мнениях по этому поводу). Здесь мы вновь можем выделить вербальный и невербальный факторы.

Корреляция между этими двумя факторами составляет 0,57. Это высокая корреляция. Если бы корреляция была низкой, нам нужно было бы вернуться к ортогональному вращению, чтобы не усложнять результаты анализа.

Матрица факторной структуры (или факторных нагрузок) не вычисляется. Однако ее легко получить по формуле $F = P \times Phi$, где F – матрица факторных нагрузок, P – матрица модели факторов, а Phi – матрица корреляций между факторами. Вычисления по этой формуле можно произвести при помощи следующей простой функции:

```
fsm <- function(oblique) {
    if (class(oblique)[2]=="fa" & is.null(oblique$Phi)) {
        warning("Object doesn't look like oblique EFA")
    } else {
        P <- unclass(oblique$loading)
        F <- P %*% oblique$Phi
        colnames(F) <- c("PA1", "PA2")
        return(F)
    }
}
```

Применив ее к нашему примеру, получим:

```
> fsm(fa.promax)
         PA1  PA2
general 0.64 0.69
picture 0.33 0.61
blocks  0.44 0.91
maze    0.25 0.45
reading 0.95 0.47
vocab   0.83 0.46
```

Теперь мы можем проанализировать корреляции между переменными и факторами. Сравнив их с матрицей факторных нагрузок после ортогонального вращения, можно заметить, что столбцы не такие «чистые». Это объясняется допущенной возможностью корреляции факторов друг с другом. Хотя результаты косоугольного вращения сложнее для интерпретации, они часто представляют более реалистичную модель данных.

При желании можно графически изобразить результаты ортогонального или косоугольного вращения при помощи функции `factor.plot()` или `fa.diagram()`. Инструкция

```
factor.plot(fa.promax, labels=rownames(fa.promax$loadings))
```

создаст диаграмму, показанную на рис. 14.5.

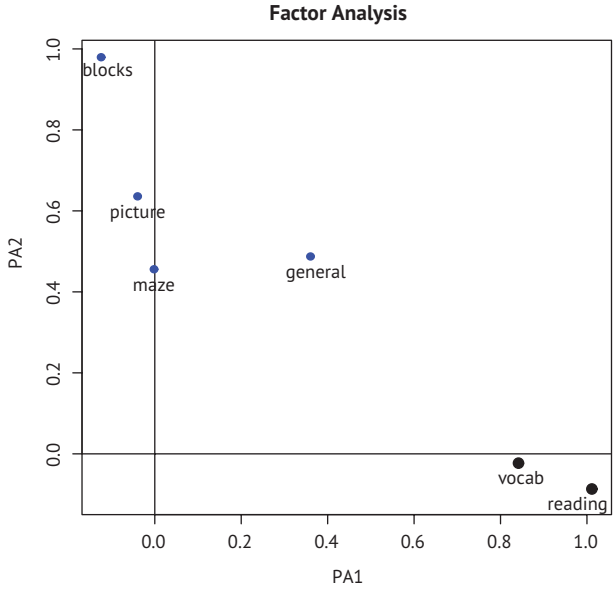


Рис. 14.5. Двухфакторная диаграмма с результатами психологических тестов из набора данных ability.cov. Словарный запас (vocab) и понимание прочтенного (reading) ассоциированы с первым фактором (PA1), а блоки, картинки и лабиринт – со вторым (PA2). Результаты теста на общее умственное развитие связаны с обоими факторами

Инструкция

```
fa.diagram(fa.promax, simple=FALSE)
```

создаст диаграмму, показанную на рис. 14.6. Если использовать параметр `simple=TRUE`, то для каждого объекта будет показана только наибольшая нагрузка. На этой диаграмме изображены нагрузки для всех факторов, а также корреляции между факторами. Такая диаграмма полезна, когда число факторов невелико.

В реальной жизни использование факторного анализа для такого небольшого числа переменных маловероятно. Я не стал использовать более объемный набор данных только ради экономии места. Если вы хотите проверить свои умения, попробуйте применить факторный анализ к результатам 24 психологических тестов, содержащимся в наборе данных `Harman74.cor`. Следующие инструкции помогут вам начать:

```
library(psych)
fa.24tests <- fa(Harman74.cor$cov, nfactors=4, rotate="promax")
```

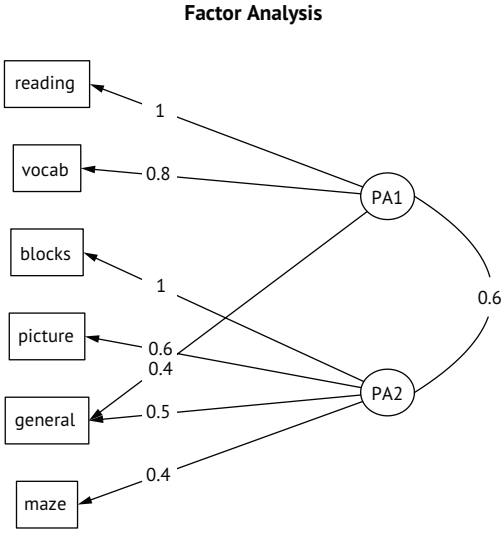


Рис. 14.6. Диаграмма, отражающая эффект косоугольного вращения двух факторов для результатов психологических тестов из набора данных ability.cov

14.3.4. Оценки факторов

В отличие от МГК, в ФА гораздо реже вычисляются оценки факторов. Однако эти значения можно легко получить при помощи функции `fa()`, добавив параметр `score=TRUE` (если доступны исходные данные). Кроме того, коэффициенты для вычисления оценок факторов (стандартизированные регрессионные коэффициенты) содержатся в элементе `weights` объекта, который возвращает функция `fa()`.

Для набора данных `ability.cov` можно получить бета-коэффициенты для вычисления оценок двух выделенных факторов после косоугольного вращения, как показано ниже:

```
> fa.promax$weights
         [,1]  [,2]
general 0.080 0.210
picture 0.021 0.090
blocks  0.044 0.695
maze    0.027 0.035
reading 0.739 0.044
vocab   0.176 0.039
```

В отличие от компонент, оценки которых вычисляются точно, для факторов можно получить лишь приближенные оценки. Сделать это можно несколькими способами. Функция `fa()` использует регрессионный подход. Более подробная информация об оценке значений факторов приводится в публикации DiStefano, Zhu & Mondrila (2009).

Прежде чем двинуться дальше, давайте кратко рассмотрим другие пакеты для разведочного факторного анализа, имеющиеся в R.

14.3.5. Другие пакеты для проведения факторного анализа

В R есть еще несколько пакетов, поддерживающих факторный анализ. В пакете `FactoMineR` реализованы МГК и ФА, а также иные модели скрытых переменных. Этот пакет предоставляет много возможностей, которые не обсуждались в данной главе, включая совместное использование числовых и категориальных переменных. Пакет `FAiR` позволяет оценивать факторные модели при помощи генетического алгоритма, который позволяет накладывать ограничения на пространство возможных значений параметров модели. В пакете `GPArotation` реализовано много дополнительных методов вращения факторов. Наконец, пакет `nFactors` дает возможность применять сложные техники для определения числа факторов, лежащих в основе данных.

14.4. Другие модели скрытых переменных

ФА – это одна из многих моделей скрытых переменных, которые используются в статистике. Мы закончим эту главу кратким описанием других моделей, которые можно подбирать в R. К ним относятся модели проверки априорных теорий, модели, способные работать со смешанными типами данных (числовые и категориальные), и модели, основанные лишь на многомерных таблицах сопряженности категориальных переменных.

Определение числа и значений факторов в ФА основано на данных. Однако, начиная анализ, можно руководствоваться теоретическими представлениями о числе факторов, лежащих в основе переменных, и о том, как эти факторы связаны друг с другом. Затем можно проверить эту теорию при помощи собранных данных. Этот подход называется *конфирматорным факторным анализом* (confirmatory factor analysis, CFA).

CFA – это частный случай методического подхода, называемого *моделированием структурных уравнений* (structural equation modeling, SEM). SEM позволяет не только задать число и состав факторов, лежащих в основе данных, но и указать, как эти факторы влияют друг на друга. SEM можно определить как сочетание конфирматорного факторного анализа (для количественных переменных) и регрессионного анализа (для факторов). В числе результатов SEM выводятся статистические критерии и индексы соответствия моделей данным. В R есть несколько замечательных пакетов (включая `sem`, `openMx` и `lavaan`), при помощи которых можно выполнить моделирование структурных уравнений и конфирматорный факторный анализ.

Пакет `ltm` можно использовать для подгонки моделей скрытых переменных по результатам тестов и опросников. Эта методология обычно используется для создания крупномасштабных стандартизированных тестов, таких как американские отборочные тесты при по-

ступлении в колледж или в университет (Scholastic Aptitude Test, SAT) и тест при поступлении в магистратуру (Graduate Record Exam, GRE).

Модели скрытых классов (latent class models), в которых предполагается, что лежащие в основе данных факторы будут категориальными, а не непрерывными, можно подобрать при помощи пакетов `FlexMix`, `lcmm`, `randomLCA` и `poLC`. Пакет `lcda` позволяет провести дискриминантный анализ для скрытых классов, а пакет `lsa` – анализ скрытых семантик – методологии, используемой при обработке естественных языков.

В пакете `ca` реализованы функции для простого и множественного анализа соответствий. Этот метод позволяет исследовать структуру категориальных переменных в двух- и многомерных таблицах соответственно.

Наконец, в R реализовано много методов многомерного масштабирования (multidimensional scaling, MDS)[1]. MDS-анализ позволяет выявлять измерения, лежащие в основе данных и объясняющие сходства и различия между объектами исследования (например, странами). Функция `cmdscale()` из стандартного дистрибутива R позволяет провести классический MDS-анализ, функция `isoMDS()` из пакета `MASS` выполняет неметрический MDS-анализ. Пакет `vegan` также содержит функции для выполнения классического и неметрического MDS-анализа.

Итоги

- Метод главных компонент (МГК) – полезный метод сокращения данных, позволяющий заменить множество коррелированных переменных меньшим количеством некоррелированных составных переменных.
- Разведочный факторный анализ (ФА) предлагает широкий спектр методов выявления скрытых или ненаблюдаемых конструкций (факторов), которые могут лежать в основе набора наблюдаемых, или явных переменных.
- В отличие от МГК, цель которого состоит в том, чтобы обобщить данные и уменьшить их размерность, ФА можно использовать как инструмент для генерации гипотез с целью понять отношения между переменными. Они часто используются в социальных науках для разработки различных теорий.
- МГК и ФА – это многоэтапные процессы, требующие от аналитика делать выбор на каждом этапе. Эти шаги показаны на рис. 14.7.

[1] Подробнее о многомерном масштабировании, анализе соответствий и других распространенных методах многомерного анализа, которые не упомянуты здесь (например, кластерном и дискриминантном), можно прочесть в книге А. Б. Шипунова с соавторами «Наглядная статистика. Используем R!» (М.: ДМК Пресс, 2012). – *Прим. перев.*

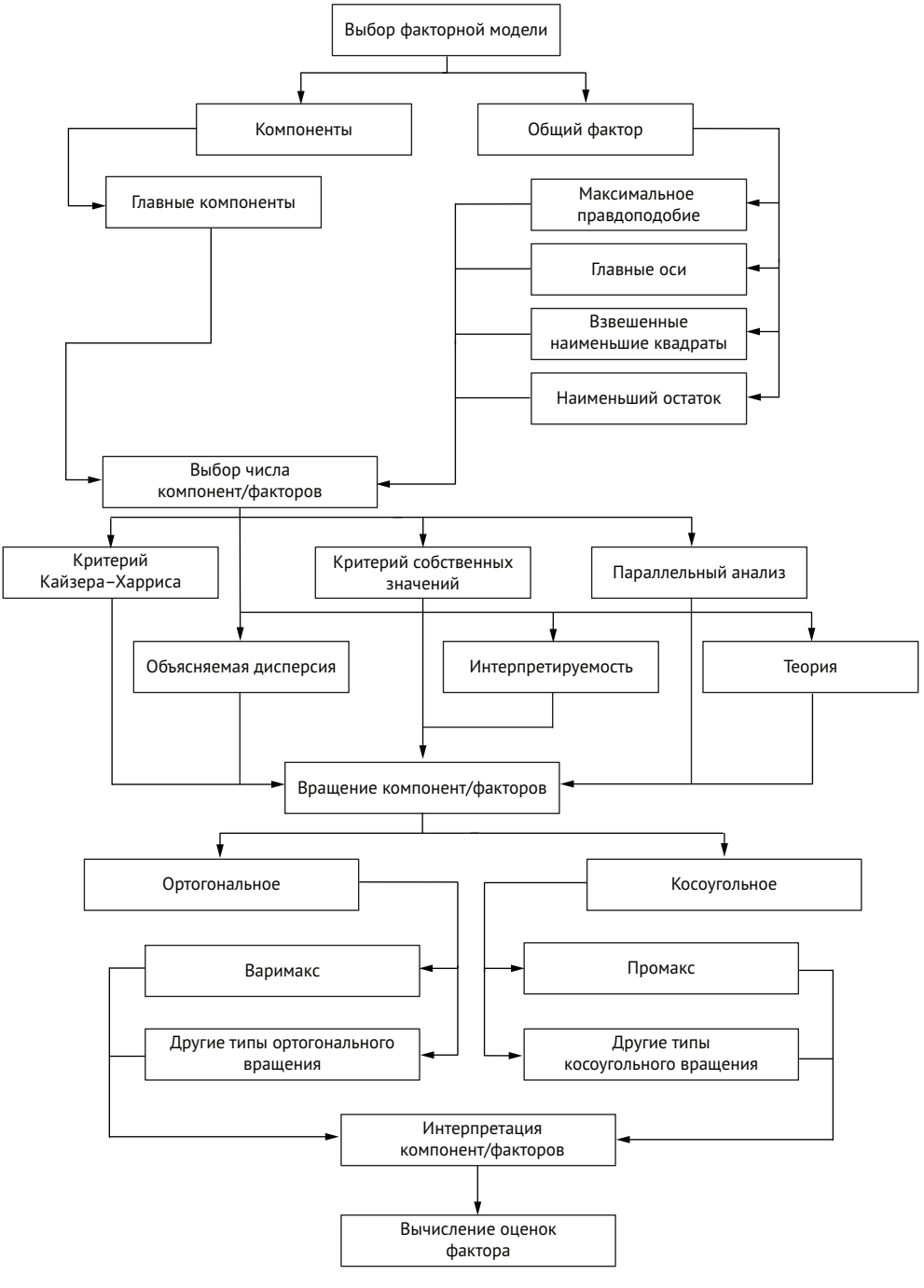


Рис. 14.7. Основные этапы метода главных компонент и разведочного факторного анализа

15 Временные ряды

В этой главе:

- создание временных рядов;
- разложение временных рядов на компоненты.
- разработка прогнозных моделей;
- предсказание будущих значений.

Насколько быстро развивается глобальное потепление и каковы будут его последствия через 10 лет? За исключением дисперсионного анализа повторных измерений, представленного в разделе 9.6, во всех предыдущих главах основное внимание уделялось *перекрестному анализу* данных. В перекрестном анализе участвуют данные, полученные в один момент времени. *Профильный анализ*, напротив, сосредоточен на исследовании данных, включающих многократные измерения переменных с течением времени. Следя за развитием явления во времени, можно многое узнать о нем.

В этой главе мы займемся исследованием наблюдений, записанных через равные промежутки в течение заданного интервала времени. Подобные наблюдения можно упорядочить в виде временного ряда Y_1, Y_2, Y_3, ..., Y_t , ..., Y_T, где Y_t – это значение Y в момент времени t, а T – общее количество наблюдений в серии.

Рассмотрим два очень разных временных ряда, показанных на рис. 15.1. График слева отражает изменение квартальной прибыли

(в долларах) на акцию Johnson & Johnson в период с 1960 по 1980 год. Всего наблюдений 84: по 1 на каждый квартал за 21 год. График справа отражает среднемесячное относительное количество пятен на Солнце с 1749 по 1983 год, зарегистрированное Швейцарской федеральной обсерваторией и Токийской астрономической обсерваторией. Временные ряды солнечных пятен намного длиннее: 2820 наблюдений – по 1 на каждый месяц в течение 235 лет.

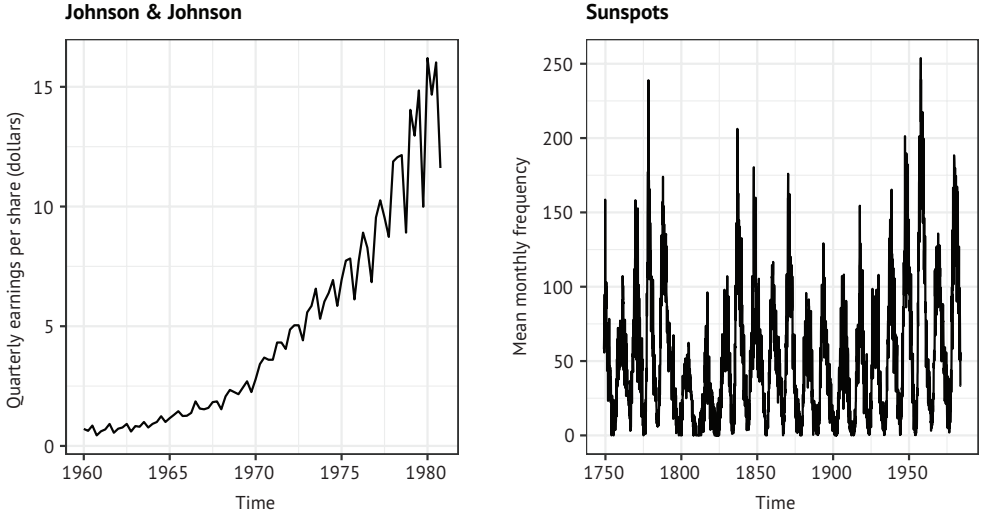


Рис. 15.1. Диаграммы временных рядов, отражающие (слева) квартальную прибыль на акцию Johnson & Johnson (в долларах) с 1960 по 1980 год и (справа) среднемесячное относительное количество пятен на Солнце, зарегистрированное с 1749 по 1983 год

Изучение временных рядов включает поиск ответов на два фундаментальных вопроса: что произошло (описание) и что будет дальше (прогнозирование)? В отношении акций Johnson & Johnson можно спросить:

- Меняется ли цена акций Johnson & Johnson со временем?
- Имеется ли какая-то квартальная цикличность в изменениях цен на акции в течение года?
- Можно ли предсказать цены на акции в будущем, и если да, то насколько точным будет такой прогноз?

В отношении пятен на Солнце можно спросить:

- Какие статистические модели лучше всего описывают изменение количества солнечных пятен?
- Есть ли модели, которые лучше аппроксимируют данные, чем другие?
- Можно ли предсказать количество солнечных пятен в данный момент времени, и если да, то насколько точно?

Возможность точно предсказывать цены на акции имеет отношение к моему (надеюсь) досрочному выходу на пенсию и отдыху на тропическом острове, а возможность предсказывать количество солнечных пятен имеет отношение к надежности работы моего мобильного телефона на этом острове.

Прогнозирование будущих значений временных рядов – это фундаментальная человеческая деятельность, и исследования временных рядов имеют важные практические приложения. Экономисты используют временные ряды для понимания и прогнозирования изменений на финансовых рынках. Городские власти используют временные ряды для прогнозирования будущих потребностей в транспорте. Ученые-климатологи используют временные ряды для изучения глобального изменения климата. Корпорации используют временные ряды для прогнозирования спроса на продукцию и изменения объемов будущих продаж. Медицинские работники используют временные ряды для изучения распространения болезни и прогнозирования числа будущих случаев заболевания в данном регионе. Сейсмологи изучают временные ряды для прогнозирования землетрясений. В каждом случае изучение исторических временных рядов является неотъемлемой частью процесса. Однако не все подходы годятся для анализа разных видов временных рядов, поэтому в данной главе мы рассмотрим множество примеров.

Существует широкий спектр методов описания временных рядов и прогнозирования будущих значений. Работающие с временными рядами могут обнаружить, что R предлагает наиболее исчерпывающий набор возможностей анализа. Однако в этой главе мы рассмотрим лишь некоторые из наиболее распространенных подходов к описанию и прогнозированию, а также функции R, используемые для их реализации. В табл. 15.1 перечислены временные ряды, которые мы будем анализировать. Все они доступны в базовом дистрибутиве R. Наборы данных сильно различаются по своим характеристикам, и, соответственно, различаются модели, лучше всего подходящие для их анализа.

Таблица 15.1. Наборы данных, использованные в этой главе

Временные ряды	Описание
`AirPassengers`	Ежемесячное количество пассажиров, перевозившихся авиакомпаниями в период 1949–1960
`JohnsonJohnson`	Поквартальная доходность акций Johnson & Johnson
`nhtemp`	Среднегодовые температуры в Нью-Хейвена (штат Коннектикут) в период 1912–1971
`Nile`	Годовой сток реки Нил
`sunspots`	Среднемесячное количество пятен на Солнце в период 1749–1983

Для начала мы познакомимся с методами создания временных рядов и управления ими, их описания и построения диаграмм, а также их разложения на компоненты разных уровней: общий тренд, сезонность и нерегулярность (ошибки). Затем займемся вопросами прогнозирования, начав с популярных подходов на основе экспоненциального моделирования, которые используют средневзвешенные значения временных рядов для прогнозирования будущих значений. Далее мы рассмотрим набор методов прогнозирования, называемых *авторегрессионными интегрированными моделями скользящих средних* (AutoRegressive Integrated Moving Averages, ARIMA), которые используют корреляции между недавними замерами данных и недавними ошибками прогнозирования для предсказания будущего. Далее будут представлены методы оценки адекватности моделей и точности их предсказаний. И в заключение будут перечислены ресурсы, где можно получить дополнительную информацию по этим темам.

Для опробования примеров программного кода из этой главы обязательно установите пакеты `xts`, `forecast`, `tseries` и `directlabels` (`install.packages(c("xts", "forecast", "tseries", "directlabels"))`.

15.1. Создание объекта временного ряда

Для работы с временными рядами в R их необходимо поместить в *объекты временных рядов* – структуру, содержащую наблюдения и соответствующие им отметки времени. После этого можно использовать многочисленные функции для управления ими, моделирования и построения диаграмм.

В R имеется множество пакетов, предлагающих различные структуры для хранения временных рядов (см. врезку «Объекты временных рядов в R»). В этой главе мы используем класс `xts`, предлагаемый пакетом `xts`. Он поддерживает временные ряды с измерениями, сделанными как через равные, так и через неравные промежутки времени, и предлагает множество функций для управления временными рядами.

Объекты временных рядов в R

В многообразии объектов, предлагаемых языком R для хранения временных рядов, легко потеряться. В базовой версии R имеется объект `ts`, предназначенный для хранения одного временного ряда с измерениями через равные интервалы времени, и `mts`, предназначенный для хранения нескольких временных рядов с измерениями через равные интервалы времени. Пакет `zoo` предлагает одноименный класс, способный хранить временные ряды с неравномерными интервалами измерений, а пакет `xts` – надмножество класса `zoo` с дополнительными вспомогательными функциями. Также популярностью пользуются форматы: `tsibble`, `timeSeries`, `irts` и `tis`. К счастью, пакет `tsbox` предлагает функции для преобразования таблиц данных в любой из этих форматов, а также для преобразования временных рядов из одного формата в другой.

Вот синтаксис создания объекта xts:

```
library(xts)
myseries <- xts(data, index)
```

где *data* – числовой вектор значений, а *index* – вектор с датами, соответствующими наблюдениям в *data*. В листинге 15.1 приводится пример временного ряда, включающего ежемесячные показатели продаж за два года, начиная с января 2018-го.

Листинг 15.1. Создание объекта временного ряда

```
library(xts)
sales <- c(18, 33, 41,  7, 34, 35, 24, 25, 24, 21, 25, 20,
           22, 31, 40, 29, 25, 21, 22, 54, 31, 25, 26, 35)
date  <- seq(from = as.Date("2018/1/1"),
               to = as.Date("2019/12/1"),
               by = "month")

sales.xts <- xts(sales, date)
```

Объекты временных рядов в формате xts можно делить на подмножества с помощью квадратных скобок []. Например, sales.xts["2018"] вернет все данные за 2018 год; sales.xts ["2018-3/2019-5"], вернет все данные с марта 2018 года по май 2019 года.

Существуют также функции apply.*, предназначенные для применения некоторой функции к каждому отдельному периоду в объекте временного ряда. Они особенно удобны для агрегирования временных рядов в более крупные периоды времени и имеют следующий синтаксис:

```
newseries <- apply.period(x, FUN, ...)
```

где *period* может быть daily, weekly, monthly, quarterly или yearly, *x* – объект временного ряда xts, *FUN* – применяемая функция, а ... – аргументы, передаваемые функции *FUN*.

Например, quarterlies <- apply.quarterly(sales.xts, sum) вернет временной ряд с восемью квартальными суммами продаж. Функцию sum можно заменить на mean, median, min, max или любую другую функцию, возвращающую единственное значение.

Функцию autoplot() из пакета forecast можно использовать для построения диаграмм временных рядов с использованием ggplot2, как показано в листинге 15.2.

Листинг 15.2. Два примера создания диаграмм на основе временных рядов

```
library(ggplot2)
library(forecast)
autoplot(sales.xts)                                    1
```

```
autoplot(sales.xts) +
  geom_line(color="blue") +                                    2
  scale_x_date(date_breaks="1 months",                         3
               date_labels="%b %y") +
  labs(x="", y="Sales", title="Customized Time Series Plot") +
  theme_bw() +                                                 4
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        panel.grid.minor.x=element_blank())
```

1 **Диаграмма по умолчанию.**

2 **Настройка цвета линии.**

3 **Определение меток на оси *X*.**

4 **Настройка темы.**

В первом примере функция `autoplot()` используется для создания простого линейного графика `ggplot2` 1 (рис. 15.2).

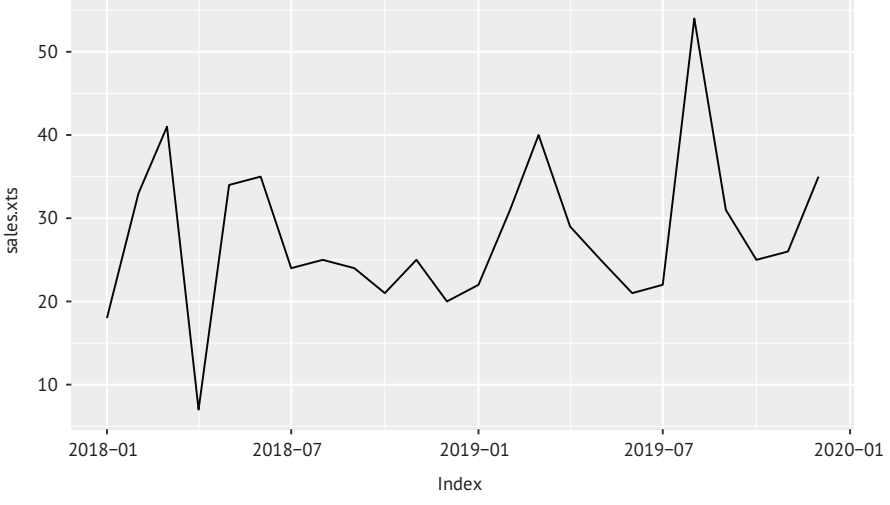


Рис. 15.2. График временного ряда с данными о продажах, созданный первым примером в листинге 15.1. Такой формат используется по умолчанию функцией autoplot()

Во втором примере выполняются дополнительные настройки диаграммы для придания большей привлекательности. Цвет линии изменен на синий 2. Функция `scale_x_date()` используется для создания более подробных меток на оси *x* 3. Параметр `data_breaks` определяет расстояние между делениями и может принимать такие значения, как `"1 day"`, `"2 weeks"`, `"5 years"` и т. д. Параметр `date_labels` определяет формат меток. Здесь спецификаторы `"%b %y"` задают месяц (3 буквы) и год (2 цифры) с пробелом между ними. Подробнее эти спецификаторы описаны в разделе 4.6. Наконец, для общего оформления выбирается черно-белая тема, метки на оси *x* поворачиваются на 90°, а вывод второстепенных

вертикальных линий сетки подавляется (4). На рис. 15.3 показан получившийся график.

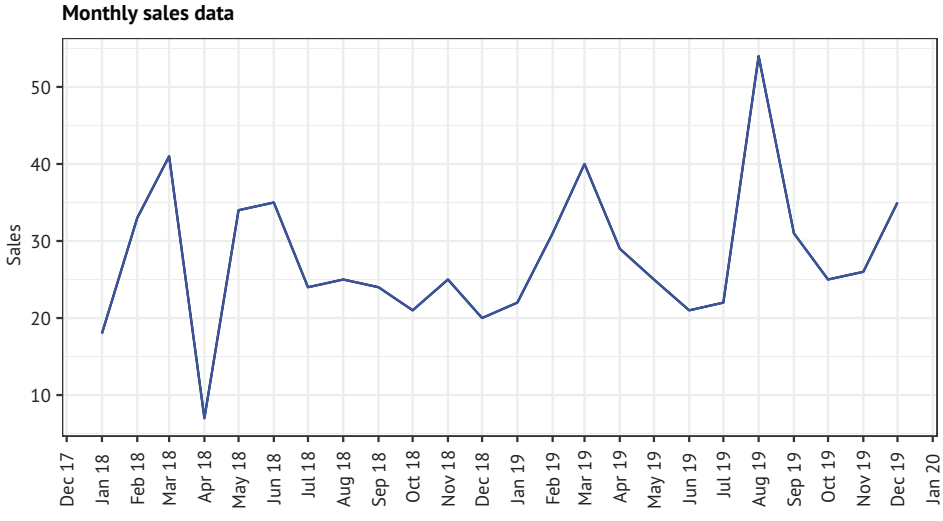


Рис. 15.3. График временного ряда с данными о продажах, созданный вторым примером в листинге 15.1. Для этой диаграммы изменен цвет линии, метки на оси *x* и выбрана более привлекательная тема оформления

Примеры временных рядов, входящие в базовый дистрибутив R (табл. 15.1), на самом деле имеют формат `ts`, но, к счастью, функции, представленные в этой главе, могут обрабатывать временные ряды как в формате `ts`, так и в формате `xts`.

15.2. Сглаживание и сезонная декомпозиция

Приступая к изучению наборов данных, аналитики первым делом вычисляют описательные статистики и строят – и только потом пытаются смоделировать – данные. Аналогично, приступая к изучению временных рядов, первым шагом необходимо получить их численное и визуальное описания. В этом разделе мы рассмотрим прием сглаживания временного ряда для выяснения его общей тенденции и декомпозицию временного ряда для выявления любых сезонных эффектов.

15.2.1 Сглаживание с помощью простых скользящих средних

Первый шаг при исследовании временного ряда – построение его графика, как было показано в листинге 15.1. Рассмотрим временной ряд `Nile`. Он представляет объем годового стока реки Нил в Асуане с 1871 по 1970 год. Соответствующий график показан вверху слева на рис. 15.4. Похоже, что постепенно объем годового стока уменьшается, но из года в год наблюдаются большие колебания.

Временные ряды обычно содержат значительный нерегулярный или ошибочный компонент. Чтобы заметить какие-либо закономерности, часто требуется построить сглаженную кривую, которая гасит эти колебания. Один из самых простых методов сглаживания временного ряда – использование метода скользящего среднего. Например, каждую точку в данных можно заменить средним значением этого наблюдения и одного наблюдения до и после него. Полученная таким способом величина называется *центрированным скользящим средним* и определяется как

$$S_t = (Y_{t-q} + ... + Y_t + ... + Y_{t+q}) / (2q + 1),$$

где S_t – сглаженное значение в момент времени t, а $k = 2q + 1$ – количество усредняемых наблюдений. Для k обычно выбирается нечетное число (в данном примере 3). При использовании центрированного скользящего среднего теряется $(k - 1) / 2$ наблюдений на каждом конце ряда.

В R есть несколько функций, вычисляющих простое скользящее среднее, включая `SMA()` в пакете `TTR`, `rollmean()` в пакете `zoo` и `ma()` в пакете `forecast`. Для сглаживания временного ряда `Nile` мы используем функцию `ma()`, входящую в состав базового дистрибутива R.

Код в листинге 15.3 создает диаграммы, отображающие исходный временной ряд и сглаженные версии с k, равным 3, 7 и 15. Графики показаны на рис. 15.3.

Листинг 15.3. Сглаживание простым скользящим средним

```
library(forecast)
library(ggplot2)

theme_set(theme_bw())
ylim <- c(min(Nile), max(Nile))

autoplot(Nile) +
  ggtitle("Raw time series") +
  scale_y_continuous(limits=ylim)

autoplot(ma(Nile, 3)) +
  ggtitle("Simple Moving Averages (k=3)") +
  scale_y_continuous(limits=ylim)

autoplot(ma(Nile, 7)) +
  ggtitle("Simple Moving Averages (k=7)") +
  scale_y_continuous(limits=ylim)

autoplot(ma(Nile, 15)) +
  ggtitle("Simple Moving Averages (k=15)") +
  scale_y_continuous(limits=ylim)
```

С увеличением k график становится все более гладким. Задача состоит в том, чтобы найти такое значение k, при котором

будут выделены основные закономерности в данных, но без избыточного или недостаточного сглаживания. Это больше искусство, чем наука, и я советую попробовать несколько значений k, прежде чем остановиться на каком-то одном. Судя по графикам на рис. 15.4, в период с 1892 по 1900 год объем годового стока явно уменьшился. Другие изменения должны интерпретироваться отдельно. Например, между 1941 и 1961 годами наблюдается небольшая тенденция к увеличению стока, но это могло быть и случайным изменением.

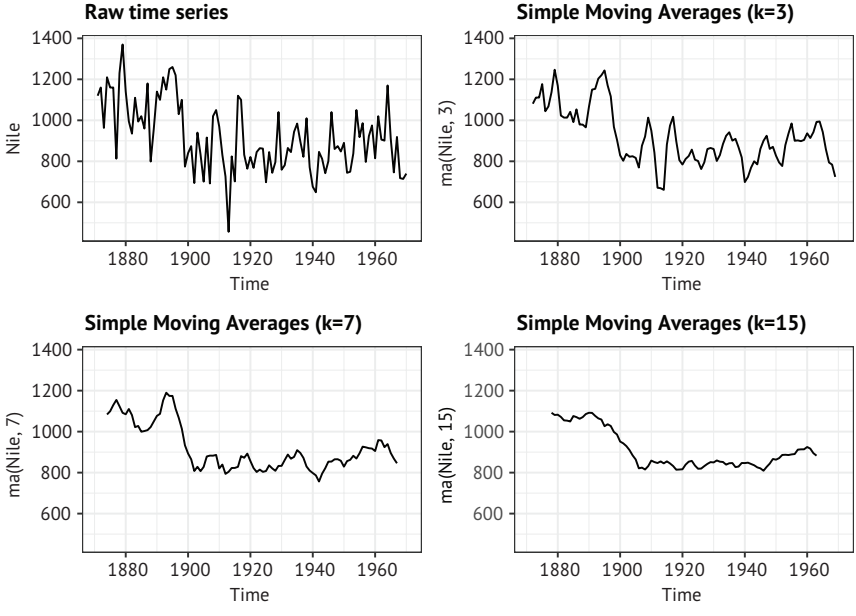


Рис. 15.4. Временной ряд Nile с данными об объеме годового стока реки Нил в Асуане с 1871 по 1970 год (вверху слева). Другие графики – это версии, сглаженные с использованием простого скользящего среднего на трех уровнях сглаживания (k = 3, 7 и 15)

Для временных рядов с периодичностью больше 1 (то есть с сезонным компонентом) нужно выйти за рамки описания общей тенденции. Для изучения как сезонных, так и общих тенденций можно использовать сезонную декомпозицию.

15.2.2. Сезонная декомпозиция

Временные ряды, имеющие сезонный аспект (например, ежемесячные или квартальные данные), можно разложить на компонент тренда, сезонный компонент и нерегулярный компонент. Компонент тренда фиксирует изменения уровня с течением времени. Сезонная составляющая фиксирует циклические эффекты, обусловленные, например, временем года. Нерегулярный компонент (или

ошибка) фиксирует влияния, которые не объясняются общим трендом и/или сезонными эффектами.

Декомпозиция может быть аддитивной или мультипликативной. В аддитивной модели компоненты суммируются, чтобы получить значения временного ряда. В частности:

$$Y_t = Тренд_t + Сезонность_t + Нерегулярность_t,$$

где каждое наблюдение в момент времени t является суммой вклада общего тренда, сезонного эффекта и нерегулярного компонента в момент времени t.

В мультипликативной модели компоненты – тренд, сезонный эффект и нерегулярность – перемножаются:

$$Y_t = Тренд_t \times Сезонность_t \times Нерегулярность_t.$$

На рис. 15.5 показано несколько примеров.

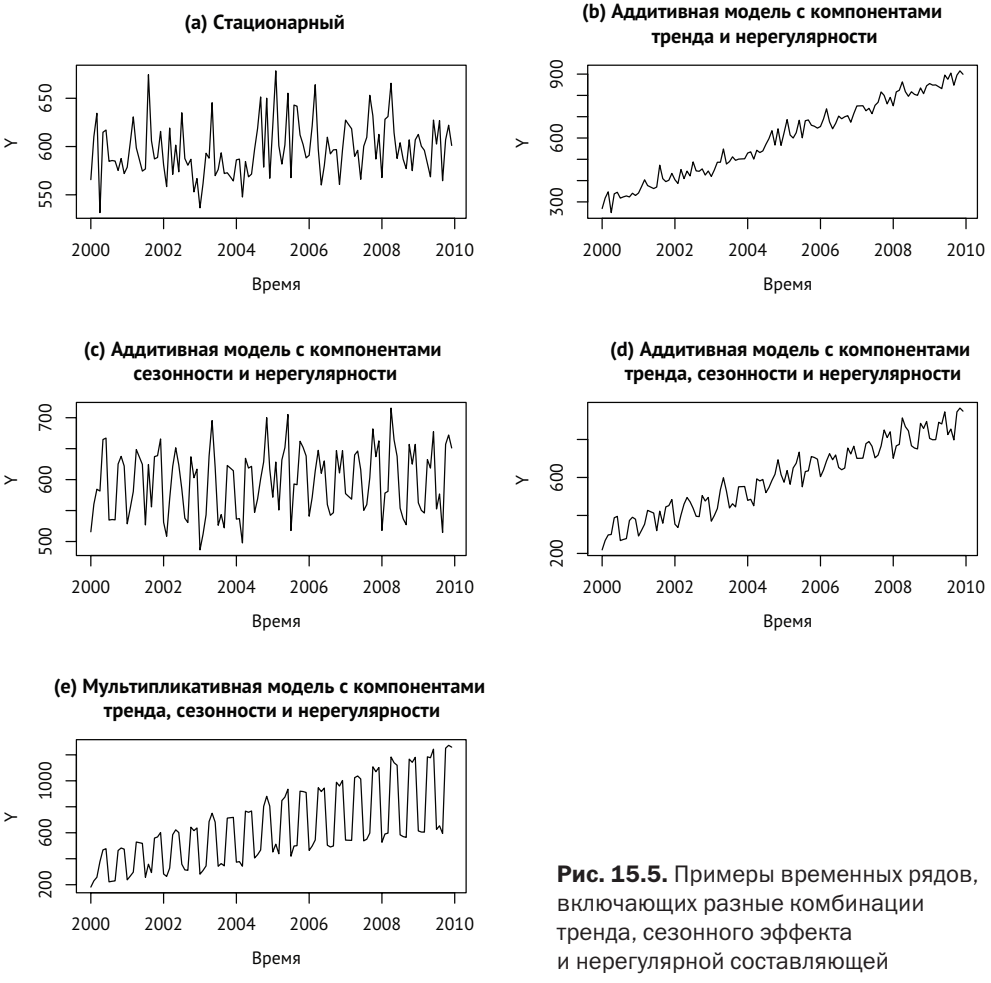


Рис. 15.5. Примеры временных рядов, включающих разные комбинации тренда, сезонного эффекта и нерегулярной составляющей

На первом графике (a) нет ни тренда, ни сезонной составляющей; на нем показано случайное колебание вокруг заданного уровня. На втором графике (b) наблюдается восходящий тренд с течением времени, а также случайные колебания. На третьем графике (c) имеют место сезонные эффекты и случайные колебания, но нет общей тенденции отклонения от горизонтальной линии. На четвертом графике (d) присутствуют все три компонента: восходящий тренд, сезонные и случайные колебания. Те же три компонента можно видеть на последнем графике (e), но здесь они объединяются мультипликативно. Обратите внимание, что изменчивость пропорциональна уровню: с увеличением уровня увеличивается и изменчивость. Это усиление (или возможное затухание), основанное на текущем уровне ряда, убедительно свидетельствует о мультипликативной модели.

Данный пример помогает прояснить разницу между аддитивной и мультипликативной моделями. Рассмотрим временной ряд с ежемесячными продажами мотоциклов за 10-летний период. В аддитивной модели с сезонным эффектом количество проданных мотоциклов имеет тенденцию к увеличению на 500 в ноябре и декабре (из-за рождественского ажиотажа) и уменьшению на 200 в январе (когда продажи замедляются). Сезонное увеличение или уменьшение не зависит от текущего объема продаж.

В мультипликативной модели с сезонным эффектом продажи мотоциклов в ноябре и декабре увеличиваются на 20 %, а в январе снижаются на 10 %. В этом случае влияние сезонного эффекта пропорционально текущему объему продаж, чего нельзя сказать об аддитивной модели. Во многих случаях мультипликативная модель оказывается более близкой к реальности.

Популярным методом разложения временного ряда на тренд, сезонные и нерегулярные составляющие является сезонная декомпозиция путем сглаживания кривой. В R такое сглаживание можно выполнить с помощью функции `stl()`:

```
stl(ts, s.window=, t.window=)
```

где *`ts`* – временной ряд, `s.window` определяет скорость изменения сезонных эффектов во времени, а `t.window` – скорость изменения тренда во времени. Параметр `s.window="periodic"` сообщает, что сезонные эффекты должны быть одинаковыми по годам. Обязательными являются только параметры *`ts`* и `s.window`. Подробности ищите в `help(stl)`.

Функция `stl()` поддерживает лишь аддитивные модели, но это не является серьезным ограничением. Мультипликативные модели можно преобразовать в аддитивные с помощью логарифмического преобразования:

$$\log(Y_t) = \log(Тренд_t \times Сезонность_t \times Нерегулярность_t)$$
$$= \log(Тренд_t) + \log(Сезонность_t) + \log(Нерегулярность_t).$$

После подгонки аддитивной модели к временному ряду, подвергнутому логарифмическому преобразованию, результаты можно преобразовать обратно в исходный масштаб. Рассмотрим пример.

Временной ряд `AirPassengers` входит в состав базового дистрибутива R и описывает ежемесячное количество (в тысячах) пассажиров, перевозившихся авиакомпаниями в период 1949–1960. Вверху на рис. 15.6 показан график, построенный по исходным данным. Как видите, изменчивость ряда (размах колебаний) постепенно увеличивается, что указывает на мультипликативную модель.

График внизу на рис. 15.6 отображает временной ряд, созданный путем логарифмирования каждого наблюдения. Дисперсия стабилизировалась, и логарифмированный ряд выглядит как пригодный для аддитивного разложения. Такое разложение выполняется с помощью функции `stl()`, как показано в листинге 15.4.

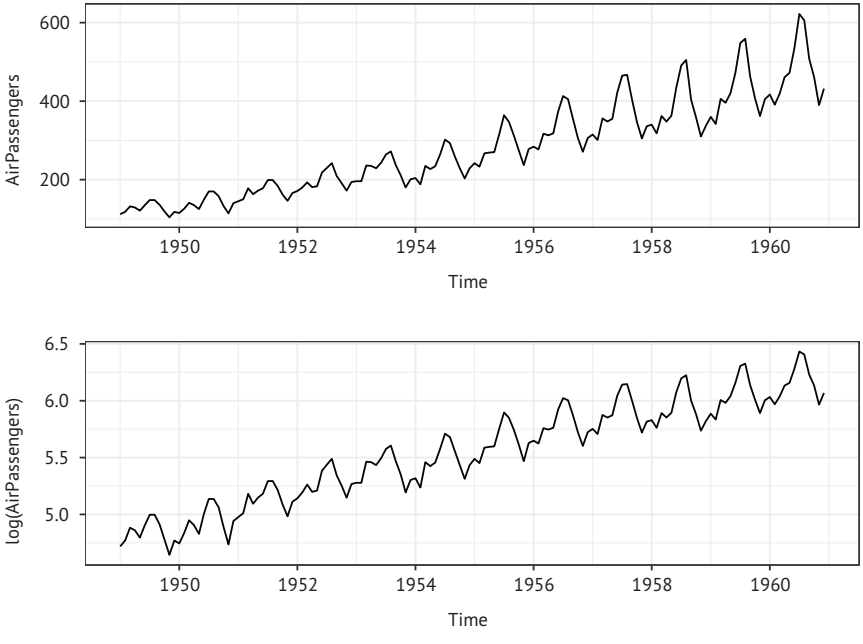


Рис. 15.6. График временного ряда AirPassengers (вверху). Временной ряд описывает ежемесячное количество (в тысячах) пассажиров, перевозившихся авиакомпаниями в период 1949–1960. После логарифмирования значений (внизу) дисперсия временного ряда стабилизировалась, и теперь данные лучше подходят для аддитивной сезонной декомпозиции

Листинг 15.4. Сезонная декомпозиция при помощи stl()

```
> library(forecast)
> library(ggplot2)
> autoplot(AirPassengers)                                    (1)
> lAirPassengers <- log(AirPassengers)
> autoplot(lAirPassengers, ylab="log(AirPassengers)")

> fit <- stl(lAirPassengers, s.window="period")              (2)
> autoplot(fit)

> fit$time.series                                            (3)

         seasonal trend  remainder
Jan 1949 -0.09164 4.829 -0.0192494
Feb 1949 -0.11403 4.830  0.0543448
Mar 1949  0.01587 4.831  0.0355884
Apr 1949 -0.01403 4.833  0.0404633
May 1949 -0.01502 4.835 -0.0245905
Jun 1949  0.10979 4.838 -0.0426814
... вывод опущен ...

> exp(fit$time.series)

         seasonal trend remainder
Jan 1949   0.9124 125.1    0.9809
Feb 1949   0.8922 125.3    1.0558
Mar 1949   1.0160 125.4    1.0362
Apr 1949   0.9861 125.6    1.0413
May 1949   0.9851 125.9    0.9757
Jun 1949   1.1160 126.2    0.9582

... вывод опущен ...
```

(1) Вывод графика временного ряда.

(2) Декомпозиция временного ряда.

(3) Компоненты для каждого наблюдения.

Сначала выводится график, отражающий исходные данные, после чего временной ряд преобразуется (1). Затем выполняется сезонная декомпозиция, и результат сохраняется в объекте с именем fit (2). После этого создаются графики (рис. 15.7), отражающие тренд, сезонную и нерегулярную составляющие с 1949 по 1960 год. Обратите внимание, что сезонные колебания должны оставаться одинаковыми в течение каждого года (это обеспечивает параметр `s.window="period"`). Тренд монотонно увеличивается, а сезонный эффект предполагает увеличение количества пассажиров летом (возможно, в период отпусков). Серые столбики справа отражают масштаб – все столбики представляют одну и ту же величину. Это удобно, потому что оси *y* различны для разных графиков.

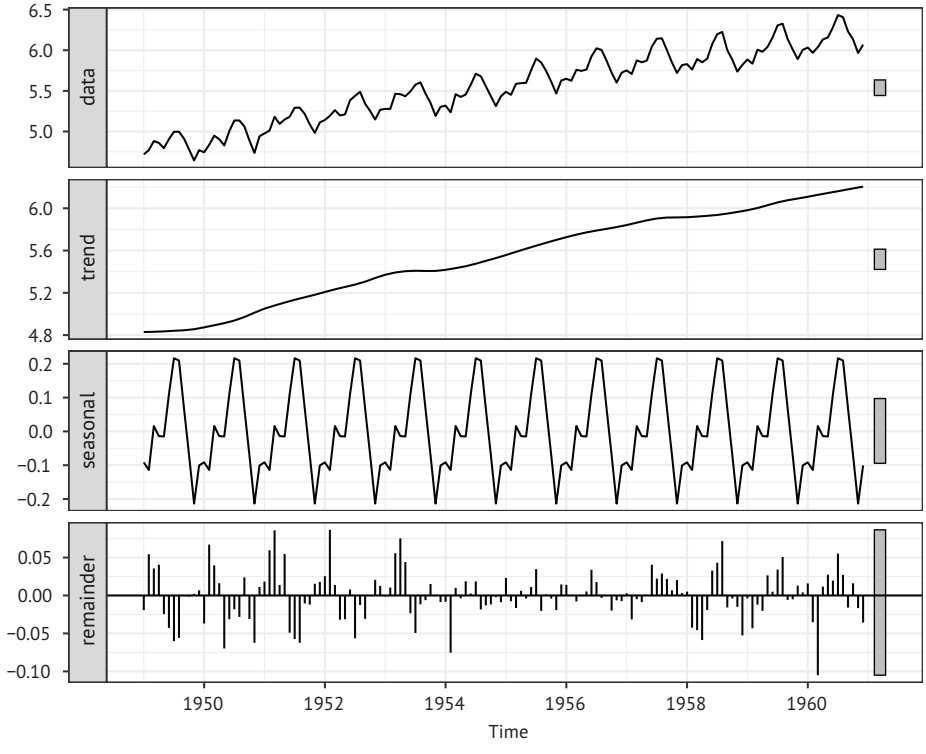


Рис. 15.7. Сезонная декомпозиция временного ряда AirPassengers после логарифмирования с использованием функции stl(). Временной ряд разбивается на тренд, сезонную и нерегулярную составляющие

Функция `stl()` возвращает объект, включающий компонент с именем `time.series`, содержащий тренд, сезонную и нерегулярную составляющие каждого наблюдения (3). В этом случае `fit$time.series` основан на логарифмированных данных из временного ряда. `exp(fit$time.series)` преобразует результаты разложения обратно в исходную метрику. Изучение сезонных эффектов показывает, что количество пассажиров увеличивается на 24 % в июле (множитель 1,24) и уменьшается на 20 % в ноябре (множитель 0,80).

Пакет `forecast` предоставляет дополнительные инструменты для визуализации результатов сезонной декомпозиции. В листинге 15.5 показано создание месячного и сезонного графиков.

Листинг 15.5. Месячный и сезонный графики

```
library(forecast)
library(ggplot2)
library(directlabels)
```

```
ggmonthplot(AirPassengers)  +                                    1
  labs(title="Month plot: AirPassengers",                        1
       x="",                                                     1
       y="Passengers (thousands)")                               1

p <- ggseasonplot(AirPassengers) + geom_point() +                2
  labs(title="Seasonal plot: AirPassengers",                     2
       x="",                                                     2
       y="Passengers (thousands)")                               2
direct.label(p)                                                  2
```

1 **Месячный график.**

2 **Сезонный график.**

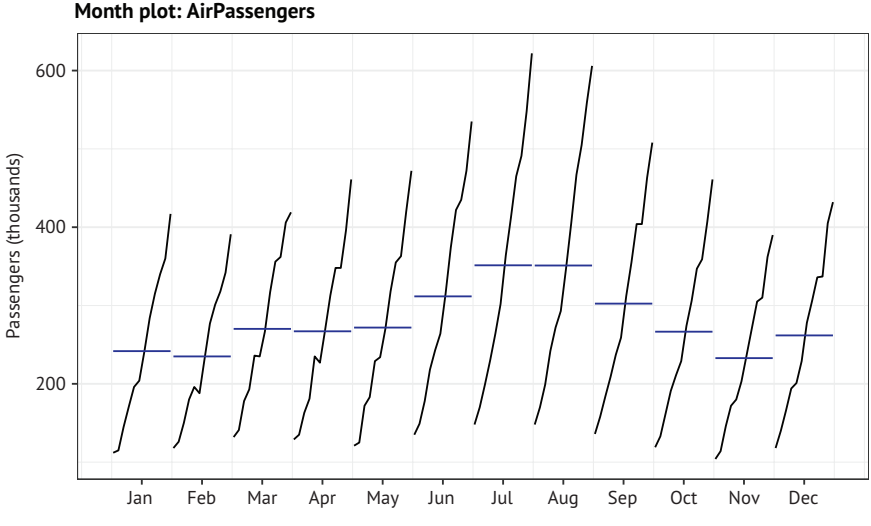


Рис. 15.8. Месячный график временного ряда AirPassengers. На месячном графике отображаются ряды для каждого месяца (все значения за январь с 1949 по 1960 год, все значения за февраль и т. д.), а также среднее значение для каждого ряда. Во всех месяцах наблюдается равномерная тенденция к увеличению, причем большинство пассажиров летают в июле и августе

На месячном графике (рис. 15.8) отображаются ряды для каждого месяца (все значения за январь с 1949 по 1960 год, все значения за февраль и т. д.) вместе со средними значениями для каждого ряда. Из этого графика видно, что в каждом месяце имеет место примерно одинаковая тенденция к увеличению. Кроме того, наибольшее количество пассажиров приходится на июль и август.

Сезонный график (рис. 15.9) отображает ряды по годам. И снова видна аналогичная картина с увеличением количества пассажиров каждый год и с той же сезонной моделью. По умолчанию пакет `ggplot2` создает легенду для переменной `year`. Пакет `directlabels` используется для размещения меток с номером года непосредственно на графике рядом с каждой линией, представляющей свой временной ряд.

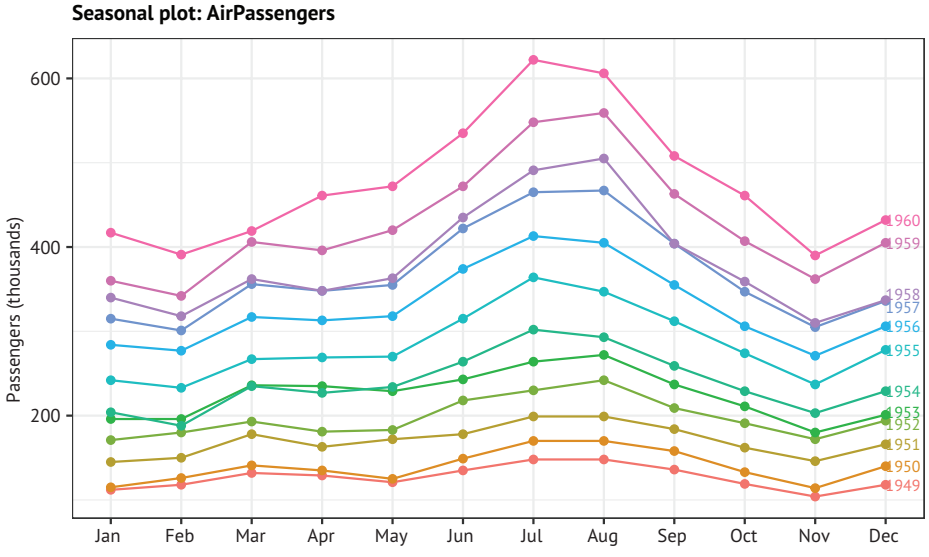


Рис. 15.9. Сезонный график для временного ряда AirPassengers. График показывает тенденцию к увеличению и похожую сезонную картину, повторяющуюся из года в год

Итак, мы описали временной ряд, но пока не предсказали никаких будущих значений. В следующем разделе мы рассмотрим использование экспоненциальных моделей для прогнозирования значений, выходящих за пределы доступных данных.

15.3. Экспоненциальные модели прогнозирования

Экспоненциальные модели являются одним из самых популярных подходов к прогнозированию будущих значений на основе имеющегося временного ряда. Они проще многих других типов моделей, но при этом способны давать хорошие краткосрочные прогнозы в широком диапазоне приложений. Они отличаются друг от друга составляющими моделируемого временного ряда. *Простая экспоненциальная модель* (также называется *одинарной экспоненциальной моделью*) используется, когда временной ряд имеет постоянный уровень и нерегулярный компонент в момент времени i, но не имеет ни тренда, ни сезонного компонента. *Двойная экспоненциальная модель* (также называется *экспоненциальным сглаживанием Хольта*) используется, когда временной ряд имеет тренд. Наконец, *тройная экспоненциальная модель* (также называется *экспоненциальным сглаживанием Хольта–Уинтерса*) используется, когда временной ряд имеет компоненты тренда и сезонности.

Подгонять экспоненциальные модели можно с помощью функции `ets()` из пакета `forecast`. Она имеет следующий синтаксис:

```
ets(ts, model="ZZZ")
```

где `ts` – временной ряд, а `model` задает модель тремя буквами. Первая буква обозначает тип ошибки, вторая – тип тренда, а третья – сезонный тип. Допустимые буквы: `A` – аддитивная модель, `M` – мультипликативная, `N` – нет типа и `Z` – автоматический выбор. В табл. 15.2 показаны примеры подгонки распространенных моделей.

Таблица 15.2. Функции подгонки простой, двойной и тройной моделей прогнозирования

Тип модели	Параметры подгонки	Функции
Простая	Без тренда и сезонной составляющей	`ets(ts, model="ANN")` `ses(ts)`
Двойная	С одним трендом	`ets(ts, model="AAN")` `holt(ts)`
Тройная	С трендом и сезонной составляющей	`ets(ts, model="AAA")` `hw(ts)`

Функции `ses()`, `holt()` и `hw()` – это просто удобные обертки для функции `ets()`, задающие определенные значения по умолчанию. Сначала мы рассмотрим самую простую экспоненциальную модель: простое экспоненциальное сглаживание.

15.3.1. Простое экспоненциальное сглаживание

Простое экспоненциальное сглаживание использует взвешенное среднее существующих значений временного ряда для краткосрочного прогнозирования будущих значений. Веса выбираются так, чтобы наблюдения оказывали экспоненциально уменьшающееся влияние на среднее по мере удаления в прошлое.

Модель простого экспоненциального сглаживания предполагает, что наблюдение во временном ряду может быть описано как

Y_t = *уровень* + *нерегулярность*$_t$.

Прогнозное значение в момент времени Y_{t+1} (называется *на шаг вперед в будущее*) записывается как

$$Y_{t+1} = c_0 Y_t + c_1 Y_{t-1} + c_2 Y_{t-2} + ...,$$

где $c_i = \alpha(1-\alpha)^i$, $t = 0, 1, 2, ...$ и $0 \le \alpha \le 1$. Сумма весов c_i равна единице, и прогноз на один шаг вперед можно рассматривать как взвешенное среднее значение текущего значения и всех прошлых значений временного ряда. Параметр альфа (α) определяет скорость убывания весов. Чем ближе значение альфа к единице, тем больший вес придается недавним наблюдениям. Чем ближе альфа к нулю, тем больший вес придается прошлым наблюдениям. Фактическое значение альфа обычно выбирается компьютером для оптимизации критерия соответствия. Общим критерием соответствия является

сумма квадратов ошибок между фактическими и прогнозируемыми значениями. Рассмотрим пример, чтобы лучше понять идею.

Временной ряд `nhtemp` содержит среднегодовые температуры в градусах Фаренгейта в Нью-Хейвене, штат Коннектикут, с 1912 по 1971 год. На рис. 15.10 показан линейный график этого временного ряда.

Очевидной тенденции не наблюдается, и в годовых данных отсутствует сезонный компонент, поэтому разумно будет для начала выбрать простую экспоненциальную модель. Код, осуществляющий прогнозирование на один шаг вперед с использованием функции `ses()`, приводится в листинге 15.6.

Листинг 15.6. Простое экспоненциальное сглаживание

```
> library(forecast)
> fit <- ets(nhtemp, model="ANN")      (1)
> fit

ETS(A,N,N)

Call:
 ets(y = nhtemp, model = "ANN")

  Smoothing parameters:
    alpha = 0.1819

  Initial states:
    l = 50.2762

  sigma:  1.1455

     AIC     AICc      BIC
265.9298 266.3584 272.2129

> forecast(fit, 1)                     (2)

     Point Forecast  Lo 80  Hi 80  Lo 95  Hi 95
1972          51.87 50.402 53.338 49.625 54.115

> autoplot(forecast(fit, 1)) +
  labs(x = "Year",
       y = expression(paste("Temperature (", degree*F,")",)),
       title = "New Haven Annual Mean Temperature")

> accuracy(fit)                        (3)

                ME  RMSE   MAE  MPE  MAPE   MASE
Training set 0.146 1.126 0.895 0.242 1.749 0.751
```

(1) **Подгонка модели.**

(2) **Прогнозирование на шаг вперед.**

(3) **Вывод метрик точности.**

Инструкция `ets(mode="ANN")` подбирает простую экспоненциальную модель к временному ряду `nhtemp` (1). Буква `"A"` указывает, что модель считается аддитивной, а буквы `"NN"` указывают на отсутствие тренда и сезонной составляющей. Относительно низкое значение параметра альфа (0,18) указывает на то, что в прогнозе учитываются не только недавние, но и отдаленные наблюдения. Это значение выбирается автоматически, чтобы максимизировать соответствие модели заданному набору данных.

Для прогнозирования по временному ряду на k шагов вперед используется функция `forecast()`. Она имеет следующий синтаксис: `forecast(`*`fit, k`*`)`. Для этого ряда прогноз на один шаг вперед равен 51,9 °F с доверительным интервалом 95 % (от 49,6 °F до 54,1 °F) (2). На рис. 15.10 показаны график временного ряда, значение прогноза и доверительные интервалы 80 % и 95 %.

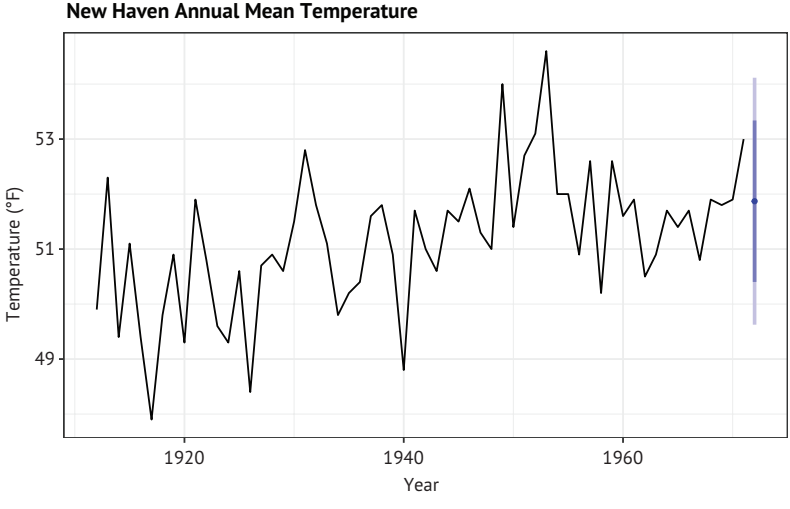


Рис. 15.10. Среднегодовые температуры в Нью-Хейвене, штат Коннектикут, и прогноз на один шаг вперед на основе простого экспоненциального сглаживания с использованием функции ets()

В пакете `forecast` также имеется функция `accuracy()`, которая отображает наиболее популярные метрики точности прогнозов по временным рядам (3). Все они описаны в табл. 15.3. Метрика e_t представляет ошибку, или нерегулярный компонент каждого наблюдения $(Y_i - \hat{Y}_i)$.

Средняя ошибка и средняя процентная ошибка могут быть не особенно полезными, потому что положительные и отрицательные ошибки могут компенсироваться. RMSE дает квадратный корень из среднеквадратичной ошибки, которая в данном случае составляет 1,13 °F. Средняя абсолютная процентная ошибка сообщает об ошибке в процентах от значений временного ряда. Она безразмерная

и может использоваться для сравнения точности прогнозов в разных временных рядах, но предполагает шкалу измерения с истинной нулевой точкой (например, количество пассажиров в день). Поскольку шкала Фаренгейта не имеет истинного нуля, эту оценку лучше не использовать в данном случае. Средняя абсолютная масштабированная ошибка является самой последней оценкой точности и используется для сравнения точности прогнозов по временным рядам с разными масштабами. Не существует единственной наилучшей оценки точности прогнозирования, но RMSE, безусловно, используется и упоминается в литературе чаще других.

Таблица 15.3. Метрики точности прогнозов

Метрика	Сокращение	Определение
Средняя ошибка	ME	mean(e_t)
Среднеквадратичная ошибка	RMSE	sqrt(mean(e_t^2))
Средняя абсолютная ошибка	MAE	mean(|e_t|)
Средняя процентная ошибка	MPE	mean(100 × e_t / Y_t)
Средняя абсолютная процентная ошибка	MAPE	mean(|100 × e_t / Y_t|)
Средняя абсолютная масштабированная ошибка	MASE	mean(|q_t|), где $q_t = e_t$ / (1/(T–1) * sum(|$y_t - y_{t-1}$|)), T – число наблюдений, а sum выполняет суммирование от $t = 2$ до $t = T$

Простое экспоненциальное сглаживание предполагает отсутствие тренда или сезонной составляющей. В следующем разделе мы рассмотрим экспоненциальные модели, когда в исходных данных имеются оба этих компонента.

15.3.2. Экспоненциальное сглаживание Холта и Холта–Уинтерса

Экспоненциальное сглаживание Холта может применяться к временным рядам, имеющим общий уровень и тренд (наклон). Модель, представляющая наблюдение в момент времени t, имеет вид:

$$Y_t = \textit{уровень} + \textit{наклон} \times t + \textit{нерегулярность}_t.$$

Параметр альфа управляет экспоненциальным затуханием уровня, а параметр бета – экспоненциальным затуханием наклона. Значения обоих параметров меняются в диапазоне от нуля до единицы, причем бóльшие значения придают больший вес недавним наблюдениям.

Экспоненциальное сглаживание Холта–Уинтерса можно использовать для подгонки модели к временному ряду, имеющему общий уровень, тренд и сезонный компонент. Модель имеет вид:

$$Y_t = уровень + наклон \times t + s_t + нерегулярность_t$$

где s_t представляет влияние сезонных колебаний в момент времени t. В дополнение к параметрам альфа и бета имеется параметр гамма, управляющий экспоненциальным затуханием сезонной составляющей. Как и другие параметры, он изменяется в диапазоне от нуля до единицы, и чем больше его значение, тем больший вес придается недавним наблюдениям при расчете влияния сезонного компонента.

В разделе 15.2 мы разложили временной ряд, описывающий пассажиропоток международных авиалиний, на аддитивные трендовые, сезонные и нерегулярные компоненты. Давайте воспользуемся экспоненциальной моделью для прогнозирования изменения пассажиропотока в будущем. И снова используем логарифмические значения, чтобы обеспечить соответствие аддитивной модели данным. Код в листинге 15.7 применяет метод экспоненциального сглаживания Холта–Уинтерса для прогнозирования следующих пяти значений по данным из временного ряда `AirPassengers`.

Листинг 15.7. Экспоненциальное сглаживание с учетом уровня, наклона и сезонной составляющей

```
> library(forecast)
> fit <- ets(log(AirPassengers), model="AAA")
> fit

ETS(A,A,A)

Call:
 ets(y = log(AirPassengers), model = "AAA")

    Smoothing parameters:                                    (1)
    alpha = 0.6975
    beta  = 0.0031
    gamma = 1e-04

  Initial states:
    l = 4.7925
    b = 0.0111
    s = -0.1045 -0.2206 -0.0787 0.0562 0.2049 0.2149
           0.1146 -0.0081 -0.0059 0.0225 -0.1113 -0.0841

  sigma:  0.0383

     AIC     AICc      BIC
-207.17 -202.31 -156.68

>accuracy(fit)

                      ME     RMSE      MAE       MPE    MAPE     MASE
Training set -0.0018307 0.03607 0.027709 -0.034356 0.50791 0.22892
```

```
> pred <- forecast(fit, 5)                                    (2)
> pred
         Point Forecast  Lo 80  Hi 80  Lo 95  Hi 95
Jan 1961         6.1093 6.0603 6.1584 6.0344 6.1843
Feb 1961         6.0925 6.0327 6.1524 6.0010 6.1841
Mar 1961         6.2366 6.1675 6.3057 6.1310 6.3423
Apr 1961         6.2185 6.1412 6.2958 6.1003 6.3367
May 1961         6.2267 6.1420 6.3115 6.0971 6.3564

> autoplot(pred) +
  labs(title = "Forecast for Air Travel",
       y = "Log(AirPassengers)",
       x ="Time")

> pred$mean <- exp(pred$mean)                                 (3)
> pred$lower <- exp(pred$lower)                               (3)
> pred$upper <- exp(pred$upper)                               (3)
> p <- cbind(pred$mean, pred$lower, pred$upper)
> dimnames(p)[[2]] <- c("mean", "Lo 80", "Lo 95", "Hi 80", "Hi 95")
> p

            mean  Lo 80  Lo 95  Hi 80  Hi 95
Jan 1961 450.04 428.51 417.53 472.65 485.08
Feb 1961 442.54 416.83 403.83 469.85 484.97
Mar 1961 511.13 477.01 459.88 547.69 568.10
Apr 1961 501.97 464.63 446.00 542.30 564.95
May 1961 506.10 464.97 444.57 550.87 576.15
```

(1) **Параметры сглаживания.**

(2) **Вычисление прогнозных значений.**

(3) **Преобразование прогнозов в исходный масштаб.**

Подобранные параметры сглаживания для уровня (0,70), тренда (0,0004) и сезонных составляющих (0,003) показаны в (1). Низкое значение тренда (0,0001) не означает отсутствия наклона; оно лишь указывает, что наклон, оцененный по ранним наблюдениям, не нуждался в корректировке.

Функция `forecast()` выдает прогнозы на следующие пять месяцев (2), как показано на рис. 15.11. Поскольку прогнозы представлены в логарифмическом масштабе, для получения значений в исходном масштабе (количество в тысячах пассажиров (3)) используется возведение в степень. Матрица `pred$mean` содержит точечные прогнозы, а матрицы `pred$lower` и `pred$upper` содержат 80%-ный и 95%-ный нижний и верхний доверительные пределы соответственно. Функция `exp()` используется для возврата прогнозов в исходный масштаб, а `cbind()` создает единую таблицу. Таким образом, модель предсказывает 509 200 пассажиров в марте с доверительной вероятностью 95 %, т. е. в диапазоне от 454 900 до 570 000.

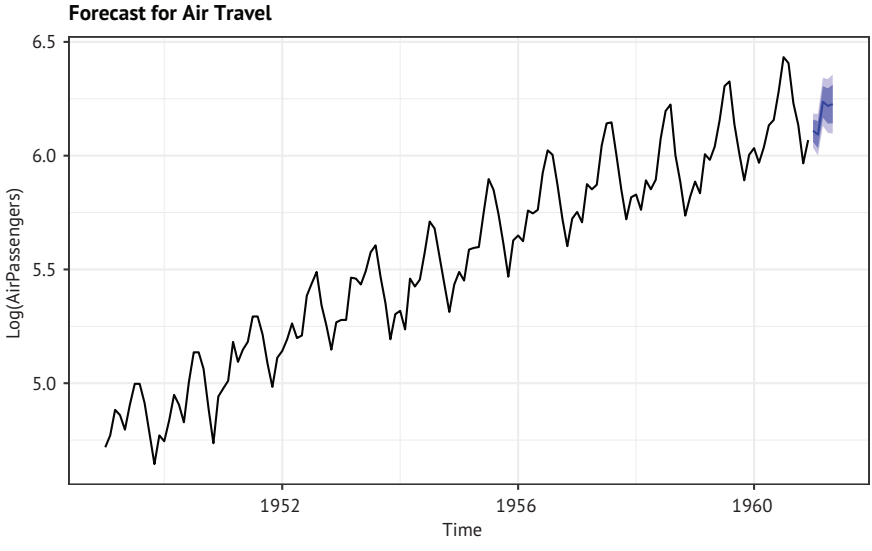


Рис. 15.11. Пятилетний прогноз логарифма количества авиапассажиров в тысячах на основе модели экспоненциального сглаживания Холта–Уинтерса. Данные взяты из временного ряда AirPassengers

15.3.3. Функция ets() и автоматизация прогнозирования

Функция `ets()` предлагает дополнительные возможности. Ее можно использовать для подбора экспоненциальных моделей с мультипликативными компонентами, добавления демпфирующего компонента и выполнения автоматических прогнозов. Рассмотрим их по очереди.

В предыдущем разделе мы подобрали аддитивную экспоненциальную модель к логарифмам данных во временном ряду `AirPassengers`. Аналогично можно подобрать мультипликативную модель. Вызов функции будет выглядеть так: `ets(AirPassengers, model="MAM")`. Тренд остается аддитивным, но предполагается, что компоненты сезонности и нерегулярности являются мультипликативными. При использовании мультипликативной модели статистики точности и прогнозируемые значения сообщаются в исходной метрике (в тысячах пассажиров), что, несомненно, является преимуществом.

Функция `ets()` также может подобрать демпфирующий компонент. Прогнозы по временным рядам часто предполагают, что тренд будет продолжаться вечно (рынок жилья, например, согласны?). Демпфирующий компонент с течением времени приводит тренд к горизонтальной асимптоте. Во многих случаях демпфированная модель дает более реалистичные прогнозы.

Наконец, можно вызвать функцию `ets()` для автоматического подбора модели для данных. Давайте выполним автоматическую подгонку экспоненциальной модели к данным Johnson & Johnson, описанным в начале этой главы, как показано в листинге 15.8.

Листинг 15.8. Автоматическая подгонка модели экспоненциального прогнозирования с помощью ets()

```
> library(forecast)
> fit <- ets(JohnsonJohnson)
> fit

ETS(M,M,M)

Call:
 ets(y = JohnsonJohnson)

    Smoothing parameters:
    alpha = 0.2776
    beta  = 0.0636
    gamma = 0.5867

  Initial states:
    l = 0.6276
    b = 0.0165
    s = -0.2293 0.1913 -0.0074 0.0454

  sigma:  0.0921

   AIC   AICc    BIC
163.64 166.07 185.52

> autoplot(forecast(fit)) +
  labs(x = "Time",
       y = "Quarterly Earnings (Dollars)",
       title="Johnson and Johnson Forecasts")
```

Поскольку модель не указана, код выполняет поиск по широкому набору моделей, чтобы найти ту, которая минимизирует критерий соответствия (логарифмическое правдоподобие по умолчанию). Выбранная модель имеет мультипликативную составляющую тренда, сезонности и ошибки. На рис. 15.12 показан график вместе с прогнозами на следующие восемь кварталов (по умолчанию).

Как отмечалось выше, экспоненциальное моделирование временных рядов пользуется большой популярностью, потому что во многих ситуациях дает хорошие краткосрочные прогнозы. Вторым популярным подходом является методология Бокса–Дженкинса (Box–Jenkins), обычно называемая авторегрессионными интегрированными моделями скользящих средних (AutoRegressive Integrated Moving Averages, ARIMA). Они описаны в следующем разделе.

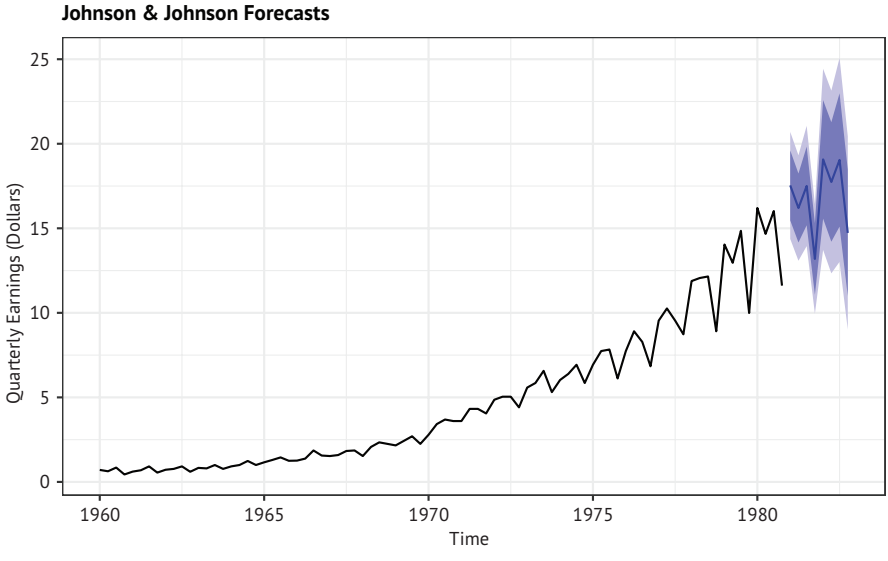


Рис. 15.12. Прогноз, полученный методом мультипликативного экспоненциального сглаживания с трендом и сезонным компонентом. Прогнозы представлены пунктирной линией, а доверительные интервалы 80 % и 95 % – светло- и темно-синим цветом соответственно

15.4. Модели прогнозирования ARIMA

В подходе на основе авторегрессионных интегрированных моделей скользящих средних (ARIMA) прогнозируемые значения являются линейной функцией недавних фактических значений и недавних ошибок прогнозирования (остатков). ARIMA – это комплексный подход к прогнозированию. В этом разделе мы ограничимся обсуждением моделей ARIMA для временных рядов без сезонной составляющей.

Перед описанием моделей ARIMA необходимо определить некоторые термины, включая лаг (сдвиг), автокорреляцию, частичную автокорреляцию, дифференцирование и стационарность. Все эти определения даны в следующем разделе.

15.4.1. Основные понятия

Под *лагом* временного ряда подразумевается сдвиг назад на заданное количество наблюдений. Рассмотрим несколько первых наблюдений из временного ряда `Nile`, представленных в табл. 15.4. Лаг 0 – это временной ряд без сдвига. Лаг 1 – временной ряд, сдвинутый на одну позицию влево. Лаг 2 сдвигает временной ряд на две позиции влево и т. д. Сдвигать временные ряды можно с помощью функции `lag(ts, k)`, где *ts* – временной ряд, а *k* – величина лага.

Таблица 15.4. Временной ряд Nile с различными лагами

Лаг	1869	1870	1871	1872	1873	1874	1875	...
0			1120	1160	963	1210	1160	...
1		1120	1160	963	1210	1160	1160	...
2	1120	1160	963	1210	1160	1160	813	...

Автокорреляция измеряет силу взаимосвязей между наблюдениями во временном ряду. AC_k – это корреляция между набором наблюдений (Y_t) и наблюдениями на k периодов ранее ($Y_{t\,k}$). Таким образом, AC_1 – это корреляция между временными рядами с лагом 1 и с лагом 0, AC_2 – корреляция между временными рядами с лагом 2 и с лагом 0 и т. д. При построении диаграммы с этими корреляциями (AC_1, AC_2, ..., AC_k) получается график *автокорреляционной функции* (ACF). График ACF используется для выбора подходящих параметров модели ARIMA и оценки адекватности окончательной модели.

График ACF можно создать с помощью функции `Acf()` из пакета `forecast`: `Acf(ts)`, где *`ts`* – исходный временной ряд. График ACF для временного ряда `Nile` с k от 1 до 18 показан вверху на рис. 15.13.

Частичная автокорреляция (PACF) – это корреляция между Y_t и Y_{t-k} с удалением влияния всех значений Y между ними (Y_{t-1}, Y_{t-2}, ..., Y_{t-k+1}). Частичные автокорреляции тоже можно вычислить для нескольких значений k, а построить график PACF можно с помощью функции `Pacf()` из пакета `forecast`: `Pacf(ts)`, где *`ts`* – оцениваемый временной ряд. Графики PACF тоже используются для определения наиболее подходящих параметров моделей ARIMA. Пример графика PACF для временного ряда `Nile` показан внизу на рис. 15.13.

Модели ARIMA разработаны для подгонки под *стационарные* временные ряды (или временные ряды, которые можно преобразовать в стационарные). Стационарными называются такие временные ряды, статистические свойства которых не меняются со временем. Например, среднее значение и дисперсия Y_t постоянны, также не меняются со временем автокорреляции для любого лага k.

Иногда может потребоваться преобразовать значения временного ряда для достижения постоянной дисперсии, прежде чем переходить к подгонке модели ARIMA. В таких случаях часто оказывается полезным логарифмическое преобразование, как было показано в разделе 15.1.3. Также могут пригодиться другие преобразования, такие как преобразование Бокса–Кокса (Box-Cox), описанное в разделе 8.5.2.

Поскольку предполагается, что стационарные временные ряды имеют постоянные средние значения, они не могут иметь тренда. Многие нестационарные временные ряды можно сделать стационарными путем *дифференцирования*. При дифференцировании каждое значение временного ряда Y_t заменяется на $Y_{t-1} - Y_t$. Однократное

дифференцирование временного ряда удаляет линейный тренд. Повторное дифференцирование удаляет квадратичный тренд. Третье дифференцирование удаляет кубический тренд. Однако на практике редко бывает необходимо дифференцировать более двух раз.

Дифференцирование можно выполнить с помощью функции `diff()`: `diff(ts, differences=d)`, где *d* определяет количество дифференцирований временного ряда *ts*. По умолчанию `d=1`. Для подбора наилучшего значения *d* можно использовать функцию `ndiffs()` из пакета `forecast`: `ndiffs(ts)`.

Стационарность часто оценивается визуальным осмотром графика временного ряда. Если дисперсия непостоянна, то производится преобразование данных. Если наблюдается какой-то тренд, то данные дифференцируются. Также для оценки предположения о стационарности можно использовать статистическую процедуру, называемую *расширенным критерием Дикки–Фуллера* (Augmented Dickey–Fuller test, ADF). Этот критерий можно вычислить с помощью функции `adf.test()` из пакета `tseries`: `adf.test(ts)`, где *ts* – оцениваемый временной ряд. Достоверность результата свидетельствует о стационарности.

В заключение отмечу, что графики ACF и PACF используются для определения параметров моделей ARIMA. Стационарность является важным допущением, и преобразования с дифференцированием используются для достижения стационарности. После знакомства с этими основными понятиями мы можем перейти к подгонке моделей с компонентами авторегрессии (AR), скользящих средних (MA) или обоими (ARMA). Под занавес мы рассмотрим модели ARIMA, включающие компоненты ARMA и дифференцирование для достижения стационарности.

15.4.2. Модели ARMA и ARIMA

В *авторегрессионной* модели порядка *p* каждое значение во временном ряду прогнозируется на основе линейной комбинации предыдущих *p* значений.

$$AR(p): Y_t = \mu + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p} + \varepsilon_t,$$

где Y_t – данное значение ряда, μ – среднее значение ряда, коэффициенты β – веса, а ε_t – нерегулярная составляющая. В модели *скользящего среднего* порядка *q* каждое значение во временном ряду предсказывается на основе линейной комбинации *q* предыдущих ошибок. В таком случае:

$$MA(q): Y_t = \mu + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_p \varepsilon_{t-q} + \varepsilon_t,$$

где ε – ошибки предсказания, а θ – веса. (Важно отметить, что описанные здесь скользящие средние не являются простыми скользящими средними, описанными в разделе 15.1.2.)

Объединение двух подходов дает модель ARMA(p, q) вида:

$$Y_t = \mu + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p} - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_p \varepsilon_{t-q} + \varepsilon_t,$$

которая предсказывает каждое значение временного ряда на основе предыдущих p значений и остатков q.

Модель ARIMA(p, d, q) – это модель, применившая дифференцирование к временному ряду d раз, а полученные значения прогнозируются на основе предыдущих p фактических значений и q ошибок. Прогнозы не дифференцируются и не интегрируются для получения окончательного прогноза.

При моделировании с помощью ARIMA выполняются следующие шаги:

1. Проверить стационарность временного ряда.
2. Определить разумную модель или модели (возможные значения p и q).
3. Выполнить подгонку модели.
4. Оценить адекватность модели, включая статистические допущения и прогностическую точность.
5. Вычислить прогнозы.

Давайте применим эти шаги и подберем модель ARIMA к временному ряду `Nile`.

Проверка стационарности временного ряда

Сначала построим график временного ряда и оценим его стационарность (листинг 15.7 и верхний график на рис. 15.13). Дисперсия кажется стабильной в течение лет, когда проводились наблюдения, поэтому в преобразовании нет необходимости. Возможен тренд, который подтверждается результатами функции `ndiffs()`.

Листинг 15.9. Преобразование временного ряда и оценка стационарности

```
> library(forecast)
> library(tseries)
> autoplot(Nile)
> ndiffs(Nile)

[1] 1

> dNile <- diff(Nile)
> autoplot(dNile)
> adf.test(dNile)

    Augmented Dickey-Fuller Test

data:  dNile
Dickey-Fuller = -6.5924, Lag order = 4, p-value = 0.01
alternative hypothesis: stationary
```

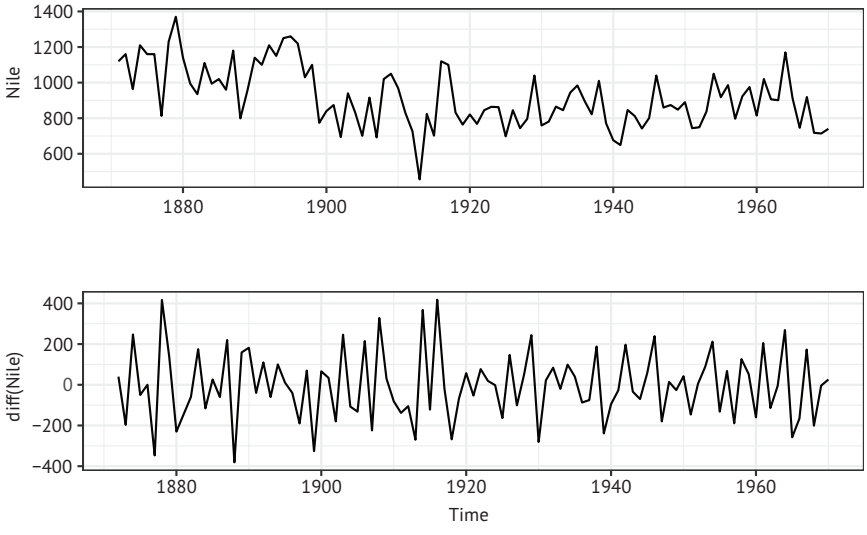


Рис. 15.13. Временной ряд, отражающий годовой сток реки Нил в Асуане с 1871 по 1970. Вверху – исходный временной ряд, внизу – однократно дифференцированный. Дифференциация устраняет тенденцию к уменьшению, очевидную на графике с исходными значениями

Дифференциация выполнена один раз (лаг = 1, по умолчанию), и полученный результат сохраняется в `dNile`. Результат дифференциации показан внизу на рис. 15.13 и, безусловно, выглядит более стационарным. Применение критерия ADF к дифференцированному ряду показывает, что теперь он стационарен, поэтому можно переходить к следующему шагу.

Определение одной или нескольких разумных моделей

Возможные модели выбираются на основе графиков ACF и PACF:

```
autoplot(Acf(dNile))
autoplot(Pacf(dNile))
```

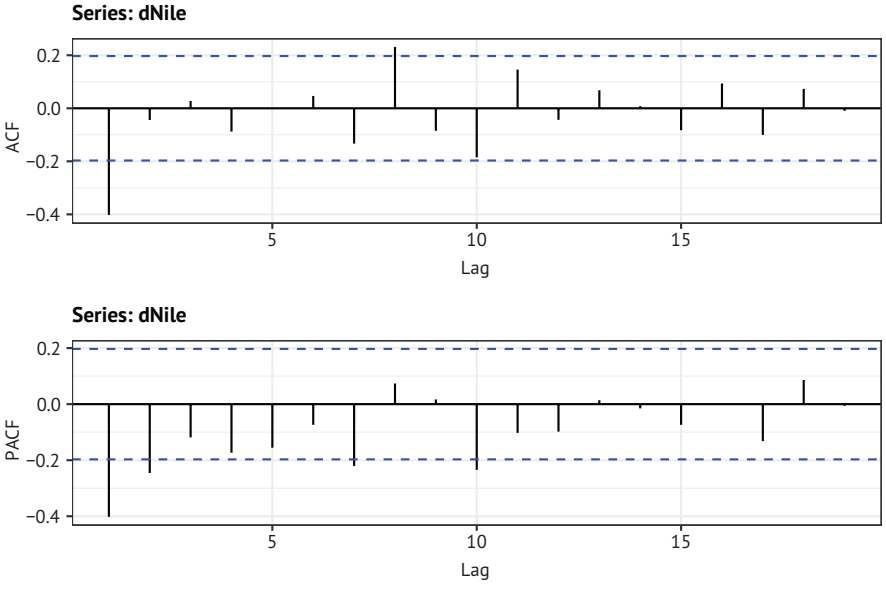


Рис. 15.14. Графики автокорреляции и частичной автокорреляции для исходного и дифференцированного временных рядов Nile

Цель состоит в том, чтобы определить оптимальные параметры p, d и q. Вы уже знаете, что $d = 1$ (из предыдущего раздела). Параметры p и q можно получить, сравнивая графики ACF и PACF с рекомендациями, приведенными в табл. 15.5.

Таблица 15.5. Рекомендации по выбору параметров модели ARIMA

Модель	ACF	PACF
ARIMA(p, d, 0)	Сходит на нет	Ноль после лага p
ARIMA(0, d, q)	Ноль после лага q	Сходит на нет
ARIMA(p, d, q)	Сходит на нет	Сходит на нет

Результаты в табл. 15.5 являются теоретическими, и фактические значения ACF и PACF могут не совпадать с ними в точности, но их можно использовать для приблизительного выбора моделей, которые можно попробовать. Для временного ряда `Nile` (рис. 15.13) имеется одна большая автокорреляция при лаге = 1, а частичные автокорреляции сходят на нет по мере увеличения лага. Отсюда следует, что можно попробовать модель ARIMA(0, 1, 1).

Подгонка модели

Подгонка модели ARIMA производится вызовом функции `Arima()`, имеющей следующий синтаксис: `Arima(ts, order=c(q, d, q))`. В лис-

тинге 15.10 показан результат подгонки модели ARIMA(0, 1, 1) к временному ряду `Nile`.

Листинг 15.10. Подгонка модели ARIMA

```
> library(forecast)
> fit <- arima(Nile, order=c(0,1,1))
> fit

Series: Nile
ARIMA(0,1,1)

Coefficients:
          ma1
      -0.7329
s.e.   0.1143

sigma^2 estimated as 20600:  log likelihood=-632.55
AIC=1269.09   AICc=1269.22   BIC=1274.28

> accuracy(fit)

                ME  RMSE   MAE    MPE  MAPE   MASE
Training set -11.94 142.8 112.2 -3.575 12.94 0.8089
```

Обратите внимание, что модель применяется к исходному временному ряду. С параметром `d = 1` функция вычислит первые разности. Коэффициент для метода скользящих средних (–0,73) предоставляется вместе со значением информационного критерия Акаике (Akaike Information Criterion, AIC; раздел 8.6.1). Если вам потребуется попробовать подобрать другие модели, то значение AIC может помочь вам выбрать наиболее подходящую. Считается, что чем меньше значение AIC, тем лучше модель. Меры точности могут помочь определить, обладает ли полученная модель достаточной точностью. Здесь средняя абсолютная процентная ошибка составляет 13 %.

Оценка адекватности модели

Адекватная модель должна давать остатки с нормальным распределением и средним значением, равным нулю, а автокорреляции должны быть равны нулю для каждого возможного лага. Другими словами, остатки должны быть распределены нормально и независимо. Достоверность этих предположений можно оценить с помощью кода в листинге 15.11.

Листинг 15.11. Оценка адекватности модели

```
> library(ggplot2)
> df <- data.frame(resid = as.numeric(fit$residuals))     1
> ggplot(df, aes(sample = resid)) +                       2
      stat_qq() + stat_qq_line() +
      labs(title="Normal Q-Q Plot")
```

```
> Box.test(fit$residuals, type="Ljung-Box")                    ❸
    Box-Ljung test

data:  fit$residuals
X-squared = 1.3711, df = 1, p-value = 0.2416
```

❶ **Извлечь остатки.**

❷ **Создать Q-Q-диаграмму.**

❸ **Проверить равенство нулю автокорреляций при всех лагах.**

Первым делом из объекта `fit` извлекаются остатки и сохраняются в таблице данных. Затем с помощью функций `qq_*` строится Q-Q-диаграмма (рис. 15.15). Нормально распределенные данные должны располагаться вдоль линии. В данном случае результаты выглядят хорошо.

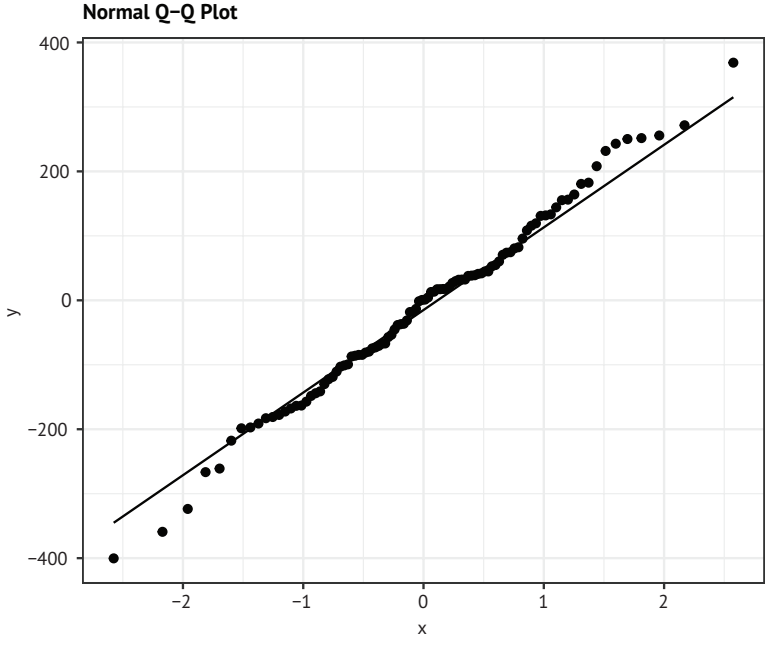


Рис. 15.15. Q-Q-диаграмма для определения нормальности распределения остатков модели. Нормально распределенные данные должны располагаться вдоль линии

Функция `Box.test()` проверяет равенство всех автокорреляций нулю. Считается, что результаты не значимы, если автокорреляции не отличаются от нуля. Эта модель ARIMA хорошо согласуется с данными.

Прогнозирование

Если бы модель не удовлетворяла предположениям о нормальном распределении остатков и нулевых автокорреляциях, потребова-

лось бы изменить модель, добавить параметры или попробовать другой подход. После выбора окончательной модели ее можно использовать для прогнозирования будущих значений. В листинге 15.12 показано применение функции `forecast()` из пакета `forecast` для прогнозирования на три года вперед.

Листинг 15.12. Прогнозирование с помощью модели ARIMA

```
> forecast(fit, 3)

     Point Forecast    Lo 80     Hi 80    Lo 95    Hi 95
1971       798.3673 614.4307  982.3040 517.0605 1079.674
1972       798.3673 607.9845  988.7502 507.2019 1089.533
1973       798.3673 601.7495  994.9851 497.6663 1099.068

> autoplot(forecast(fit, 3)) + labs(x="Year", y="Annual Flow")
```

Функция `autoplot()` используется здесь для построения графика прогноза (рис. 15.16). Сами оценки представлены черной линией, а границы достоверности 80 % и 95 % – темно-синими и светло-синими линиями соответственно.

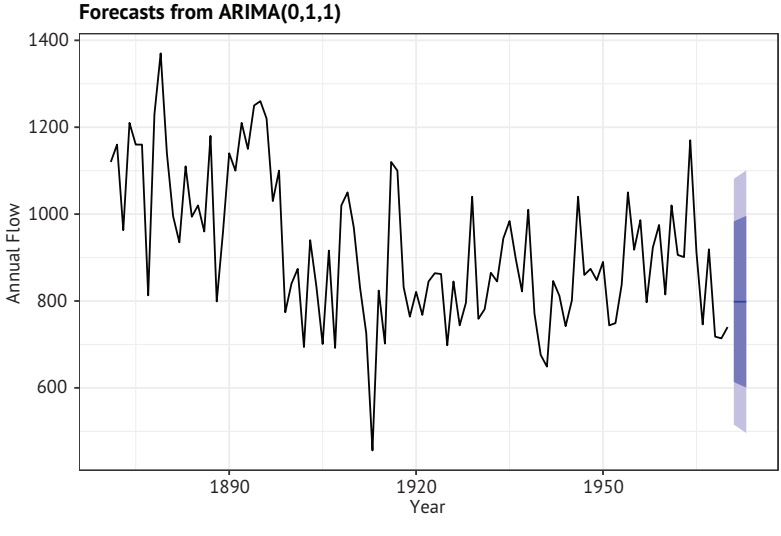


Рис. 15.16. Трехлетний прогноз для временного ряда Nile с использованием подобранной модели ARIMA(0,1,1). Черная линия представляет прогнозные оценки, а светло- и темно-синие линии – пределы доверительных интервалов 80 % и 95 % соответственно

В разделе 15.2.3 мы использовали функцию `ets()` из пакета `forecast` для автоматизации выбора наилучшей экспоненциальной модели. Пакет также предлагает функцию `auto.arima()` для выбора лучшей модели ARIMA. Листинг 15.13 демонстрирует применение этого подхода к временному ряду, описывающему изменение количества солнечных пятен.

Листинг 15.13. Автоматический выбор модели прогнозирования ARIMA

```
> library(forecast)
> fit <- auto.arima(sunspots)
> fit
Series: sunspots
ARIMA(2,1,2)
Coefficients:
       ar1     ar2    ma1    ma2
      1.35  -0.396  -1.77  0.810
s.e.  0.03   0.029   0.02  0.019

sigma^2 estimated as 243:  log likelihood=-11746
AIC=23501   AICc=23501   BIC=23531

> forecast(fit, 3)


         Point Forecast       Lo 80    Hi 80      Lo 95    Hi 95
Jan 1984      40.437722  20.4412613 60.43418   9.855774 71.01967
Feb 1984      41.352897  18.2795867 64.42621   6.065314 76.64048
Mar 1984      39.796425  15.2537785 64.33907   2.261686 77.33116

> accuracy(fit)
                    ME RMSE   MAE MPE MAPE MASE
Training set -0.02673 15.6 11.03 NaN  Inf 0.32
```

В этом примере функция выбрала модель ARIMA с параметрами $p = 2$, $d = 1$ и $q = 2$. Эти значения минимизируют критерий AIC по большому количеству возможных моделей. Точность MPE и MAPE резко возрастает из-за нулевых значений в ряду (что является недостатком этих двух статистик). Вывод результатов и оценку адекватности я оставляю вам в качестве упражнения.

Будьте осторожны с прогнозированием

Прогнозирование имеет долгую и разнообразную историю: от шаманов в первобытные времена, предсказывающих погоду, до современных специалистов по данным, прогнозирующих результаты выборов. Возможность предсказания имеет фундаментальное значение как для науки, так и для человеческой природы. Описанные методологии могут иметь решающее значение для понимания и прогнозирования самых разных явлений, но важно помнить, что каждая из них предполагает экстраполяцию – выход за пределы имеющихся данных. Они предполагают, что в будущем сохранятся текущие условия и тенденции. Финансовые прогнозы, сделанные в 2007 году, предполагали продолжение экономического роста в 2008 году. Но, как мы теперь знаем, все обернулось иначе. Важные события могут изменить тенденцию и структуру временного ряда, и чем более отдаленное будущее вы пытаетесь предсказать, тем больше неопределенность.

15.5. Дополнительная информация

Существует много хороших книг, посвященных анализу временных рядов и прогнозированию. Например, «Forecasting: Principles and Practice» (http://otexts.com/fpp2, 2018 г.) – простое и лаконичное онлайн-руководство Роба Хайндмана (Rob Hyndman) и Джорджа Атанасопулоса (George Athanasopoulos) с примерами на R. Я настоятельно рекомендую прочитать его. Кроме того, Cowpertwait и Metcalfe (2009) написали отличную книгу по анализу временных рядов с помощью R. Еще один продвинутый подход, тоже включающий код на R, можно найти в публикации Shumway & Stoffer (2010). Наконец, вы можете обратиться к обзору задач по анализу временных рядов «CRAN Task View: Time Series Analysis» (http://cran.r-project.org/web/views/TimeSeries.html), который содержит исчерпывающую сводку всех возможностей R в отношении временных рядов.

Итоги

- R предоставляет обширный набор структур данных для хранения временных рядов. В базовой версии R имеются классы для хранения одной (`ts`) или нескольких (`mts`) серий наблюдений, записанных через равные промежутки времени. Пакеты `xts` и `zoo` расширяют эти классы, добавляя поддержку последовательностей наблюдений, сделанных через нерегулярные интервалы.
- Временные ряды, хранящиеся в виде объектов `xts`, легко разделить на подмножества с использованием нотации квадратных скобок `[]` и агрегировать с применением функций `apply.period`.
- Пакет `forecast` предлагает несколько функций для визуального изучения временных рядов. Функцию `autoplot()` можно использовать для отображения временных рядов в виде линейных графиков `ggplot2`. Функция `ma()` позволяет сглаживать неравномерности во временном ряду, чтобы выделить тренды.
- С помощью функции `stl()` можно разложить временной ряд на трендовые, сезонные и нерегулярные (остаточные) составляющие.
- Пакет `forecast` также можно использовать для прогнозирования будущих значений по временным рядам. В этой главе мы рассмотрели два популярных подхода к прогнозированию: экспоненциальные модели и авторегрессионные интегрированные модели скользящих средних (ARIMA).

16 Кластерный анализ

В этой главе:

- выявление связных подгрупп (кластеров) наблюдений;
- определение количества имеющихся кластеров;
- получение вложенной иерархии кластеров;
- получение дискретных кластеров.

Кластерный анализ – это метод сокращения объема данных, предназначенный для выявления подгрупп взаимосвязанных наблюдений. Он позволяет сократить большое количество наблюдений до гораздо меньшего количества кластеров или типов. *Кластер* – это группа наблюдений, больше похожих друг на друга, чем на наблюдения в других группах. Это определение допускает разные толкования, вследствие чего появилось огромное разнообразие методов кластеризации.

Кластерный анализ широко используется в биологических науках и в бихевиоризме (науке о поведении), в маркетинге и медицинских исследованиях. Например, исследователь-психолог может сгруппировать данные о симптомах и демографических характеристиках пациентов с депрессией, стремясь выявить подтипы депрессии в надежде, что выявление таких подтипов может способствовать более целенаправленному и эффективному лечению и лучшему пониманию расстройства. Маркетологи используют кластерный анализ

в качестве стратегии сегментации клиентов. Клиенты группируются по схожести их демографических данных и покупательского поведения. Затем маркетинговые кампании адаптируются для воздействия на одну или несколько выявленных подгрупп. Исследователи в области медицины используют кластерный анализ для классификации проявлений генов, полученных из данных ДНК-микрочипов. Это может помочь им понять нормальный рост и развитие, а также основные причины многих болезней человека.

Двумя наиболее популярными подходами к кластеризации являются *иерархическая агломеративная кластеризация* и *разделяющая кластеризация*. При использовании иерерхической агломеративной кластеризации каждое наблюдение первоначально рассматривается как отдельный кластер. Затем кластеры объединяются попарно, пока все они не будут объединены в один кластер. При использовании разделяющей кластеризации заранее указывается количество искомых кластеров *K*. Затем наблюдения случайным образом делятся на *K* групп и перетасовываются, пока не будут сформированы связные кластеры.

В рамках каждого из этих двух широких подходов существует множество алгоритмов кластеризации. Для иерархической кластеризации чаще используются методы одиночной связи, полной связи, средней связи, центроидов и Уорда. Для разделяющей кластеризации чаще других применяются методы *k*-средних и разделения вокруг медоидов (Partitioning Around Medoids, PAM). Все методы кластеризации имеют свои преимущества и недостатки, которые мы обсудим далее.

Примеры в этой главе сосредоточены на еде и вине (подозреваю, что мои друзья ничуть не удивлены таким выбором). Мы применим иерархическую кластеризацию к набору данных о `nutrient`, входящему в пакет `flexclust`, чтобы ответить на следующие вопросы:

- В чем сходства и отличия 27 видов рыб, птиц и мяса, для которых измерены содержание 5 питательных веществ?
- Можно ли более или менее разумно сгруппировать эти продукты в меньшее количество групп?

Разделяющую кластеризацию мы используем для оценки 178 образцов итальянских вин по результатам химического анализа по 13 показателям. Данные содержатся в наборе `wine`, доступном в пакете `rattle`. А вот вопросы, на которые мы попробуем ответить:

- Можно ли разделить вина на ярко выраженные подтипы?
- Если да, то сколько таких подтипов имеется и каковы их характеристики?

На самом деле образцы вин представлены тремя сортами (определяются значением переменной `Type`). Ориентируясь на эту переменную, мы сможем оценить, насколько хорошо кластерный анализ восстанавливает базовую структуру.

Несмотря на многообразие подходов к кластерному анализу, все они основаны на одной и той же последовательности шагов, описанной в разделе 16.1. Иерархическая агломеративная кластеризация описана в разделе 16.3, а методы разделяющей кластеризации – в разделе 16.4. В разделе 16.6 будут даны некоторые заключительные советы и рекомендации. Для опробования примеров из этой главы обязательно установите пакеты `cluster`, `NbClust`, `flexclust`, `fMultivar`, `ggplot2`, `ggdendro`, `factoextra`, `clusterability` и `rattle`. Пакет `rattle` также будет использоваться в главе 17.

16.1. Общие этапы кластерного анализа

Подобно факторному анализу (глава 14), эффективный кластерный анализ – это многоэтапный процесс, включающий множество точек принятия решений. Каждое решение может повлиять на качество и полезность результатов. В этом разделе я опишу 11 основных этапов кластерного анализа.

Выбор подходящих атрибутов. Первый (и, возможно, самый важный) шаг – выбор переменных, которые могут играть важную роль в выявлении и понимании различий между группами наблюдений в данных. Например, при исследовании депрессии может понадобиться оценить один или несколько из следующих признаков: психологические симптомы; физические симптомы; возраст, когда возникло заболевание; количество, продолжительность и время депрессивных периодов; количество госпитализаций; функциональное состояние в смысле способности к самообслуживанию; социальная и трудовая история; текущий возраст; пол; этническая принадлежность; социоэкономический статус; семейный статус; семейный анамнез; результат предыдущего лечения. Сложность кластерного анализа не может компенсировать плохой выбор переменных.

Масштабирование данных. Если значения переменных, участвующих в анализе, сильно отличаются диапазонами значений, то переменные с наибольшим диапазоном будут оказывать большее влияние на результаты. Часто это нежелательно, поэтому аналитики обычно масштабируют данные. Наиболее популярный способ заключается в стандартизации каждой переменной, чтобы привести к среднему, равному нулю, и стандартному отклонению, равному единице. Другие альтернативы: деление каждой переменной на ее максимальное значение или вычитание среднего значения переменной и деление на среднее абсолютное отклонение этой переменной. Эти три подхода показаны в следующем фрагменте кода:

```
df1 <- apply(mydata, 2, function(x){(x-mean(x))/sd(x)})
df2 <- apply(mydata, 2, function(x){x/max(x)})
df3 <- apply(mydata, 2, function(x){(x - mean(x))/mad(x)})
```

В этой главе мы используем функцию scale() и применим подход на основе стандартизации переменных до среднего значения, равного нулю, и стандартного отклонения, равного единице. Эту задачу решает первая инструкция в предыдущем фрагменте (df1).

Анализ выбросов. Многие методы кластеризации чувствительны к выбросам, которые могут искажать решения. По этой причине часто бывает желательно выявить (и удалить) одномерные выбросы, используя функции из пакета outliers. Для идентификации многомерных выбросов можно использовать функции из пакета mvoutlier. Альтернативный подход – использовать метод кластеризации, устойчивый к выбросам. Одним из таких устойчивых методов является метод разделения вокруг медоидов (раздел 16.4.2).

Выбор меры расстояния. Несмотря на существенные различия между алгоритмами кластеризации, практически все они требуют наличия меры расстояния между кластеризуемыми объектами. Наиболее популярной мерой расстояния между двумя наблюдениями является евклидово расстояние, но также можно использовать манхэттенское, канберрское, асимметричное двоичное, максимальное расстояния и расстояния Минковского (подробности ищите в ?dist). В этой главе везде используется евклидово расстояние. Вычисление евклидовых расстояний описано в разделе 16.2.

Выбор алгоритма кластеризации. На следующем шаге выбирается метод кластеризации. Иерархическая кластеризация удобна для решения небольших задач (скажем, 150 наблюдений или меньше) и когда желательна вложенная иерархия группировок. Метод разделяющей кластеризации способен справляться с гораздо более объемными наборами данных, но требует заранее указывать количество кластеров. После выбора метода кластеризации необходимо выбрать конкретный алгоритм. И снова каждый имеет свои преимущества и недостатки. Наиболее популярные алгоритмы описаны в разделах 16.3 и 16.4. При желании можно попробовать несколько алгоритмов, чтобы выяснить, насколько их результаты согласуются друг с другом.

Получение одного или нескольких кластерных решений. На этом шаге применяются методы и алгоритмы, выбранные на шаге 5.

Определение количества имеющихся кластеров. Чтобы получить окончательное решение, необходимо определить, на сколько кластеров можно разделить наблюдения. Это сложная задача, для решения которой было предложено множество подходов. Обычно для ее решения предлагается извлечь разное количество кластеров (скажем, от 2 до *K*) и сравнить адекватность решений. Функция NbClust() в пакете NbClust предлагает опробование 26 различных критериев, которые помогут вам принять это решение (что элегантно демонстрирует, насколько сложной является эта задача). Пакет NbClust будет использоваться на протяжении всей этой главы.

Получение окончательного решения кластеризации. После определения количества кластеров выполняется окончательная кластеризация для извлечения этого количества подгрупп.

Визуализация результатов. Визуализация может помочь определить значимость и полезность кластерного решения. Результаты иерархической кластеризации обычно представляются в виде дендрограмм. Результаты разделяющей кластеризации – в виде двумерной кластерной диаграммы.

Интерпретация кластеров. После получения кластерного решения его необходимо интерпретировать (и, возможно, назвать физический смысл кластеров). Что общего между наблюдениями в одном кластере? Чем они отличаются от наблюдений в других кластерах? Этот шаг обычно выполняется путем получения сводной статистики для каждой переменной по кластеру. Для непрерывных данных вычисляются средние значения или медианы всех переменных в каждом кластере. Для смешанных данных (содержащих категориальные переменные) сводная статистика также будет включать типы или распределения категорий.

Оценка результатов. Оценка кластерного решения включает ответ на вопрос: «Насколько эти кластеры отражают реальность и не являются ли они проявлением уникальных особенностей этого набора данных или статистического метода?» Если использовать другой метод кластеризации или другую выборку, будут ли получены такие же кластеры? Пакеты `fpc`, `clv` и `clValid` содержат функции для оценки стабильности кластерного решения.

Поскольку вычисление расстояний между наблюдениями является неотъемлемой частью кластерного анализа, остановимся на нем более подробно.

16.2. Вычисление расстояний

Каждый кластерный анализ начинается с вычисления расстояния, определяющего непохожесть или близость кластеризуемых объектов. Евклидово расстояние между двумя наблюдениями определяется формулой:

$$d_{ij} = \sqrt{\sum_{p=1}^{P}\left(x_{ip} - x_{jp}\right)^2},$$

где i и j – наблюдения, а P – количество переменных. Другими словами, евклидово расстояние между двумя наблюдениями равно квадратному корню из суммы квадратов разностей значений каждой переменной.

Рассмотрим набор данных `nutrient`, поставляемый с пакетом `flexclust`. Набор данных содержит измерения содержания питательных веществ в 27 видах мяса, рыбы и птицы. Вот первые несколько наблюдений:

```
> data(nutrient, package="flexclust")
> head(nutrient, 4)

             energy protein fat calcium iron
BEEF BRAISED    340      20  28       9  2.6
HAMBURGER       245      21  17       9  2.7
BEEF ROAST      420      15  39       7  2.0
BEEF STEAK      375      19  32       9  2.6
```

и евклидово расстояние между первыми двумя наблюдениями (BEEF BRAISED [тушеная говядина] и HAMBURGER [гамбургер]) равно:

$$d = \sqrt{(340-245)^2 + (20-21)^2 + (28-17)^2 + (9-9)^2 + (26-27)^2} = 95.64.$$

Для вычисления расстояний между всеми строками (наблюдениями) в матрице или таблице данных можно использовать стандартную функцию `dist()`, имеющую следующий синтаксис: `dist(`*x*`, method=)`, где *x* – входные данные, а параметр `method=` по умолчанию принимает значение `"euclidean"`. Функция возвращает нижнюю треугольную матрицу, но вообще можно использовать функцию `as.matrix()`, чтобы получить возможность доступа к расстояниям с применением стандартной нотации с квадратными скобками. Для таблицы данных `nutrient`:

```
> d <- dist(nutrient)
> as.matrix(d)[1:4,1:4]

             BEEF BRAISED HAMBURGER BEEF ROAST BEEF STEAK
BEEF BRAISED          0.0      95.6        80.9        35.2
HAMBURGER            95.6       0.0       176.5       130.9
BEEF ROAST           80.9     176.5         0.0        45.8
BEEF STEAK           35.2     130.9        45.8         0.0
```

Большие расстояния указывают на большие различия между наблюдениями. Расстояние между наблюдением и самим собой равно 0. Как и ожидалось, функция `dist()` вернула то же расстояние между тушеной говядиной и гамбургером, что мы получили вручную.

Кластерный анализ со смешанными типами данных

Евклидовы расстояния обычно являются предпочтительной мерой для непрерывных данных. Но если присутствуют другие типы переменных, необходимо использовать альтернативные меры несходства. Например, чтобы получить матрицу различий наблюдений, имеющих любую комбинацию двоичных, номинальных, порядковых и непрерывных атрибутов, можно использовать функцию `daisy()` из пакета `cluster`. Другие функции в пакете `cluster` могут использовать эти меры различия для кластерного анализа. Например, `agnes()` выполняет иерархическую агломеративную кластеризацию, а `pam()` – кластеризацию разделением вокруг медоидов.

Обратите внимание, что расстояния в таблице данных `nutrient` сильно зависят от вклада переменной `energy` (энергетическая ценность), которая имеет гораздо больший диапазон значений. Масштабирование данных поможет уравнять влияние переменных. В следующем разделе мы выполним иерархический кластерный анализ для этого набора данных.

16.3. Иерархический кластерный анализ

Как отмечалось выше, метод иерархической агломеративной кластеризации начинается с того, что рассматривает каждое наблюдение как отдельный кластер. Затем кластеры попарно объединяются, пока все они не объединятся в один кластер. Описать алгоритм в общих чертах можно так:

1. Определить каждое наблюдение (строку, случай) как кластер.
2. Вычислить расстояния между каждым кластером и всеми другими кластерами.
3. Объединить два кластера с наименьшим расстоянием. Это уменьшит количество кластеров на один.
4. Повторять шаги 2 и 3, пока все кластеры не будут объединены в один кластер, содержащий все наблюдения.

Основное различие между алгоритмами иерархической кластеризации заключается в том, как они определяют расстояния между кластерами (шаг 2). В табл. 16.1 перечислены пять наиболее распространенных иерархических алгоритмов кластеризации и описано, как они определяют расстояния между двумя кластерами.

Таблица 16.1. Алгоритмы иерархической кластеризации

Алгоритм кластеризации	Определение расстояния между двумя кластерами
Одиночной связи	Наименьшее расстояние между точками в двух кластерах
Полной связи	Наибольшее расстояние между точками в двух кластерах
Средней связи	Среднее расстояние между точками в двух кластерах (также называется усреднением невзвешенного парного группового среднего [Unweighted Pair Group Mean Averaging, UPGMA])
Центроидов	Расстояние между центроидами (векторами средних значений переменных) двух кластеров. Для одного наблюдения центроид – это значения переменной
Уорда	Сумма квадратов ANOVA между двумя кластерами по всем переменным

Алгоритм одиночной связи имеет тенденцию обнаруживать удлиненные сигарообразные кластеры и часто демонстрирует явление, называемое *цепочкой*, когда разнородные наблюдения объеди-

няются в один кластер, потому что они похожи на промежуточные наблюдения между ними. Алгоритм полной связи имеет тенденцию находить компактные кластеры примерно одинакового диаметра. Он также чувствителен к выбросам. Алгоритм средней связи предлагает компромисс между ними. Он менее склонен к созданию цепочек и более устойчив к выбросам, а также имеет тенденцию объединять кластеры с небольшими отклонениями.

Алгоритм Уорда обычно объединяет кластеры с небольшим и примерно равным количеством наблюдений. Он также чувствителен к выбросам. Алгоритм центроидов предлагает привлекательную альтернативу благодаря простому и понятному определению меры расстояний. Он менее чувствителен к выбросам, чем другие иерархические методы, но может давать не такие хорошие результаты, как метод средней связи или метод Уорда.

Иерархическую кластеризацию можно выполнить с помощью функции `hclust()`, имеющей следующий синтаксис: `hclust(d, method=)`, где `d` – матрица расстояний, созданная функцией `dist()`, а в параметре `method=` можно передать значение `"single"`, `"complete"`, `"average"`, `"centroid"` или `"ward"`.

В этом разделе мы применим кластеризацию методом средней связи (`method="average"`) к набору данных `nutrient`, представленному в разделе 16.2, чтобы определить сходства, различия и однородные группы среди 27 типов продуктов питания. В листинге 16.1 показано, как это сделать.

Листинг 16.1. Кластеризация средней связи данных из набора nutrient

```
data(nutrient, package="flexclust")
row.names(nutrient) <- tolower(row.names(nutrient))
nutrient.scaled <- scale(nutrient)

d <- dist(nutrient.scaled)

fit.average <- hclust(d, method="average")

library(ggplot2)
library(ggdendro)
ggdedgrogram(fit.average) + labs(title="Average Linkage Clustering")
```

Этот код сначала импортирует данные и преобразует имена строк в нижний регистр (потому что я ненавижу МЕТКИ В ВЕРХНЕМ РЕГИСТРЕ). Поскольку переменные сильно различаются диапазонами значений, они стандартизируются и приводятся к среднему значению 0 и стандартному отклонению 1. Далее вычисляются евклидовы расстояния между всеми 27 типами продуктов питания и выполняется кластеризация по средней связи. Наконец, результаты отображаются в виде дендрограммы с помощью пакетов `ggplot2` и `ggdendro` (рис. 16.1).

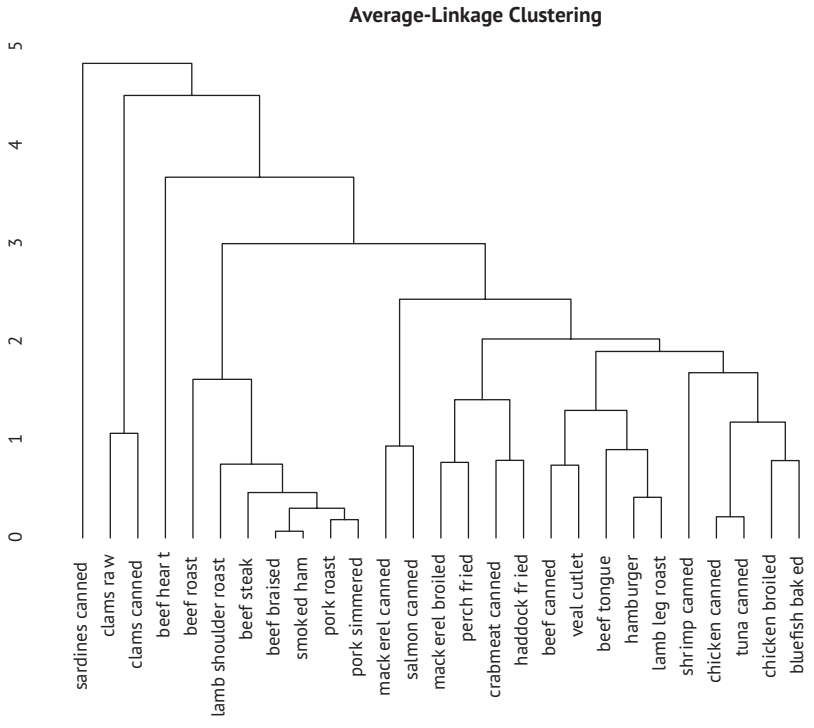


Рис. 16.1. Кластеризация набора данных nutrient методом средней связи

Дендрограмма показывает, как элементы объединяются в кластеры, и читать ее следует снизу вверх. Каждое наблюдение изначально считается отдельным кластером. Затем два наиболее близких наблюдения (beef braised [тушеная говядина] и smoked ham [копченая ветчина]) объединяются. Далее объединяются жареная свинина (pork roast) и тушеная свинина (pork simmered), потом консервированная курица (chicken canned) и консервированный тунец (tuna canned). На четвертом этапе объединяются кластеры тушеная говядина (beef braised) / копченая ветчина (smoked ham) и жареная свинина (pork roast) / тушеная свинина (pork simmered) – вновь образованный кластер содержит четыре продукта, и т. д., пока все наблюдения не будут объединены в один кластер. Высота кластера отражает уровень критерия, на котором произошло объединение. Для кластеризации методом средней связи критерием является среднее расстояние между точками в двух кластерах.

Если ваша цель состоит в том, чтобы выяснить сходства и различия продуктов питания с точки зрения содержащихся в них питательных веществ, то дендрограммы на рис. 16.1 может быть достаточно. Она дает иерархическое представление о сходстве/различии между 27 элементами. Консервированные тунец (canned tuna) и курица (canned chicken) похожи друг на друга и сильно отличают-

ся от консервированных моллюсков (canned clams). Но если цель состоит в том, чтобы сформировать меньшее количество (желательно значимых) групп, то потребуется дополнительный анализ для выбора соответствующего количества кластеров.

Пакет `NbClust` предлагает множество метрик для определения оптимального количества кластеров. Конечно, нет никакой гарантии, что все они дадут одинаковый результат. Более того, их результаты почти наверняка будут немного отличаться. И тем не менее эти результаты можно использовать в качестве руководства для выбора подходящего значения K – количества кластеров. Функция `NbClust()` принимает матрицу или таблицу данных, используемую меру расстояния, метод кластеризации, а также минимальное и максимальное количества рассматриваемых кластеров и возвращает метрики вместе с соответствующими количествами кластеров. Листинг 16.2 иллюстрирует этот подход с применением метода кластеризации средней связи к набору данных `nutrient`.

Листинг 16.2. Выбор количества кластеров

```
> library(NbClust)
> library(factoextra)
> nc <- NbClust(nutrient.scaled, distance="euclidean",
                min.nc=2, max.nc=15, method="average")
> fviz_nbclust(nc)
```

Здесь два критерия отдают предпочтение нулевому числу кластеров, один критерий – одному кластеру, четыре критерия – двум кластерам и т. д. Результаты можно отобразить в виде диаграммы с помощью функции `fviz_nbclust()` (рис. 16.2). Оптимальным можно считать количество кластеров, получившее больше всего голосов. В случае ничьей предпочтение обычно отдается решению с меньшим количеством кластеров.

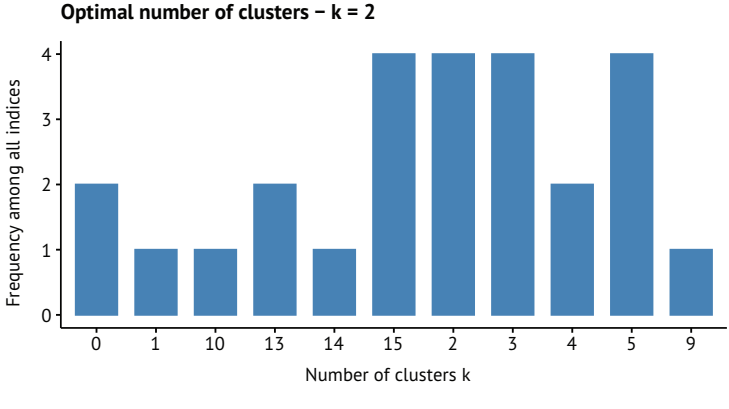


Рис. 16.2. Количество кластеров, рекомендуемое 26 критериями из пакета NbClust

Диаграмма предлагает считать оптимальным два кластера, но также можно попробовать решения с 3, 5 и 15 кластерами и выбрать то, которое будет иметь более удачную интерпретацию. В листинге 16.3 демонстрируется решение с 5 кластерами.

Листинг 16.3. Получение окончательного решения кластеризации

```
> clusters <- cutree(fit.average, k=5)                                    (1)
> table(clusters)

clusters
 1  2  3  4  5
 7 16  1  2  1

> nutrient.scaled$clusters <- clusters

> library(dplyr)
> profiles <- nutrient.scaled %>%                                         (2)
     group_by(clusters) %>%                                               (2)
     summarize_all(median)                                                (2)

> profiles %>% round(3) %>% data.frame()

  cluster energy protein    fat calcium    iron
1       1  1.310   0.000  1.379  -0.448  0.0811
2       2 -0.370   0.235 -0.487  -0.397 -0.6374
3       3 -0.468   1.646 -0.753  -0.384  2.4078
4       4 -1.481  -2.352 -1.109   0.436  2.2709
5       5 -0.271   0.706 -0.398   4.140  0.0811

> library(colorhcplot)                                                    (3)
> cl <-factor(clusters, levels=c(1:5),                                    (3)
              labels=paste("cluster", 1:5))                              (3)
> colorhcplot(fit.average, cl, hang=-1, lab.cex=.8, lwd=2,                (3)
            main="Average-Linkage Clustering\n5 Cluster Solution")       (3)
```

(1) Распределение данных по заданному числу кластеров.

(2) Получение обобщенных статистик.

(3) Вывод результатов в виде диаграммы.

Функция `cutree()` «режет» дерево на 5 кластеров (1). В первый кластер попадают 7 наблюдений, во второй – 16 наблюдений и т. д. Затем применяются функции из пакета `dplyr` для получения медианного профиля каждого кластера (2). Наконец, выводится окончательная дендрограмма (рис. 16.3) и используется функция `colorhcplot()` для идентификации 5 кластеров (3). Здесь аргумент `cl` – это фактор с метками кластеров, `hang=-1` выравнивает метки по нижней границе графика, `lab.cex` управляет размером меток (здесь используется значение по умолчанию 8 %), а `lwd` управляет толщиной линий дендрограммы. Если вы читаете печатную версию этой книги, то обязатель-

но запустите код в листинге 16.3, потому что в черно-белой печати трудно заметить различия, которые хорошо видны в цвете.

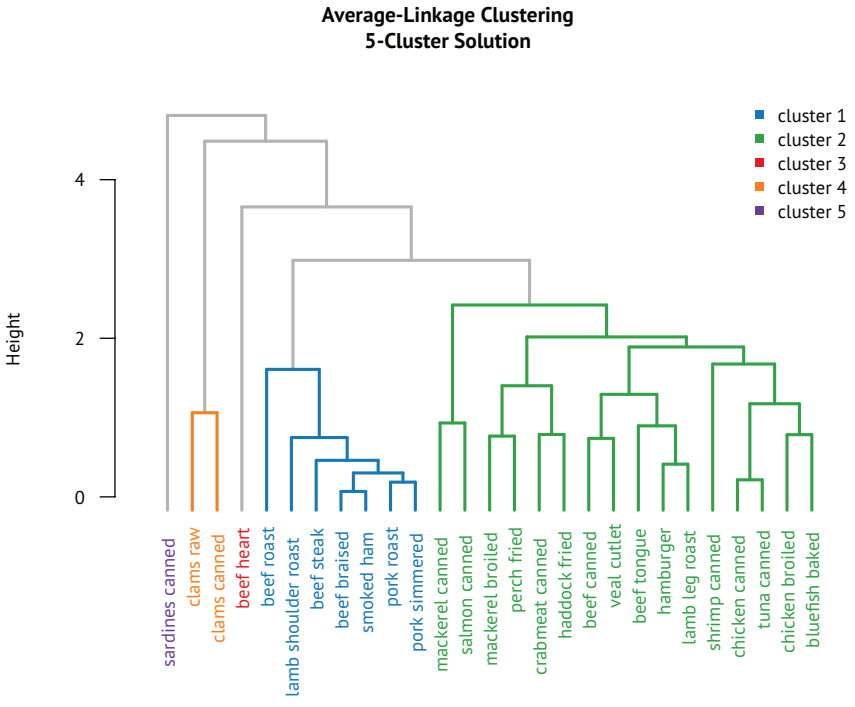


Рис. 16.3. Результат кластеризации набора данных nutrient методом средней связи на 5 кластеров

Сардины (sardines canned) были выделены в отдельный кластер и содержат гораздо больше кальция, чем другие группы продуктов. Говяжье сердце (beef heart) тоже оказалось в отдельном кластере, как продукт, богатый белком и железом. Кластер моллюсков (clams) отличается низким содержанием белка и высоким содержанием железа. Продукты в кластере с жареной говядиной (beef roast) и тушеной свининой (pork simmered) содержат много жира и имеют высокую энергетическую ценность. Наконец, продукты в самой большой группе (от скумбрии [mackerel] до луфаря [bluefish]) содержат относительно мало железа.

Иерархическая кластеризация может быть особенно полезной, когда ожидаются наличие вложенных кластеров и осмысленная иерархия. Это часто имеет место в биологических науках. Но иерархические алгоритмы – жадные в том смысле, что после закрепления наблюдения за некоторым кластером оно не может перекочевать в другой кластер позже. Кроме того, иерархическую кластеризацию трудно применять к большим выборкам, включающим сотни или даже тысячи наблюдений. В таких случаях прибегают к разделяющим методам.

16.4. Разделяющие методы кластерного анализа

В разделяющих методах наблюдения делятся на K групп и перетасовываются, чтобы сформировать наиболее связные кластеры в соответствии с заданным критерием. В этом разделе мы рассмотрим два метода: метод k-средних и разделение вокруг медоидов (Partitioning Around Medoids, PAM).

16.4.1. Кластеризация методом k-средних

Наиболее распространенным методом разделяющей кластеризации является метод k-средних. Концептуально алгоритм k-средних выглядит следующим образом:

1. Выбрать K центроидов (выбираются K случайных строк).
2. Связать каждую точку данных с ближайшим к ней центроидом.
3. Пересчитайте центроиды как среднее значение всех точек, оказавшихся в кластере (т. е. центроиды теперь будут представлять собой средние векторы с длиной p, где p – количество переменных).
4. Связать каждую точку данных с ближайшим к ней центроидом.
5. Продолжать выполнять шаги 3 и 4, пока не будет достигнута итерация, в которой ни одна точка не сменила кластер, или не будет превышено максимальное количество итераций (в R по умолчанию выполнятся не более 10 итераций).

Детали реализации этого подхода могут различаться.

В R используется эффективный алгоритм Хартигана и Вонга (Hartigan & Wong [1979]), который разбивает наблюдения на k групп так, чтобы сумма квадратов расстояний между наблюдениями и назначенными им центроидами была минимальной. Это означает, что на шагах 2 и 4 каждое наблюдение связывается с ближайшим кластером.

$$ss(k) = \sum_{i=1}^{n} \sum_{j=0}^{p} \left(x_{ij} - \bar{x}_{kj}\right)^2,$$

где k – кластер, x_{ij} – значение j-й переменной в i-м наблюдении, $\bar{x}_{kj}$ – среднее значение j-й переменной для k-го кластера, а p – количество переменных.

Кластеризация методом k-средних может обрабатывать бóльшие наборы данных, чем методы иерархической кластеризации. Кроме того, наблюдения не фиксируются в кластере навсегда – они могут перемещаться между кластерами, если это улучшает общее решение. Но использование средних значений подразумевает, что все переменные должны быть непрерывными, а выбросы могут сильно повлиять на результат. Кроме того, этот метод плохо работает при наличии невыпуклых (например, U-образных) кластеров.

Синтаксис функции, реализующей метод *k*-средних в R: `kmeans(x, center)`, где *x* – числовой набор данных (матрица или таблица данных), а *center* – количество извлекаемых кластеров. Функция возвращает состав кластеров, их центроиды, суммы квадратов расстояний (внутри, между, всего) и размеры кластеров.

Поскольку кластерный анализ методом *k*-средних начинается с *k* случайно выбранных центроидов, при каждом последующем вызове функции могут возвращаться разные решения. Чтобы гарантировать воспроизводимость результатов, можно использовать функцию `set.seed()`. Кроме того, этот метод кластеризации может быть чувствителен к первоначальному выбору центроидов. Функция `kmeans()` имеет также параметр `nstart`, с помощью которого можно задать число попыток кластеризации с разными начальными конфигурациями, и в этом случае функция вернет наилучший результат. Например, получив параметр `nstart=25`, функция выполнит 25 попыток. Этот подход часто рекомендуется.

В отличие от иерархической кластеризации, кластеризация методом *k*-средних требует заранее указать количество кластеров. Для этого снова можно использовать пакет `NbClust`. Кроме того, может оказаться полезным вывести график зависимости сумм квадратов расстояний внутри групп от количества кластеров в решении методом *k*-средних. Изгиб на графике (аналогичный изгибу на графике критерия собственных значений, описанном в разделе 14.2.1) может указывать на соответствующее количество кластеров.

График можно построить с помощью следующей функции:

```
wssplot <- function(data, nc=15, seed=1234){
  require(ggplot2)
  wss <- numeric(nc)
  for (i in 1:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)
  }
  results <- data.frame(cluster=1:nc, wss=wss)
  ggplot(results, aes(x=cluster,y=wss)) +
    geom_point(color="steelblue", size=2) +
    geom_line(color="grey") +
    theme_bw() +
    labs(x="Number of Clusters",
         y="Within groups sum of squares")
}
```

Параметр `data` – это набор числовых данных для анализа, `nc` – максимальное количество рассматриваемых кластеров, а `seed` – начальное случайное число.

Давайте применим кластеризацию методом *k*-средних к набору данных, содержащему результаты химического анализа 178 образцов итальянских вин по 13 показателям. Данные взяты из репози-

тория машинного обучения UCI (http://www.ics.uci.edu/~mlearn/MLRepository.html) и доступны в составе пакета `rattle`. В этом наборе данных наблюдения представляют три сорта вин, на что указывает первая переменная (`Type`). Отбросим эту переменную, выполним кластерный анализ и посмотрим, сможет ли он восстановить известную структуру (листинг 16.4).

Листинг 16.4. Кластеризация вин методом *k*-средних

```
> data(wine, package="rattle")
> library(NbClust)
> library(factoextra)
> head(wine)

  Type Alcohol Malic  Ash Alcalinity Magnesium Phenols Flavanoids
1    1   14.23  1.71 2.43       15.6       127    2.80       3.06
2    1   13.20  1.78 2.14       11.2       100    2.65       2.76
3    1   13.16  2.36 2.67       18.6       101    2.80       3.24
4    1   14.37  1.95 2.50       16.8       113    3.85       3.49
5    1   13.24  2.59 2.87       21.0       118    2.80       2.69
6    1   14.20  1.76 2.45       15.2       112    3.27       3.39
  Nonflavanoids Proanthocyanins Color  Hue Dilution Proline
1          0.28            2.29  5.64 1.04     3.92    1065
2          0.26            1.28  4.38 1.05     3.40    1050
3          0.30            2.81  5.68 1.03     3.17    1185
4          0.24            2.18  7.80 0.86     3.45    1480
5          0.39            1.82  4.32 1.04     2.93     735
6          0.34            1.97  6.75 1.05     2.85    1450

> df <- scale(wine[-1])                                          (1)
> head(df)

  Alcohol Malic   Ash Alcalinity Magnesium Phenols Flavanoids
1    1.51 -0.56  0.23      -1.17      1.91    0.81       1.03
2    0.25 -0.50 -0.83      -2.48      0.02    0.57       0.73
3    0.20  0.02  1.11      -0.27      0.09    0.81       1.21
4    1.69 -0.35  0.49      -0.81      0.93    2.48       1.46
5    0.29  0.23  1.84       0.45      1.28    0.81       0.66
6    1.48 -0.52  0.30      -1.29      0.86    1.56       1.36
  Nonflavanoids Proanthocyanins Color   Hue Dilution Proline
1         -0.66            1.22  0.25  0.36     1.84    1.01
2         -0.82           -0.54 -0.29  0.40     1.11    0.96
3         -0.50            2.13  0.27  0.32     0.79    1.39
4         -0.98            1.03  1.18 -0.43     1.18    2.33
5          0.23            0.40 -0.32  0.36     0.45   -0.04
6         -0.18            0.66  0.73  0.40     0.34    2.23

> wssplot(df)                                                    (2)
> set.seed(1234)                                                 (2)
> nc <- NbClust(df, min.nc=2, max.nc=15, method="kmeans")        (2)
> fviz_nbclust(nc)                                               (2)
```

```
> set.seed(1234)
> fit.km <- kmeans(df, 3, nstart=25)                    ❸
> fit.km$size

[1] 62 65 51

> fit.km$centers

  Alcohol Malic   Ash Alcalinity Magnesium Phenols Flavanoids Nonflavanoids
1    0.83 -0.30  0.36      -0.61     0.576   0.883      0.975        -0.561
2   -0.92 -0.39 -0.49       0.17    -0.490  -0.076      0.021        -0.033
3    0.16  0.87  0.19       0.52    -0.075  -0.977     -1.212         0.724
  Proanthocyanins Color   Hue Dilution Proline
1           0.579  0.17  0.47     0.78    1.12
2           0.058 -0.90  0.46     0.27   -0.75
3          -0.778  0.94 -1.16    -1.29   -0.41

> aggregate(wine[-1], by=list(cluster=fit.km$cluster), mean)

  cluster Alcohol Malic Ash Alcalinity Magnesium Phenols Flavanoids
1       1      14   1.8 2.4         17       106     2.8        3.0
2       2      12   1.6 2.2         20        88     2.2        2.0
3       3      13   3.3 2.4         21        97     1.6        0.7
  Nonflavanoids Proanthocyanins Color  Hue Dilution Proline
1          0.29             1.9   5.4 1.07      3.2    1072
2          0.35             1.6   2.9 1.04      2.8     495
3          0.47             1.1   7.3 0.67      1.7     620
```

❶ **Стандартизация данных.**

❷ **Определение количества кластеров.**

❸ **Выполнение кластеризации методом *k*-средних.**

Поскольку переменные различаются диапазонами значений, они стандартизируются перед кластеризацией ❶. Далее с помощью функций `wssplot()` и `NbClust()` определяется оптимальное количество кластеров ❷. На рис. 16.4 видно отчетливое падение суммы квадратов расстояний внутри групп при переходе от одного к трем кластерам. С увеличением числа кластеров больше трех падение замедляется, что позволяет предположить, что решение с тремя кластерами может хорошо соответствовать данным. На рис. 16.5 видно, что 19 критериев из 23, предоставляемых пакетом `NbClust`, предполагают трехкластерное решение. Обратите внимание, что не все 30 критериев могут быть вычислены для каждого набора данных.

Для получения окончательного решения вызывается функция `kmeans()` и центроиды выводятся на экран ❸. Поскольку полученные центроиды основаны на стандартизированных данных, к набору кластеров применяется функция `aggregate()`, чтобы определить средние значения переменных для каждого кластера в исходной метрике.

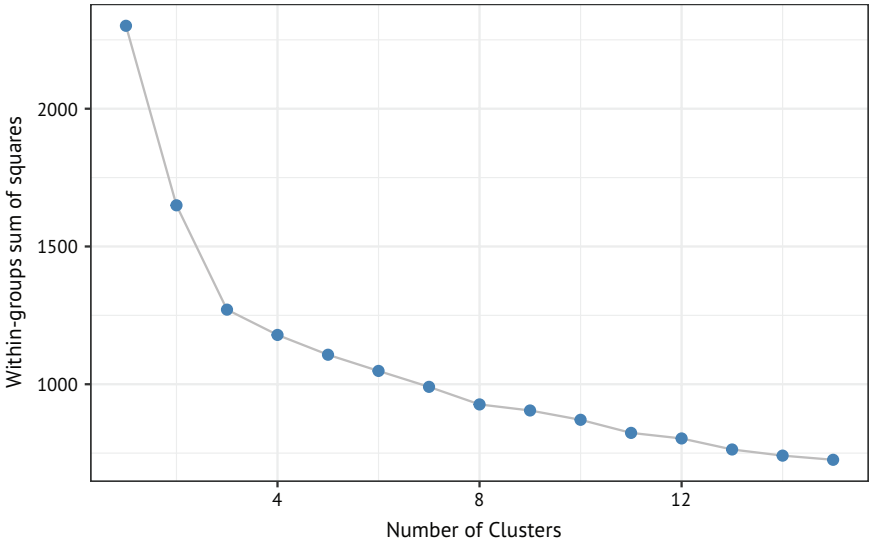


Рис. 16.4. График зависимости суммы квадратов расстояний внутри групп от количества извлеченных кластеров. Резкое уменьшение суммы квадратов при увеличении числа кластеров от одного до трех (и последующее более плавное уменьшение) предполагает, что оптимальным будет решение с тремя кластерами

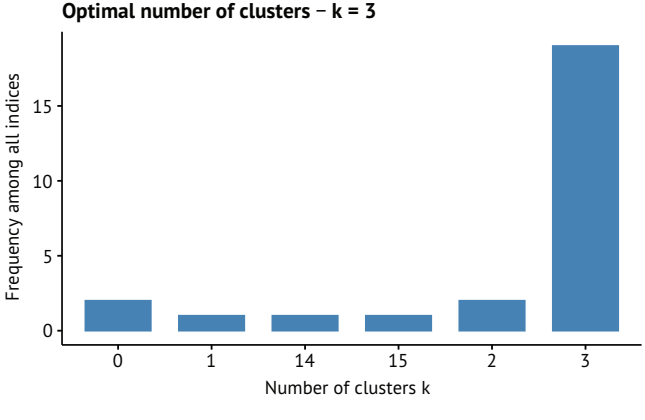


Рис. 16.5. Рекомендуемое количество кластеров по результатам вычисления 26 критериев, поддерживаемых пакетом NbClust

Самый простой способ сравнить кластеры – это построить диаграмму профиля кластера. Листинг 16.5 продолжает пример из листинга 16.4.

Листинг 16.5. Диаграмма профиля кластера

```
library(ggplot2)
library(tidyr)
means <- as.data.frame(fit.km$centers)                    1
means$cluster <- 1:nrow(means)                            1
```

```
plotdata <- gather(means, key="variable", value="value", -cluster)    (2)

ggplot(plotdata,                                                      (3)
       aes(x=variable,
           y=value,
           fill=variable,
           group=cluster)) +
  geom_bar(stat="identity") +
  geom_hline(yintercept=0) +
  facet_wrap(~cluster) +
  theme_bw() +
  theme(axis.text.x=element_text(angle=90, vjust=0),
        legend.position="none") +
  labs(x="", y="Standardized scores",
       title = "Mean Cluster Profiles")
```

(1) Подготовка профилей средних значений.

(2) Преобразование данных в длинный формат.

(3) Вывод профилей в виде столбиковых диаграмм.

Сначала вычисляются средние значения стандартизированных переменных по кластерам и добавляется переменная, описывающая принадлежность к кластеру (1). Затем эта таблица данных преобразуется из широкого формата в длинный (это преобразование описано в разделе 5.5.2) (2). Наконец, профили отображаются в виде набора столбиковых диаграмм (3). Результаты показаны на рис. 16.6. Профили средних значений по кластерам помогают увидеть, в чем заключается уникальность каждого кластера. Например, по сравнению с кластерами 2 и 3 кластер 1 имеет высокие средние значения по крепости (Alcohol), фенолам (Flavanoids), проантоцианам (Proanthocyanins) и пролину (Proline).

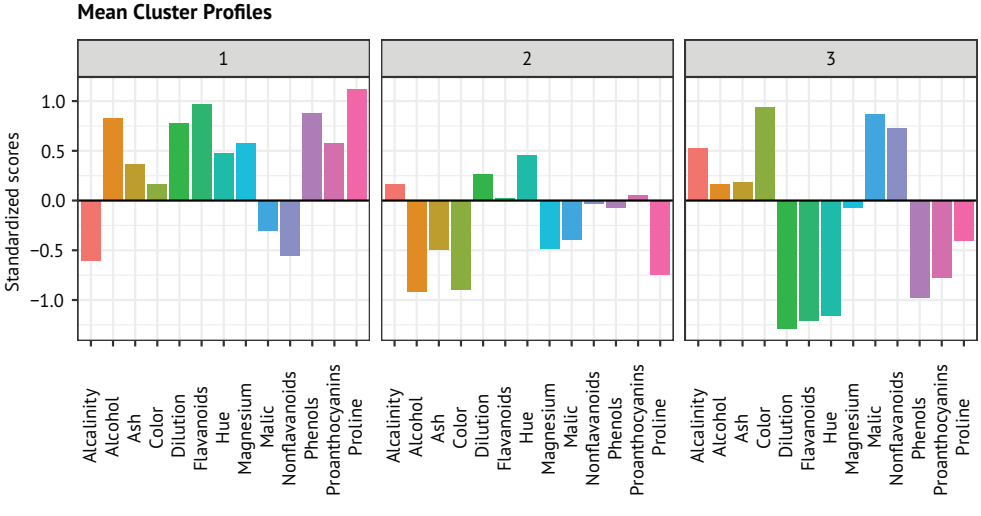


Рис. 16.6. Профили средних значений по кластерам на основе стандартизированных данных. Эта диаграмма помогает определить отличительные особенности каждого кластера

Другой подход к визуализации результатов кластерного анализа – построение *двумерной кластерной диаграммы*. Диаграмма создается путем нанесения координат каждого наблюдения (вина) на систему координат, представленную первыми двумя главными компонентами, полученными из 13 переменных анализа. (Главные компоненты описаны в главе 14.) Цвет и форма каждой точки указывают на ее принадлежность к тому или иному кластеру. Метки точек – это номера строк в таблице с исходными данными. Кроме того, каждый кластер окружен наименьшим эллипсом, включающим все точки этого кластера.

Двумерную кластерную диаграмму можно создать с помощью функции `fviz_cluster()` из пакета `factoextra`:

```
library(factoextra)
fviz_cluster(fit.km, data=df)
```

Получившаяся диаграмма показана на рис. 16.7. Как видите, кластеры 1 и 3 наиболее не похожи. Вина 4 и 19 похожи, а вина 4 и 171 очень разные.

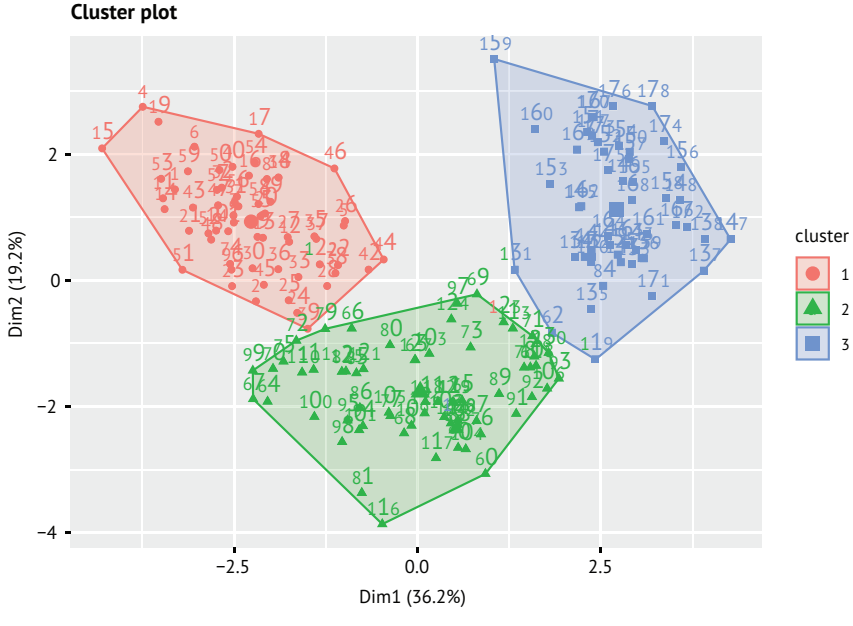


Рис. 16.7. Кластерная диаграмма для 178 вин, сгруппированных в 3 кластера. Каждое вино наносится на координатную сетку, образованную двумя первыми главными компонентами. Такие диаграммы могут помочь увидеть сходства/различия между винами и кластерами

Кластерный анализ обычно относят к категории методов машинного обучения без учителя, потому что он не пытается предсказать истинное значение результата. Однако в примере с набором данных о винах на самом деле есть три сорта вин (`Type`).

Насколько хорошо кластеризация методом *k*-средних вскрыла фактическую структуру данных, описываемую переменной Type? Давайте получим перекрестную таблицу для Type (сорта вина) и информации о принадлежности к кластерам, как показано ниже:

```
> ct.km <- table(wine$Type, fit.km$cluster)
> ct.km
     1  2  3
  1 59  0  0
  2  3 65  3
  3  0  0 48
```

Соответствие между типами и кластерами можно оценить количественно, используя скорректированный индекс Рэнда (adjusted Rand index), предоставляемый пакетом flexclust:

```
> library(flexclust)
> randIndex(ct.km)
[1] 0.897
```

Скорректированный индекс Рэнда – это мера согласия между двумя способами разделения на группы с поправкой на случайность. Его величина варьируется от –1 (нет согласия) до 1 (полное согласие). Согласие между сортом вина, указанным в данных, и кластерным решением составляет 0,9. Неплохо! Может, теперь выпьем вина?

16.4.2. Разделение вокруг медоидов

Метод *k*-средних основан на средних значениях, поэтому он может быть чувствителен к выбросам. Более надежное решение обеспечивается методом разделения вокруг медоидов (Partitioning Around Medoids, PAM). В этом случае кластеры идентифицируются не центроидами (векторами средних значений переменных), а наиболее репрезентативными наблюдениями, называемыми *медоидами* (medoid). В отличие от метода *k*-средних, использующего евклидовы расстояния, метод PAM может использовать любую меру расстояния и, соответственно, работать со смешанными типами данных, а не только с непрерывными переменными.

Алгоритм PAM выглядит следующим образом.

1. Произвольно выбрать *K* наблюдений (медоидов).
2. Вычислить расстояние/несходство между каждым наблюдением и каждым медоидом.
3. Связать каждое наблюдение с ближайшим медоидом.
4. Вычислить сумму расстояний между каждым наблюдением и его медоидом (общая стоимость).
5. Выбрать точку, не являющуюся медоидом, и поменять ее местами с медоидом.
6. Повторно связать каждую точку с ближайшим медоидом.

7 Вычислить общую стоимость.

8 Если общая стоимость уменьшилась, оставить новую точку медоидом.

9 Повторять шаги 5–8, пока не будет достигнута итерация, в которой медоиды останутся неизменными.

Хорошее описание математического аппарата, лежащего в основе метода PAM, можно найти по адресу: http://en.wikipedia.org/wiki/k-medoids (обычно я не цитирую Википедию, но это действительно отличное описание).

Для разделения вокруг медоидов можно использовать функцию `pam()` из пакета `cluster`. Она имеет следующий синтаксис: `pam(x, k, metric="euclidean", stand=FALSE)`, где *x* – матрица или таблица данных, *k* – количество кластеров, `metric` – тип используемой меры расстояния/несходства, `stand` – это логическое значение, указывающее на необходимость стандартизации переменных перед вычислением расстояний. В листинге 16.6 показано применение метода PAM для кластеризации набора данных `wine`.

Листинг 16.6. Разделение вокруг медоидов для данных в наборе wine

```
> library(cluster)
> set.seed(1234)
> fit.pam <- pam(wine[-1], k=3, stand=TRUE)          (1)
> fit.pam$medoids                                    (2)

     Alcohol Malic  Ash Alcalinity Magnesium Phenols Flavanoids
[1,]    13.5  1.81 2.41       20.5       100    2.70       2.98
[2,]    12.2  1.73 2.12       19.0        80    1.65       2.03
[3,]    13.4  3.91 2.48       23.0       102    1.80       0.75
     Nonflavanoids Proanthocyanins Color  Hue Dilution Proline
[1,]          0.26            1.86   5.1 1.04     3.47     920
[2,]          0.37            1.63   3.4 1.00     3.17     510
[3,]          0.43            1.41   7.3 0.70     1.56     750
```

(1) **Кластеризовать стандартизированные данные.**

(2) **Вывести информацию о медоидах.**

Обратите внимание, что медоиды – это фактические наблюдения в наборе данных `wine`. В данном случае это наблюдения 36, 107 и 175, и они были выбраны для представления трех кластеров.

Также обратите внимание, что в этом случае PAM работает не так хорошо, как метод *k*-средних:

```
> ct.pam <- table(wine$Type, fit.pam$clustering)

    1  2  3
1 59  0  0
2 16 53  2
3  0  1 47
```

```
> randIndex(ct.pam)
[1] 0.699
```

Скорректированный индекс Рэнда уменьшился с 0,9 (для *k*-средних) до 0,7. Создание диаграммы кластерного профиля и двумерной кластерной диаграммы я оставляю вам в качестве упражнения.

16.5. Исключение несуществующих кластеров

Прежде чем закончить обсуждение, хотелось бы сделать одно предостережение. Кластерный анализ – это методология, предназначенная для выявления связанных подгрупп в наборе данных. Однако эта методология способна находить кластеры там, где их нет на самом деле.

Взгляните на следующий код:

```
library(fMultivar)

library(ggplot2)
set.seed(1234)
df <- rnorm2d(1000, rho=.5)
df <- as.data.frame(df)
ggplot(df, aes(x=V1, y=V2)) +
  geom_point(alpha=.3) + theme_minimal() +
  labs(title="Bivariate Normal Distribution with rho=0.5")
```

Функция `rnorm2d()` из пакета `fMultivar` выбирает 1000 наблюдений из двумерного нормального распределения с корреляцией 0,5. На рис. 16.8 показана получившаяся диаграмма. Очевидно, что в этих данных нет кластеров.

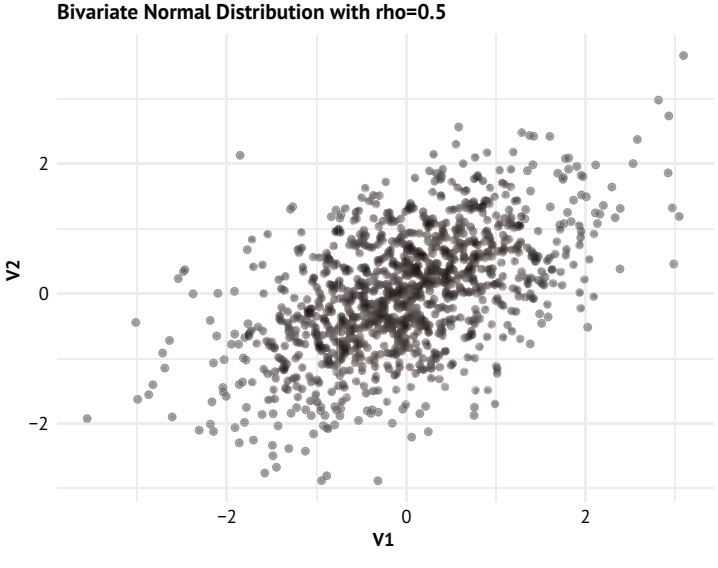


Рис. 16.8. Двумерные нормально распределенные данные (n = 1000). В этих данных нет кластеров

Теперь используем функции `wssplot()` и `NbClust()` для определения количества имеющихся кластеров:

```
wssplot(df)
library(NbClust)
library(factoextra)
nc <- NbClust(df, min.nc=2, max.nc=15, method="kmeans")
fviz_nbclust(nc)
```

Результаты показаны на рис. 16.9 и 16.10.

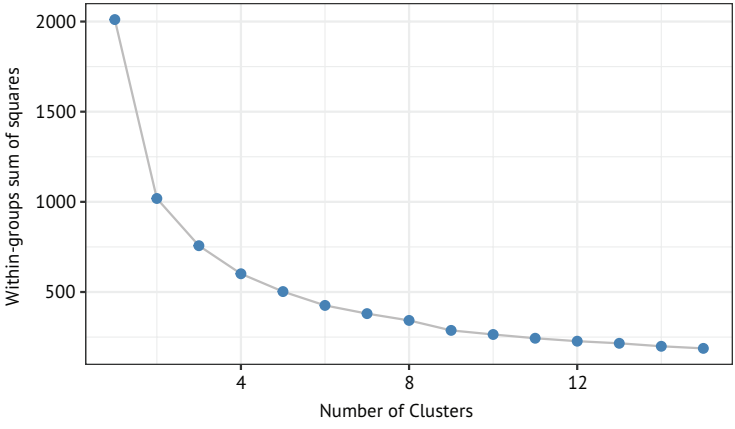


Рис. 16.9. График зависимости суммы квадратов расстояний внутри групп от количества кластеров, полученных методом *k*-средних, для двумерных нормальных данных

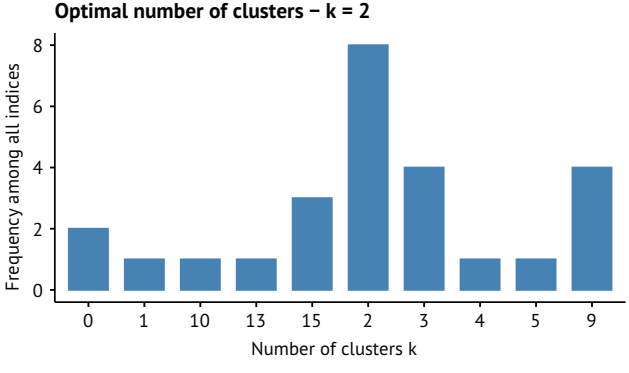


Рис. 16.10. Количество кластеров, рекомендуемое для двумерных нормально распределенных данных по критериям в пакете NbClust. Предлагаются два кластера

Оба подхода предполагают наличие как минимум двух кластеров. Если провести кластерный анализ методом *k*-средних и попытаться выделить два кластера:

```
library(ggplot2)
fit <- kmeans(df, 2)
df$cluster <- factor(fit$cluster)
ggplot(data=df, aes(x=V1, y=V2, color=cluster, shape=cluster)) +
       theme_minimal() +
       geom_point(alpha=.5) +
       ggtitle("Clustering of Bivariate Normal Data")
```

то получится картина, изображенная на рис. 16.11.

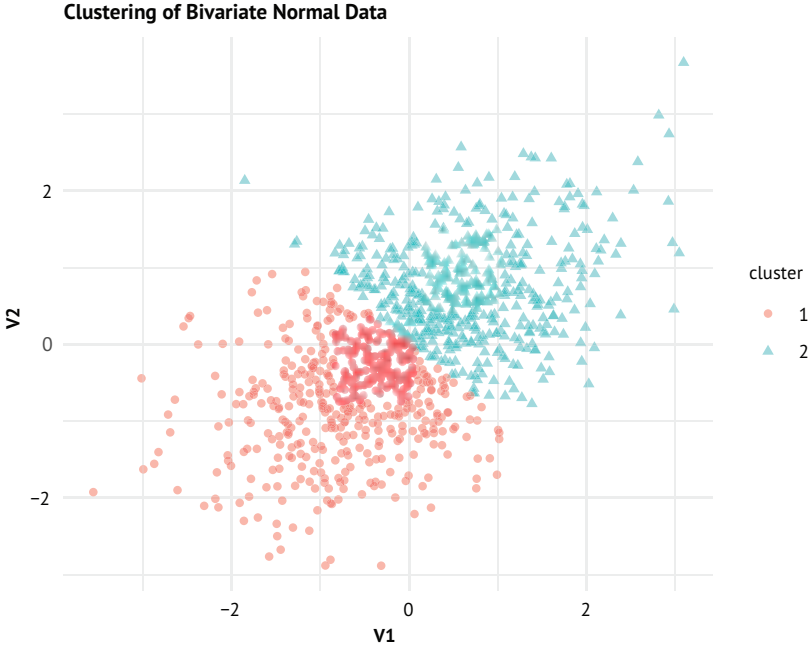


Рис. 16.11. Результаты кластерного анализа методом *k*-средних для двумерных нормально распределенных данных с выделением двух кластеров. Обратите внимание, что кластеры произвольно делят данные

Очевидна неестественность этого деления данных. Здесь нет настоящих кластеров. Как избежать этой ошибки? Я нашел два полезных решения, несмотря на кажущуюся ненадежность. Первый – критерий DIP, предоставляемый пакетом `clusterability`:

```
> library(clusterability)
> clusterabilitytest(df[-3], "dip")

Null Hypothesis: number of modes = 1
Alternative Hypothesis: number of modes > 1
p-value: 0.9655
Dip statistic: 0.00823
```

`df[-3]` удаляет из данных факторную переменную (принадлежность к кластеру). Нулевая гипотеза утверждает, что существует

один кластер (мода). Поскольку $p > 0{,}05$, мы не можем отвергнуть эту гипотезу, т. е. данные не имеют кластерной структуры.

Другой подход основан на кубическом критерии кластеризации (Cubic Clustering Criteria, CCC), поддерживаемом пакетом `NbClust`. CCC часто помогает выявить ситуации, когда данные не имеют структуры. Следующий код:

```
CCC = nc$All.index[, 4]
k <- length(CCC)
plotdata <- data.frame(CCC = CCC, k = seq_len(k))
ggplot(plotdata, aes(x=k, y=CCC)) +
  geom_point() + geom_line() +
  theme_minimal() +
  scale_x_continuous(breaks=seq_len(k)) +
  labs(x="Number of Clusters")
```

создает диаграмму, изображенную на рис. 16.12. Когда все значения CCC отрицательные и уменьшаются для двух или более кластеров, то это обычно говорит об унимодальности распределения данных.

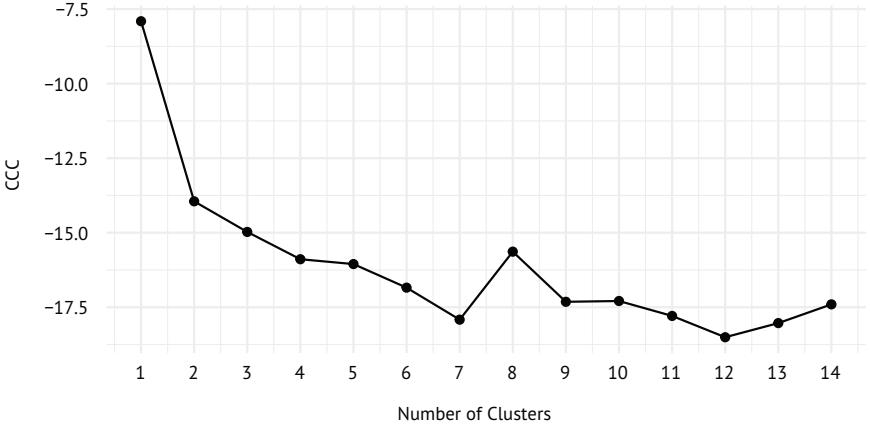


Рис. 16.12. График кубического критерия кластеризации для двумерных нормально распределенных данных. Он правильно предполагает отсутствие кластеров

Способность методов кластерного анализа находить несуществующие кластеры делает важным этап проверки его результатов. Если вы пытаетесь идентифицировать кластеры, реальные в некотором смысле, убедитесь, что результаты устойчивы и воспроизводимы. Попробуйте разные методы кластеризации и воспроизведите результаты с новыми выборками. Если снова и снова выделяются одни и те же кластеры, то это придает дополнительную уверенность в результатах.

16.6. Дополнительная информация

Кластерный анализ – обширная тема, и в R имеется один из самых лучших наборов средств для применения этой методологии. Чтобы узнать больше об имеющихся возможностях, можно обратиться к обзору задач по кластерному анализу «CRAN Task View for Cluster Analysis & Finite Mixture Models» (http://cran.r-project.org/web/views/Cluster.html). Кроме того, Тан (Tan), Штейнбах (Steinbach) и Кумар (Kumar) (2006) написали прекрасную книгу по методам интеллектуального анализа данных, включающую замечательную главу по кластерному анализу, доступную бесплатно (http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf). Наконец, в публикации Everitt, Landau, Leese & Stahl (2011) вы найдете отличное и практичное руководство по этому предмету.

Итоги

- Кластерный анализ – распространенный подход к объединению наблюдений в связные группы.
- Не существует общепринятого определения того, что подразумевается под «кластером» или под «расстоянием между кластерами», поэтому было разработано множество подходов к кластеризации.
- Двумя наиболее популярными категориями кластерного анализа являются иерархическая и разделяющая кластеризация. Внутри каждой категории существует множество методов кластеризации. Ни один метод не является лучшим во всех ситуациях.
- Также не существует единого лучшего подхода для определения количества кластеров в наборе данных. Часто бывает полезно попробовать несколько разных подходов и остановиться на том, который наиболее значим или практичен.
- Кластерный анализ может выявлять кластеры независимо от того, существуют ли они на самом деле! Если ваша цель состоит в том, чтобы разделить данные на удобные, связные группы (например, при решении задачи сегментации клиентов), то это может быть вполне приемлемо. Но если вы стремитесь выявить естественные группы с теоретически значимыми различиями (например, подтипы депрессии на основе симптомов и анамнеза), то важно подтвердить полученные выводы, повторив анализ на новых данных.

17 Классификация

В этой главе:

- классификация с использованием деревьев решений;
- классификатор на основе леса решающих деревьев;
- создание машины опорных векторов;
- оценка точности классификации;
- оценка сложных моделей.

Аналитикам часто требуется предсказывать категориальный результат по набору независимых переменных. Вот несколько примеров:

- прогнозирование способности человека погасить кредит на основе его демографической и финансовой истории;
- определение наличия у пациента скорой помощи сердечного приступа на основании симптомов и основных показателей жизнедеятельности;
- классификация электронной почты на спам и не спам по наличию ключевых слов, изображений, гипертекста, заголовка и адреса отправителя.

Каждый из этих случаев включает прогнозирование бинарного категориального результата (хорошая/плохая кредитоспособность, есть/нет сердечного приступа, спам / не спам) на основе набора не-

зависимых переменных (также называемых *предикторами*, или *признаками*). Цель состоит в том, чтобы найти точный метод классификации, позволяющий относить новые случаи к одной из двух групп.

Область машинного обучения с учителем предлагает множество методов классификации для прогнозирования категориальных результатов, включая логистическую регрессию, деревья решений, случайные леса (ансамбли деревьев решений), метод опорных векторов и искусственные нейронные сети. Первые четыре обсуждаются в этой главе. Искусственные нейронные сети выходят за рамки данной книги, но желающим познакомиться с ними поближе рекомендую книги Ciaburro & Venkateswaran (2017) и Chollet & Allaire (2018).

Обучение с учителем начинается с получения набора наблюдений, содержащих значения не только переменных-предикторов, но и результатов. Затем набор данных делится на обучающую и контрольную выборки. Прогнозная модель разрабатывается с использованием данных из обучающей выборки, а ее точность проверяется с применением данных из контрольной выборки. Необходимы обе выборки, потому что методы классификации максимизируют предсказание для заданного набора данных. Оценки их эффективности будут чрезмерно оптимистичными, если оценивать их на тех же данных, на которых обучалась модель. Проверяя прогностическую способность обученной модели на отдельной контрольной выборке, можно получить более реалистичную оценку ее точности. Создав эффективную прогнозную модель, ее можно использовать для прогнозирования результатов в ситуациях, когда известны только предикторы.

В этой главе мы используем пакеты `rpart`, `rattle` и `partykit` для создания и визуализации деревьев решений; пакет `randomForest` для подбора случайных лесов и пакет `e1071` для создания машин опорных векторов. Модель логистической регрессии будет подбираться с помощью функции `glm()`, имеющейся в стандартном дистрибутиве R. Перед опробованием примеров обязательно установите необходимые пакеты:

```
pkgs <- c("rpart", "rattle", "partykit",
          "randomForest", "e1071")
install.packages(pkgs, depend=TRUE)
```

Основным примером в этой главе послужит набор данных о раке молочной железы в штате Висконсин, первоначально размещенный в репозитории машинного обучения UCI Machine Learning Repository. Нашей целью будет создание модели, прогнозирующей наличие у пациентки рака молочной железы по результатам тонкоигольной аспирационной биопсии (анализ образцов ткани, взятых с помощью тонкой полой иглы из опухоли или образования непосредственно под кожей).

17.1. Подготовка данных

Набор данных о раке молочной железы в штате Висконсин доступен в виде текстового файла в формате CSV на сайте UCI Machine Learning Server (http://archive.ics.uci.edu/ml). Он содержит 699 результатов тонкоигольной аспирационной биопсии, из которых 458 (65,5 %) доброкачественные и 241 (34,5 %) злокачественные. Набор данных содержит 11 переменных, но не включает их имена. В шестнадцати образцах отсутствуют данные, и такие отсутствующие данные закодированы знаком вопроса (?).

Вот эти переменные:

- ID (идентификатор);
- Clump thickness (толщина скопления клеток);
- Uniformity of cell size (однородность размеров клеток);
- Uniformity of cell shape (однородность форм клеток);
- Marginal adhesion (межклеточная мембранная адгезия);
- Single epithelial cell size (размер отдельной клетки эпителия);
- Bare nuclei (единичные «голые» ядра);
- Bland chromatin (рыхлый [деконденсированный] хроматин);
- Normal nucleoli (нормальные ядрышки);
- Mitoses (динамика митоза);
- Class (класс).

Первая переменная – ID (мы ее отбросим), а последняя переменная Class (класс) содержит результат, представленный одним из двух возможных значений: 2 = доброкачественное, 4 = злокачественное (новообразование). Мы также исключим из анализа наблюдения, содержащие пропущенные значения.

Каждое наблюдение содержит еще девять цитологических характеристик, которые, как было установлено ранее, коррелируют со злокачественным новообразованием. Каждая из этих переменных оценивается по шкале от 1 (ближе к доброкачественному новообразованию) до 10 (ближе к анапластическому раку). Но ни один предиктор в одиночку не способен отличить доброкачественные образцы от злокачественных. Задача состоит в том, чтобы найти набор правил классификации, которые можно использовать для точного предсказания злокачественности на основании некоторой комбинации этих девяти характеристик. Подробности можно найти в публикации Mangasarian & Wolberg (1990).

Код в листинге 17.1 загружает текстовый файл с данными из репозитория UCI и случайным образом делит его на обучающую (70 %) и контрольную (30 %) выборки.

Листинг 17.1. Подготовка данных о раке молочной железы

```
loc <- "http://archive.ics.uci.edu/ml/machine-learning-databases"
ds  <- "breast-cancer-wisconsin/breast-cancer-wisconsin.data"
url <- paste(loc, ds, sep="/")

breast <- read.table(url, sep=",", header=FALSE, na.strings="?")
names(breast) <- c("ID", "clumpThickness", "sizeUniformity",
                   "shapeUniformity", "maginalAdhesion",
                   "singleEpithelialCellSize", "bareNuclei",
                  "blandChromatin", "normalNucleoli", "mitosis", "class")

df <- breast[-1]
df$class <- factor(df$class, levels=c(2,4),
                   labels=c("benign", "malignant"))
df <- na.omit(df)

set.seed(1234)
index <- sample(nrow(df), 0.7*nrow(df))
train <- df[index,]
test <- df[-index,]
table(train$class)
table(test$class)
```

В обучающей выборке получилось 478 случаев (302 доброкачественных, 176 злокачественных), в контрольной – 205 случаев (142 доброкачественных, 63 злокачественных).

Обучающая выборка будет использоваться для создания моделей классификации с использованием логистической регрессии, дерева решений, условного дерева решений, случайного леса (ансамбля решающих деревьев) и метода опорных векторов. Контрольная выборка будет использоваться для оценки эффективности этих моделей. Использование одного и того же примера на протяжении всей главы позволит нам сравнивать результаты, получаемые каждым подходом.

17.2. Логистическая регрессия

Логистическая регрессия – это тип обобщенных линейных моделей, часто используемых для прогнозирования бинарного результата на основе набора числовых переменных (раздел 13.2). Для подгонки моделей логистической регрессии используется функция `glm()`, входящая в стандартный дистрибутив R. Категориальные предикторы (факторы) автоматически заменяются набором фиктивных переменных. Все предикторы в данных по раку груди в штате Висконсин являются числовыми, поэтому фиктивное кодирование не требуется. В листинге 17.2 представлен код, реализующий логистический регрессионный анализ данных.

Листинг 17.2. Логистическая регрессия с glm()

```
> fit.logit <- glm(class~., data=train, family=binomial())    (1)
> summary(fit.logit)                                          (2)

Call:
glm(formula = class ~ ., family = binomial(), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.6141  -0.1204  -0.0744   0.0236   2.1845

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)               -9.68650    1.29722  -7.467 8.20e-14 ***
clumpThickness             0.48002    0.15244   3.149  0.00164 **
sizeUniformity             0.05643    0.29272   0.193  0.84714
shapeUniformity            0.13180    0.31643   0.417  0.67703
maginalAdhesion            0.40721    0.14038   2.901  0.00372 **
singleEpithelialCellSize -0.03274    0.18095  -0.181  0.85643
bareNuclei                 0.44744    0.11176   4.004 6.24e-05 ***
blandChromatin             0.48257    0.19220   2.511  0.01205 *
normalNucleoli             0.23550    0.12903   1.825  0.06798 .
mitosis                    0.66184    0.28785   2.299  0.02149 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> prob <- predict(fit.logit, test, type="response")            (3)
> logit.pred <- factor(prob > .5, levels=c(FALSE, TRUE),       (3)
                       labels=c("benign", "malignant"))       (3)
> logit.perf <- table(test$class, logit.pred,                  (4)
                      dnn=c("Actual", "Predicted"))            (4)
> logit.perf

           Predicted
Actual      benign malignant
  benign       140         2
  malignant      3        60
```

(1) **Подбор модели логистической регрессии.**

(2) **Вывод информации о модели.**

(3) **Классификация новых случаев.**

(4) **Оценка прогнозирующей способности модели.**

В первой строке происходит обучение (подгонка) модели логистической регрессии с использованием переменной `class` в качестве зависимой переменной и остальных переменных в качестве предикторов (1). Модель обучается на случаях в обучающей выборке – таблице данных `train`. Далее на экран выводятся коэффициенты модели (2). Рекомендации по интерпретации коэффициентов логистической модели вы найдете в разделе 13.2.

Далее полученная модель используется для классификации случаев в контрольной выборке – таблице данных `test`. По умолчанию функция `predict()` предсказывает логарифмическую вероятность злокачественного исхода. При использовании параметра `type="response"` (3) возвращается простая вероятность злокачественной классификации. В следующей строке случаи с вероятностью выше 0,5 включаются в группу злокачественных исходов, а случаи с вероятностью ниже или равной 0,5 – доброкачественных.

В заключение выводится сводная таблица фактического состояния и предсказанного исхода (называемая матрицей ошибок) (4). Как можно видеть в этой таблице, 140 случаев доброкачественного исхода были верно классифицированы как доброкачественные, а 60 случаев злокачественного исхода – как злокачественные.

Общее количество правильно классифицированных случаев в контрольной выборке (также называемое точностью) составило (140 + 60)/205, или 98 %. Статистические данные для оценки точности классификации более подробно обсуждаются в разделе 17.6.

Прежде чем двигаться дальше, хочу обратить ваше внимание на то, что коэффициенты трех переменных-предикторов (`sizeUniformity`, `shapeUniformity` и `singleEpithelialCellSize`) не отличаются от нуля на уровне значимости $p < 0{,}10$. Как правильнее поступить с переменными-предикторами, имеющими незначимые коэффициенты?

В контексте прогнозирования такие переменные обычно следует удалить из окончательной модели. Это особенно важно, когда большое количество неинформативных переменных-предикторов добавляют то, что по существу является шумом.

В таких случаях для создания модели меньшего размера с меньшим количеством переменных можно использовать пошаговую логистическую регрессию. Переменные-предикторы добавляются или удаляются для получения модели с меньшим значением информационного критерия Акаике (Akaike Information Criterion, AIC). Получить уменьшенную модель в текущем примере можно так:

```
logit.fit.reduced <- step(fit.logit)
```

Из уменьшенной модели исключены три вышеупомянутые переменные. При применении к контрольной выборке эта модель показывает такие же хорошие результаты. Попробуйте и убедитесь сами.

Следующий подход, который мы рассмотрим, включает создание деревьев решений или классификаций.

17.3. Деревья решений

Деревья решений популярны в сфере интеллектуального анализа данных. Этот подход предполагает создание набора двоичных разбиений переменных-предикторов для создания дерева, которое мож-

но использовать для классификации новых наблюдений в одну из двух групп. В этом разделе мы рассмотрим два типа деревьев решений: классические деревья и деревья условного вывода.

17.3.1 Классические деревья решений

Процесс построения *классического дерева решений* начинается с выбора бинарной переменной исхода (в данном случае таковой является переменная `class` с двумя возможными значениями – доброкачественный/злокачественный) и набора переменных-предикторов (здесь девять цитологических параметров). Алгоритм можно определить так:

1. Выбрать переменную-предиктор, которая лучше всего делит данные на две группы, чтобы чистота (однородность) результата в двух группах была максимальной (то есть как можно больше доброкачественных случаев в одной группе и злокачественных случаев в другой). Если переменная-предиктор – непрерывная, то нужно выбрать точку порога, максимизирующую чистоту двух групп. Если переменная-предиктор является категориальной (что не так в данном случае), то нужно подобрать такое сочетание категорий, чтобы обеспечить разделение на две группы с максимальной чистотой.
2. Разделить данные на две группы и продолжить процесс для каждой подгруппы отдельно.
3. Повторять шаги 1 и 2, пока количество наблюдений в подгруппе не станет меньше минимального или пока не будет получено разделение с чистотой лучше заданного порога.
4. Подгруппы в конечном наборе называются конечными узлами. Каждый конечный узел относится к той или иной категории исхода на основе наиболее частого значения результата для выборки в этом узле.
5. Чтобы классифицировать случай, его нужно пропустить вниз по дереву до конечного узла и присвоить ему модальное значение результата, назначенное на шаге 3.

К сожалению, этот процесс приводит к созданию слишком большого дерева, которое страдает от эффекта переобучения. В результате новые случаи классифицируются довольно плохо. Чтобы компенсировать этот недостаток, можно усечь дерево, выбрав дерево с наименьшей ошибкой предсказания при перекрестной проверке. Это усеченное дерево затем можно использовать для прогнозов на основе новых данных.

В R деревья решений можно наращивать и усекать с помощью функций `rpart()` и `prune()` из пакета `rpart`. В листинге 17.3 показан код, создающий дерево решений для классификации данных цитологических анализов как доброкачественных или злокачественных.

Листинг 17.3. Создание классического дерева решений с помощью rpart()

```
> library(rpart)
> dtree <- rpart(class ~ ., data=train, method="class",          (1)
                 parms=list(split="information"))                (1)
> dtree$cptable

          CP nsplit  rel error    xerror       xstd
1 0.79545455      0 1.00000000 1.0000000 0.05991467
2 0.07954545      1 0.20454545 0.3068182 0.03932359
3 0.01704545      2 0.12500000 0.1590909 0.02917149
4 0.01000000      5 0.07386364 0.1704545 0.03012819

> plotcp(dtree)

> dtree.pruned <- prune(dtree, cp=.01705)                         (2)

> library(rattle)
> fancyRpartPlot(dtree.pruned,  sub="Classification Tree")

> dtree.pred <- predict(dtree.pruned, test, type="class")         (3)
> dtree.perf <- table(test$class, dtree.pred,
                      dnn=c("Actual", "Predicted"))
> dtree.perf
           Predicted
Actual      benign malignant
  benign       136         6
  malignant      3        60
```

(1) **Наращивание дерева.**

(2) **Усечение дерева.**

(3) **Классификация новых случаев.**

Сначала выполняется наращивание дерева с помощью функции `rpart()` (1). Для изучения полученной модели можно использовать `print(dtree)` и `summary(dtree)` (здесь не показаны). Получившееся дерево может оказаться слишком большим и потребовать усечения.

Чтобы выбрать окончательный размер дерева, изучите компонент `cptable` списка, возвращаемого функцией `rpart()`. Он содержит данные об ошибках предсказания для различных размеров дерева. Параметр сложности (`cp`) определяет штраф за величину дерева. Размер дерева задается количеством ветвлений (`nsplit`). Дерево с n ветвлениями имеет $n + 1$ конечных узлов. Столбец `rel error` содержит частоту ошибок для дерева заданного размера для обучающей выборки. Ошибка перекрестной проверки (`xerror`) основана на 10-кратной перекрестной проверке (также с использованием обучающей выборки). Столбец `xstd` содержит стандартную ошибку перекрестной проверки.

Функция `plotcp()` строит график изменения ошибки при перекрестной проверке в зависимости от параметра сложности

(рис. 17.1). Хорошим значением окончательного размера дерева может служить размер наименьшего дерева с ошибкой перекрестной проверки в пределах одной стандартной ошибки минимального значения ошибки перекрестной проверки.

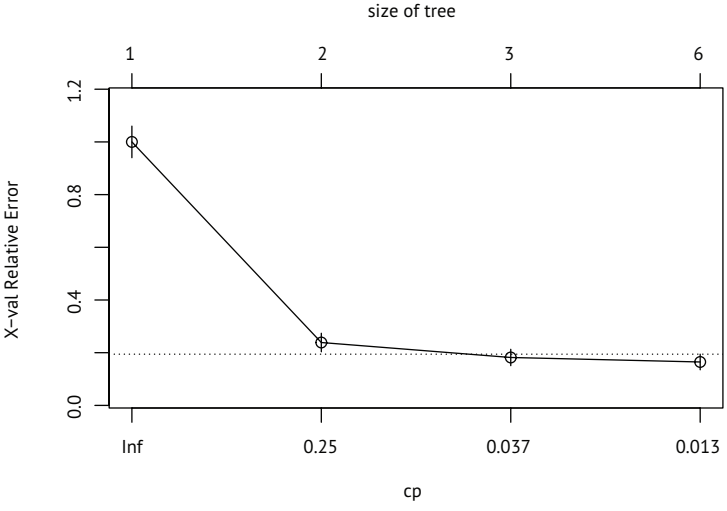


Рис. 17.1. Зависимость ошибки перекрестной проверки от параметра сложности. Пунктирная линия – верхний предел правила одного стандартного отклонения (0,16 + 0,03 = 0,19). График предлагает выбрать дерево с самым левым значением cp под линией

Минимальная ошибка перекрестной проверки составляет 0,16 при стандартной ошибке 0,03. В этом случае выбирается наименьшее дерево с ошибкой перекрестной проверки в пределах 0,16 ± 0,03 (то есть между 0,13 и 0,19). Судя по таблице `cptable` в листинге 17.3, этому требованию соответствует дерево с двумя ветвлениями (ошибка перекрестной проверки = 0,16). Точно так же можно выбрать размер дерева, соответствующий наибольшему параметру сложности под линией на рис. 17.1. И в этом случае предполагается выбор дерева с двумя ветвлениями (тремя конечными узлами).

Функция `prune()` использует параметр сложности для усечения дерева до нужного размера. Она принимает полное дерево и отсекает наименее важные ветвления на основе желаемого параметра сложности. Согласно предыдущему обсуждению, в этом примере использовался вызов `prune(dtree, cp=0.01705)`, возвращающий дерево нужного размера (2). Функция возвращает самое большое дерево с параметром сложности ниже указанного значения `cp`.

Функция `fancyRpartPlot()` из пакета `rattle` позволяет построить график конечного дерева решений (рис. 17.2). Она имеет несколько параметров (подробности ищите в справке `?fancyRpartPlot`). Параметр `type=2` (по умолчанию) выводит метки раз-

деления под каждым узлом. Дополнительно для каждого узла сообщаются доля каждого класса и процент наблюдений. Классификация наблюдений начинается с вершины дерева, и выполняется переход в левую ветвь, если условие истинно, или в правую в противном случае. Движение вниз продолжается до достижения конечного узла. Класс присваивается наблюдению на основе метки этого узла.

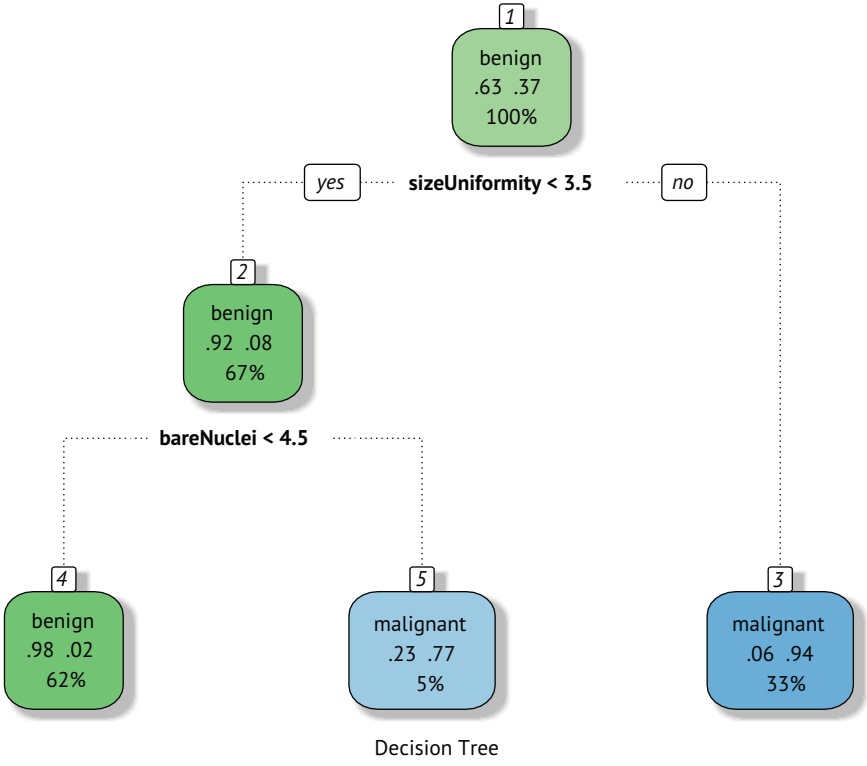


Рис. 17.2. Традиционное (усеченное) дерево решений для прогнозирования рака. Классификация начинается с вершины дерева, и выполняется переход в левую ветвь, если условие истинно, или в правую в противном случае. По достижении конечного узла производится классификация наблюдения. Каждый узел содержит вероятность отнесения к тому или иному классу, а также процент выборки

Наконец, для классификации наблюдений в контрольной выборке используется функция `predict()` (3). После классификации выводится сводная таблица соответствия фактического и прогнозируемого исхода. Общая точность на контрольной выборке составила 96 %. Обратите внимание, что деревья решений могут быть смещены в сторону выбора предикторов, которые имеют много уровней или много пропущенных значений.

17.3.2. Деревья условного вывода

Прежде чем перейти к случайным лесам (ансамблям деревьев), рассмотрим важную разновидность традиционных деревьев решений, называемую *деревом условного вывода*. Деревья условного вывода подобны традиционным деревьям, но переменные и ветвления выбираются на основе критериев значимости, а не показателей чистоты/однородности. В роли критерия значимости часто используется критерий перестановок (обсуждаемый в главе 12).

В этом случае алгоритм выглядит так:

1. Вычислить *p*-значения для взаимосвязей между каждым предиктором и зависимой переменной.
2. Выбрать предиктор с наименьшим *p*-значением.
3. Исследовать все возможные бинарные разбиения для выбранного предиктора и зависимой переменной (используя критерий перестановок) и выбрать наиболее значимое расщепление.
4. Разделить данные на две группы и продолжить процесс для каждой подгруппы.
5. Продолжать, пока разделения не перестанут быть значимыми или не будет достигнут минимальный размер узла.

Деревья условного вывода создаются функцией `ctree()` из пакета `party`. В листинге 17.4 показан код, создающий условное дерево решений на основе данных цитологических анализов.

Листинг 17.4. Создание дерева условного вывода с помощью ctree()

```
library(partykit)
fit.ctree <- ctree(class~., data=train)
plot(fit.ctree, main="Conditional Inference Tree",
     gp=gpar(fontsize=8))

> ctree.pred <- predict(fit.ctree, test, type="response")
> ctree.perf <- table(test$class, ctree.pred,
                      dnn=c("Actual", "Predicted"))
> ctree.perf

          Predicted
Actual      benign malignant
  benign       138         4
  malignant      2        61
```

Обратите внимание, что деревья условного вывода не требуют усечения, да и сам процесс выращивания более автоматизирован. Кроме того, пакет `partykit` предлагает привлекательные возможности построения диаграмм. На рис. 17.3 показано дерево условного вывода. Заштрихованные области в узлах представляют доли злокачественных случаев.

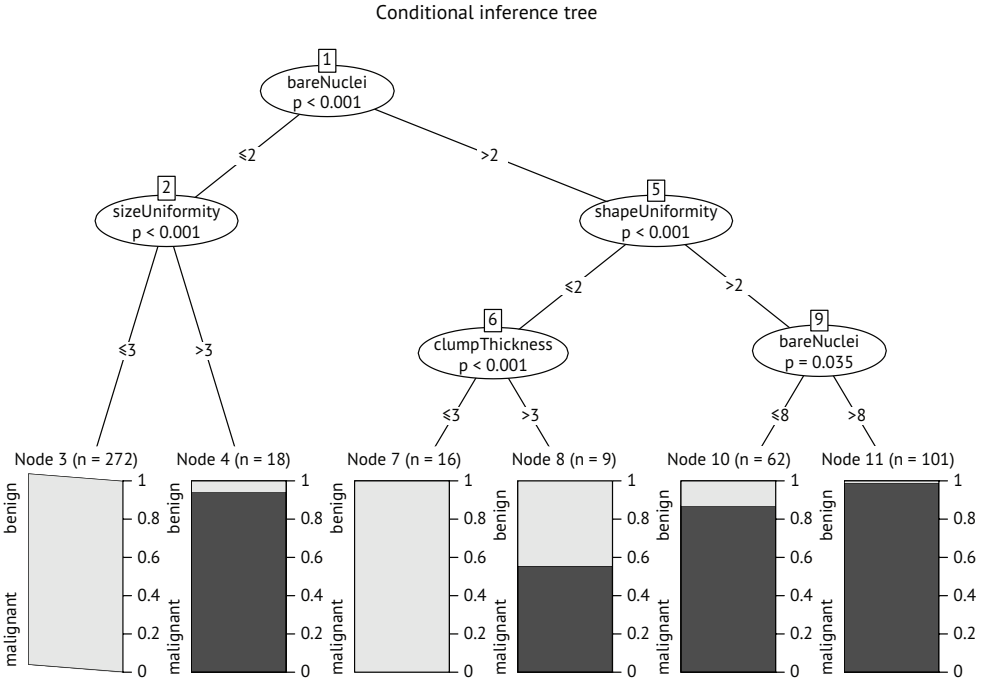


Рис. 17.3. Условное дерево вывода, полученное на основе данных о раке молочной железы

Отображение дерева, полученного функцией rpart(), в виде диаграммы, аналогичной ctree()

Если вы создали классическое дерево решений с помощью `rpart()`, но хотите отобразить результат в виде диаграммы, изображенной на рис. 17.3, то воспользуйтесь пакетом `partykit`. Вызов `plot(as.party(an.rpart.tree))` создаст для вас нужную диаграмму. Например, попробуйте создать диаграмму, как на рис. 17.3, используя объект `dtree.pruned`, созданный кодом в листинге 17.3, и сравните результаты с диаграммой на рис. 17.2.

Традиционные и условные деревья решений могут существенно различаться. В текущем примере они имеют сопоставимую точность (96 % против 97 %), но сами деревья совершенно разные. В следующем разделе мы попробуем создать большое количество деревьев решений и объединить их для классификации результатов анализов.

17.4. Случайные леса

Случайный лес – это ансамблевый подход к обучению с учителем. Суть его состоит в том, чтобы создать несколько прогнозных моделей и объединить результаты для повышения качества класси-

фикации. Подробное введение в случайные леса, написанное Лео Брейманом (Leo Breiman) и Адель Катлер (Adele Cutler), можно найти по адресу: http://mng.bz/7Nul.

Алгоритм создания случайного леса включает выборку случаев и переменных и генерирование большого количества деревьев решений. Каждый случай классифицируется всеми деревьями решений, а в качестве результата используется наиболее распространенная классификация.

Предположим, что N – количество случаев в обучающей выборке, а M – количество переменных. Тогда алгоритм будет выглядеть так:

1. Создать большое количество деревьев решений путем выбора с возвратом N случаев из обучающей выборки.
2. Выбрать $m < M$ переменных в каждом узле, которые будут считаться кандидатами на ветвление в этом узле. Значение m одинаково для всех узлов.
3. Полностью вырастить каждое дерево без усечения (минимальный размер узла равен 1).
4. Конечные узлы связываются с определенным классом на основе наиболее часто встречающихся случаев в этом узле.
5. Классифицировать новые случаи, передавая их всем деревьям и проводя голосование по правилу большинства.

Оценка ошибки вне выборки (out-of-bag, OOB) осуществляется путем классификации случаев, не вошедших в выборку, использовавшуюся при построении деревьев. Как вы увидите далее, случайные леса также поддерживают естественную меру важности переменных.

Случайные леса создаются с помощью функции `randomForest()` из пакета `random-Forest`. Количество деревьев по умолчанию – 500, количество переменных по умолчанию, выбираемых для каждого узла, – `sqrt(M)`, а минимальный размер узла – 1.

В листинге 17.5 показаны код и результаты прогнозирования злокачественности в данных о раке молочной железы.

Листинг 17.5. Случайный лес

```
> library(randomForest)
> set.seed(1234)
> fit.forest <- randomForest(class~., data=train,        (1)
                             importance=TRUE)            (1)
> fit.forest

Call:
 randomForest(formula = class ~ ., data = train, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 2.93%
```

```
Confusion matrix:
          benign malignant class.error
benign       293         9  0.02980132
malignant      5       171  0.02840909

> randomForest::importance(fit.forest, type=2)          (2)

                         MeanDecreaseGini
clumpThickness                   9.794852
sizeUniformity                  58.635963
shapeUniformity                 49.754466
maginalAdhesion                  8.373530
singleEpithelialCellSize        16.814313
bareNuclei                      36.621347
blandChromatin                  25.179804
normalNucleoli                  14.177153
mitosis                          2.015803

> forest.pred <- predict(fit.forest, test)              (3)
> forest.perf <- table(test$class, forest.pred,         (3)
                       dnn=c("Actual", "Predicted"))    (3)
> forest.perf

           Predicted
Actual      benign malignant
  benign       140         2
  malignant      3        60
```

(1) Выращивание леса.

(2) Определение важности переменных.

(3) Классификация новых случаев.

Этот код сначала вызывает функцию `randomForest()`, чтобы вырастить 500 традиционных деревьев решений путем выбора с возвратом 489 наблюдений из обучающей выборки и выбора 3 переменных в каждом узле каждого дерева (1).

Случайные леса поддерживают естественную меру важности переменных, которую можно получить, добавив параметр `importance=TRUE`, и вывести вызовом функции `importance()` (2). Эта функция имеется в обоих пакетах, `rattle` и `randomForest`, но, так как в этом примере мы загрузили оба пакета, в коде используется уточненный вызов `randomForest::importance()`, чтобы гарантировать вызов правильной функции. Мера относительной важности, заданная параметром `type=2`, – это общее уменьшение неоднородности узлов в результате деления по этой переменной, усредненное по всем деревьям. Неоднородность измеряется коэффициентом Джини. `sizeUniformity` – самая важная переменная, а `mitosis` – наименее важная.

В заключение выполняется классификация контрольной выборки с использованием случайного леса и вычисляется точность прогнозов (3). Обратите внимание, что случаи с пропущенными

значениями, присутствующие в контрольной выборке, не классифицируются. Точность прогнозирования (98 %) превосходна.

Пакет `randomForest` реализует создание лесов на основе традиционных деревьев решений. Однако при необходимости можно воспользоваться функцией `cforest()` из пакета `party`, чтобы создать случайный лес деревьев условного вывода. Если переменные-предикторы сильно коррелированы, то случайный лес деревьев условного вывода может дать довольно высокое качество прогнозов.

Случайные леса обычно дают более точные прогнозы, чем другие методы классификации. Кроме того, они могут обрабатывать большие объемы данных (со множеством наблюдений и переменных), неплохо справляются с большим количеством пропущенных данных в обучающей выборке и случаями, когда количество переменных намного превышает количество наблюдений. Значительными преимуществами также являются предоставление коэффициентов ошибок OOB и оценок важности переменных.

Однако есть и свои недостатки! При работе с лесами довольно трудно обобщить правила классификации (еще бы, в этом примере в классификации участвуют 500 деревьев!) и сообщить их другим. Кроме того, для классификации новых случаев необходимо хранить весь лес.

Случайные леса – это модель «черного ящика». Она принимает значения предикторов и выводит точные предсказания, но очень сложно понять, что происходит внутри «ящика» (модели). Мы рассмотрим этот вопрос в разделе 17.7.

Последняя модель классификации, которую мы разберем, – это машина опорных векторов.

17.5. Машины опорных векторов

Машины опорных векторов (Support Vector Machine, SVM) – это группа моделей машинного обучения с учителем, которые можно использовать для классификации и регрессии. Их большая популярность обусловлена отчасти успехами в разработке точных моделей прогнозирования, а отчасти изящным математическим аппаратом, лежащим в основе этого метода. Мы сосредоточимся на использовании SVM для бинарной классификации.

Модели SVM ищут оптимальную гиперплоскость, разделяющую два класса в многомерном пространстве. Гиперплоскость выбирается так, чтобы максимизировать пустую область – *зазор* – между ближайшими точками двух классов. Точки на границе зазора называются *опорными векторами* (они помогают определить границу), а середина зазора – разделяющей гиперплоскостью.

Для N-мерного пространства (т. е. пространства с N переменными-предикторами) оптимальная гиперплоскость (также называемая *линейной поверхностью принятия решений*) имеет размерность

N – 1. Если есть две переменные, поверхность представляет собой линию. Для трех переменных поверхность – плоскость. Для 10 переменных – 9-мерная гиперплоскость. Попытка изобразить такую гиперплоскость вызовет у вас головную боль.

Рассмотрим двумерный пример, показанный на рис. 17.4. Круги и треугольники представляют две группы. *Зазор* представляет расстояние между двумя пунктирными линиями. Точки на пунктирных линиях (закрашенные кружки и треугольники) – опорные векторы. В двумерном случае оптимальной гиперплоскостью является черная линия в середине зазора. В этом идеализированном примере две группы линейно разделимы – линия может полностью разделить две группы без ошибок.

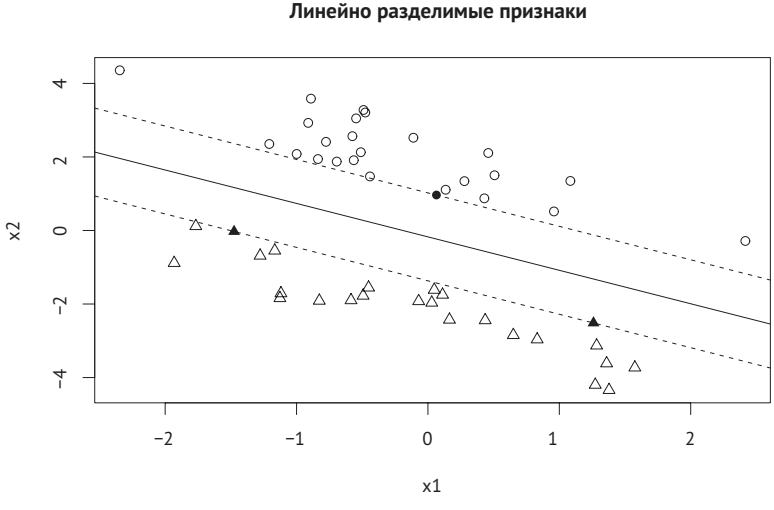


Рис. 17.4. Задача классификации двух групп, когда две группы линейно разделимы. Разделяющая гиперплоскость показана сплошной черной линией. Зазор – это расстояние от разделяющей линии до пунктирных линий с обеих сторон. Закрашенные кружки и треугольники – опорные векторы

Оптимальная гиперплоскость идентифицируется с помощью квадратичной оптимизации зазора, при условии что точки данных на одной стороне дают результат +1, а точки на другой стороне дают результат –1. Если точки данных *почти* разделимы (не все точки находятся по одну или другую сторону), то в оптимизацию добавляется штрафной член для учета ошибок и создаются *мягкие* зазоры.

Но данные могут быть принципиально нелинейными. В примере на рис. 17.5 никакая линия не может правильно разделить кружки и треугольники. Метод SVM используют ядерные функции для преобразования данных в более высокие измерения в надежде, что они станут более разделимыми линейно. Представьте себе преобразование данных на рис. 17.5 такое, что кружки как бы отрыва-

ются от плоскости страницы. Один из способов сделать это – преобразовать двумерные данные в трехмерные, используя формулу

$$(X,Y) \to (X^2, \sqrt{XY}, Y^2) \to (Z_1, Z_2, Z_3).$$

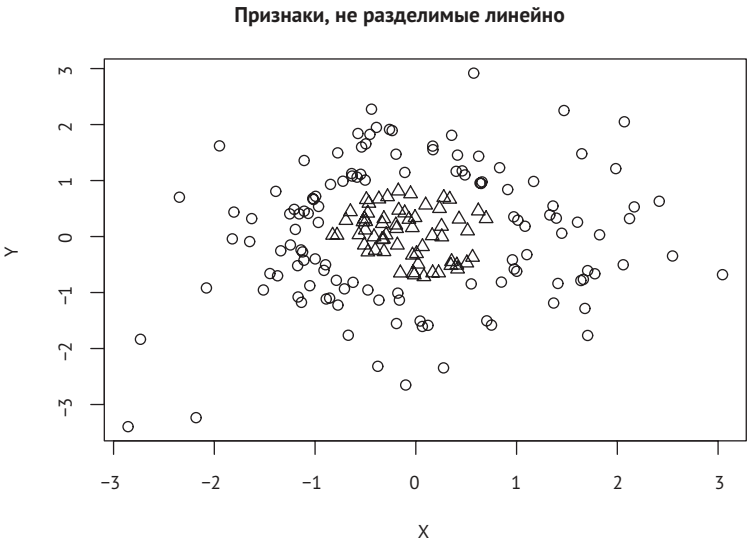


Рис. 17.5. Задача классификации двух групп, когда две группы не разделимы линейно. Группы нельзя разделить гиперплоскостью (прямой)

В таком случае можно отделить треугольники от кружков с помощью жесткого листа бумаги (то есть двумерной плоскости в пространстве, которое теперь является трехмерным).

Математический аппарат SVM сложен, и его обсуждение выходит далеко за рамки этой книги. В публикации Statnikov, Aliferis, Hardin, & Guyon (2011) предлагается ясное и интуитивно понятное представление SVM, содержащее довольно много концептуальных деталей и без запутывающих деталей высшей математики.

Метод SVM доступен в R в виде функции `ksvm()` в пакете `kernlab` и функции `svm()` в пакете `e1071`. Первая более мощная, зато вторая немного проще в использовании. Пример в листинге 17.6 использует последнюю (просто – это хорошо) для создания модели SVM на основе данных о раке молочной железы в штате Висконсин.

Листинг 17.6. Машина опорных векторов

```
> library(e1071)
> set.seed(1234)
> fit.svm <- svm(class~., data=train)
> fit.svm

Call:
```

```
svm(formula = class ~ ., data = train)

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  84

> svm.pred <- predict(fit.svm, test)
> svm.perf <- table(test$class,
                    svm.pred, dnn=c("Actual", "Predicted"))
> svm.perf

            Predicted
Actual      benign malignant
  benign       138         4
  malignant      0        63
```

Поскольку переменные-предикторы с большей дисперсией обычно оказывают большее влияние на модель SVM, функция `svm()` по умолчанию масштабирует все переменные, приводя их к среднему значению, равному 0, и стандартному отклонению, равному 1, перед подгонкой модели. Как видите, точность прогноза (99 %) получилась очень хорошая.

17.5.1. Настройка модели SVM

По умолчанию функция `svm()` использует *радиальную базисную функцию* (Radial Basis Function, RBF) для сопоставления выборок из многомерного пространства. Применение RBF в качестве ядра часто является хорошим выбором, потому что это нелинейное отображение может обрабатывать нелинейные отношения между метками классов и предикторами.

При настройке SVM с ядром RBF на результаты могут повлиять два параметра: *gamma* и *cost*. `gamma` – это параметр ядра, управляющий формой разделяющей гиперплоскости. Большие значения `gamma` обычно приводят к большему числу опорных векторов. Параметр `gamma` также можно рассматривать как контролирующий, насколько широко *простирается* обучающая выборка: чем больше значение, тем дальше простирается выборка. Параметр `gamma` должен быть больше нуля.

Параметр `cost` определяет величину штрафа за допущенные ошибки. Большое значение серьезно наказывает алгоритм за ошибки и приводит к созданию более сложной границы классификации. В обучающей выборке будет меньше ошибочных классификаций, но переобучение может привести к ухудшению предсказательной способности на новых выборках. Меньшие значения дают более плоскую границу классификации, но могут привести к недообучению. Как и `gamma`, параметр `cost` всегда положителен.

По умолчанию функция `svm()` устанавливает параметр `gamma` равным отношению 1/(количество предикторов), а параметр стоимости – равным 1. Однако иногда другие комбинации значений `gamma` и `cost` могут приводить к более эффективным моделям. Можно попробовать подобрать модель SVM, изменяя значения параметров по одному, но поиск по сетке более эффективен. Для этого можно воспользоваться функцией `tune.svm()` и передать ей диапазон значений для каждого параметра. `tune.svm()` подберет модель для каждой комбинации значений и сообщит об их качестве, как показано в примере в листинге 17.7.

Листинг 17.7. Настройка машины опорных векторов RBF

```
> set.seed(1234)
> tuned <- tune.svm(class~., data=train,                    (1)
                    gamma=10^(-6:1),                        (1)
                    cost=10^(-10:10))                       (1)
> tuned                                                     (2)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
  0.01    1

- best performance: 0.03355496

> fit.svm <- svm(class~., data=train, gamma=.01, cost=1)    (3)
> svm.pred <- predict(fit.svm, na.omit(test))               (4)
> svm.perf <- table(na.omit(test)$class,                    (4)
                    svm.pred, dnn=c("Actual", "Predicted")) (4)
> svm.perf                                                  (4)

           Predicted
Actual      benign malignant
  benign       139         3
  malignant      1        62
```

(1) **Диапазоны параметров.**

(2) **Вывод лучшей модели.**

(3) **Подгонка модели с выбранными параметрами.**

(4) **Перекрестная проверка качества классификации.**

Сначала задаются диапазоны параметров гамма и стоимости для подгонки модели SVM с ядром RBF (1). Для `gamma` определяется восемь возможных значений (в диапазоне от 0,000001 до 10) и 21 значение для `cost` (в диапазоне от 0,0000000001 до 100000000). Всего подбирается и сравнивается 168 моделей (8 × 21). Модель

с наименьшей ошибкой при 10-кратной перекрестной проверке на обучающей выборке имеет гамму = 0,01 и стоимость = 1.

Далее выполняется подгонка модели SVM с применением этих значений параметров к обучающей выборке (3). Затем модель используется для прогнозирования результатов в контрольной выборке (4) и отображается количество ошибок. Настройка модели (2) практически не повлияла на количество ошибок. Это неудивительно для данного примера. Значения параметров по умолчанию (`cost = 1`, `gamma = 0.111`) очень похожи на настроенные (`cost = 1`, `gamma = 0.01`). Но чаще настройка параметров SVM приводит к большему выигрышу.

Как уже говорилось, модели SVM пользуются большой популярностью, потому что хорошо работают во многих ситуациях. Они также могут применяться, когда количество переменных намного превышает количество наблюдений. Это добавило им популярности в области биомедицины, где количество переменных, собранных в типичном исследовании экспрессии генов с помощью ДНК-микрочипов, может быть на один или два порядка больше количества доступных случаев.

Один из недостатков SVM, как и при использовании случайных лесов, – сложность интерпретации получившихся правил классификации. Модель SVM тоже, по сути, является черным ящиком. Кроме того, метод SVM плохо подходит для построения моделей на основе больших обучающих выборок, как случайные леса. Но после создания успешной модели классификация новых наблюдений масштабируется достаточно хорошо.

17.6. Выбор лучшего прогностического решения

В разделах 17.4 и 17.5 образцы тонкоигольной аспирации классифицировались как злокачественные или доброкачественные с использованием нескольких методов машинного обучения с учителем. Какой подход оказался наиболее точным? Чтобы ответить на этот вопрос, нужно определить понятие *точности* в контексте бинарной классификации.

В практике наиболее часто используется статистика *точности*, сообщающая, как часто классификатор делает правильный выбор. Несмотря на информативность, одной только точности недостаточно. Для оценки полезности схемы классификации необходима дополнительная информация.

Рассмотрим набор правил классификации людей на шизофреников и не шизофреников. Шизофрения – редкое заболевание, распространенность которого составляет примерно 1 % от общей численности популяции. Если классифицировать всех как не шизофреников, то точность такой классификации составит 99 %. Но это плохой классификатор, потому что он ошибочно классифи-

цирует шизофреников как не шизофреников. В дополнение к точности необходимо получить ответы на следующие вопросы:

- Какой процент шизофреников правильно классифицирован?
- Какой процент не шизофреников правильно классифицирован?
- Если человек классифицирован как шизофреник, то насколько вероятно, что эта классификация верна?
- Если человек классифицирован как не шизофреника то насколько вероятно, что эта классификация верна?

Эти вопросы относятся к чувствительности классификатора, специфичности, положительной предсказательной способности и отрицательной предсказательной способности. Все эти показатели определены в табл. 17.1.

Таблица 17.1. Показатели точности прогнозирования

Показатель	Интерпретация
Чувствительность (sensitivity)	Доля верно классифицированных наблюдений среди всех наблюдений, которые к этому классу отнес классификатор (доля истинно положительных результатов, или полнота)
Специфичность (speci city)	Доля верно классифицированных наблюдений от общего числа наблюдений за пределами класса (насколько часто классификатор *верно не относит* наблюдение к данному классу, доля истинно отрицательных результатов)
Положительная прогностическая ценность	Доля верно классифицированных наблюдений среди всех наблюдений, которые к этому классу отнес классификатор (точность [precision])
Отрицательная прогностическая ценность	Доля верно классифицированных наблюдений среди всех наблюдений, которые к этому классу *не* отнес классификатор
Достоверность (accuracy)	Доля верно идентифицированных наблюдений (также называется ACC)

В листинге 17.8 показана функция для вычисления этих статистик.

Листинг 17.8. Функция оценки точности бинарной классификации

```
performance <- function(table, n=2){
  if(!all(dim(table) == c(2,2)))
      stop("Must be a 2 x 2 table")
  tn = table[1,1]                          1
  fp = table[1,2]                          1
  fn = table[2,1]                          1
  tp = table[2,2]                          1
  sensitivity = tp/(tp+fn)                 2
  specificity = tn/(tn+fp)                 2
  ppp = tp/(tp+fp)                         2
  npp = tn/(tn+fn)                         2
```

```
  hitrate = (tp+tn)/(tp+tn+fp+fn)                                    (2)
  result <- paste("Sensitivity = ", round(sensitivity, n) ,          (3)
      "\nSpecificity = ", round(specificity, n),                     (3)
      "\nPositive Predictive Value = ", round(ppp, n),               (3)
      "\nNegative Predictive Value = ", round(npp, n),               (3)
      "\nAccuracy = ", round(hitrate, n), "\n", sep="")              (3)
  cat(result)                                                        (3)
}
```

(1) Извлечение частот.

(2) Вычисление статистик.

(3) Вывод результатов.

Функция `performance()` принимает таблицу с истинными (строки) и прогнозируемыми результатами (столбцы) и возвращает пять показателей точности. Сначала извлекается количество истинно отрицательных результатов (доброкачественная ткань верно идентифицирована как доброкачественная), ложноположительных результатов (доброкачественная ткань ошибочно идентифицирована как злокачественная), ложноотрицательных (злокачественная ткань ошибочно идентифицирована как доброкачественная) и истинно положительных результатов (злокачественная ткань верно идентифицирована как злокачественная) (1). Затем эти подсчеты используются для вычисления чувствительности (sensitivity), специфичности (specificity), положительной и отрицательной прогностической ценности и достоверности (accuracy) (2). Наконец, результаты форматируются и распечатываются (3).

В листинге 17.9 функция `performance()` применяется к каждому из пяти классификаторов, разработанных в этой главе.

Листинг 17.9. Эффективность классификаторов, обученных на данных о раке молочной железы

```
> performance(logit.perf)
Sensitivity = 0.95
Specificity = 0.99
Positive Predictive Value = 0.97
Negative Predictive Value = 0.98
Accuracy = 0.98

> performance(dtree.perf)
Sensitivity = 0.95
Specificity = 0.96
Positive Predictive Value = 0.91
Negative Predictive Value = 0.98
Accuracy = 0.96

> performance(ctree.perf)
Sensitivity = 0.97
Specificity = 0.97
```

```
Positive Predictive Value = 0.94
Negative Predictive Value = 0.99
Accuracy = 0.97

> performance(forest.perf)
Sensitivity = 0.95
Specificity = 0.99
Positive Predictive Value = 0.97
Negative Predictive Value = 0.98
Accuracy = 0.98

> performance(svm.perf)
Sensitivity = 0.98
Specificity = 0.98
Positive Predictive Value = 0.95
Negative Predictive Value = 0.99
Accuracy = 0.98
```

Все представленные классификаторы (логистическая регрессия, традиционное дерево решений, дерево условного вывода, случайный лес и метод опорных векторов) показали очень хорошие результаты по каждому показателю точности. Но так будет не всегда!

В этом конкретном случае награда достается модели SVM (хотя различия настолько малы, что они могут быть случайными). Модель SVM верно идентифицировала 98 % злокачественных (чувствительность) и 98 % доброкачественных новообразований (специфичность), а общий процент правильных классификаций составил 98 % (достоверность). Диагноз злокачественного новообразования был верным в 95 % всех случаев (положительная прогностическая ценность), а диагноз доброкачественной опухоли – в 99 % случаев (отрицательная прогностическая ценность). Для диагностики рака особенно важна специфичность (доля злокачественных образцов, правильно идентифицированных как злокачественные).

Хотя это выходит за рамки данной главы, тем не менее отмечу, что часто можно улучшить систему классификации, поменяв специфичность на чувствительность, и наоборот. В модели логистической регрессии для оценки вероятности принадлежности случая к группе злокачественных новообразований использовался метод `predict()`. Если вероятность превышала 0,5, то случай относился к этой группе. Значение 0,5 называется *пороговым*, или *отсекающим*, значением. Если изменить этот порог, то можно повысить чувствительность модели за счет ее специфичности. Функция `predict()` также может генерировать вероятности для деревьев решений, случайных лесов и SVM (но синтаксис ее вызова зависит от метода).

Влияние изменения порогового значения обычно оценивается с использованием *характеристической кривой обнаружения* (Receiver Operating Characteristic, ROC). Кривая ROC показывает зависимость чувствительности от специфичности для диапазона по-

роговых значений. Глядя на нее, можно выбрать порог с лучшим балансом чувствительности и специфичности для данной задачи. Многие пакеты R генерируют кривые ROC, включая ROCR и pROC. Аналитические функции в этих пакетах могут помочь выбрать оптимальные пороговые значения для данного сценария или сравнить кривые ROC, полученные с помощью различных алгоритмов классификации, чтобы выбрать наиболее эффективный. Узнать больше можно в публикации Kuhn & Johnson (2013). Более подробное обсуждение предлагается в публикации Fawcett (2005).

17.7. Интерпретация прогнозов черного ящика

Модели классификации используются для принятия реальных решений, которые могут иметь серьезные последствия для человека. Представьте, что вы подали заявку на получение банковского кредита, но вам отказали, и вы хотите понять, почему. Если решение было основано на модели логистической регрессии или дерева классификации, то причину можно определить, взглянув на коэффициенты модели в первом случае или на дерево решений во втором случае. Но что, если бы классификация была выполнена с применением модели случайного леса, опорных векторов или искусственной нейронной сети? До недавнего времени на вопрос обычно отвечали: «потому что так сказал компьютер», – что является очень неудовлетворительным ответом!

В последние годы стали появляться средства и методы интерпретации моделей черного ящика под общим названием *объяснимый искусственный интеллект* (Explainable Artificial Intelligence, XAI, http://ema.drwhy.ai). Цель XAI – помочь понять, как работают модели черного ящика в целом (глобальное понимание) и при составлении индивидуальных прогнозов (локальное понимание).

Возьмем для примера пациента по имени Алекс (Alex). Биопсия показала следующие результаты:

```
bareNuclei = 9,
sizeUniformity = 1,
shapeUniformity = 1,
blandChromatin = 7,
maginalAdhesion = 1,
mitosis = 3,
normalNucleoli = 3,
clumpThickness = 6,
singleEpithelialCellSize = 3
```

Для этих значений модель случайного леса, разработанная в разделе 17.4, дала прогноз: «злокачественная опухоль» (с вероятностью 0,658). В оставшейся части этого раздела мы попробуем применить методы XAI, чтобы понять, почему модель случайного леса

поставила такой диагноз. В примерах будет использоваться DALEX, поэтому обязательно установите его (`install.packages("DALEX")`), прежде чем продолжить.

17.7.1. Графики разбивки

Наша цель – разделить предикторы по их вкладу в окончательную классификацию (злокачественная опухоль с вероятностью 0,658). Для этого используем значения *разбивки*.

1. Используя обучающую выборку, вычислим среднее прогнозируемое значение (в данном случае вероятность злокачественности) по всем наблюдениям (0,365). Назовем его перехватом.
2. Для всех наблюдений, где `bareNuclei = 9`, вычислим средний прогноз (0,544). Вклад `bareNuclei` в этом случае составляет 0,544 – 0365 = +0,179.
3. Для всех наблюдений, где `bareNuclei = 9` *и* `sizeUniformity = 1`, вычислим средний прогноз (0,476). Вклад `sizeUniformity` составляет 0,476 – 0,544 = –0,068.
4. Для всех наблюдений, где `bareNuclei = 9`, *и* `sizeUniformity = 1`, *и* `shapeUniformity = 1`, вычислим средний прогноз (0,42). Вклад `shapeUniformity` составляет –0,05.
5. Продолжаем до тех пор, пока не будут включены все значения предикторов. Сумма вкладов отдельных предикторов составляет общий прогноз модели для этого случая. Положительный вклад увеличивает вероятность диагноза злокачественного новообразования. Отрицательный – снижает. Величина вклада оценивает влияние переменной на окончательный прогноз.

В листинге 17.10 представлен код вычисления значений разбивки, а на рис. 17.6 показан график разбивки.

Листинг 17.10. Создание графика разбивки для интерпретации прогноза черного ящика

```
> library(DALEX)

> alex <- data.frame(                                    1
        bareNuclei = 9,                                  1
        sizeUniformity = 1,                              1
        shapeUniformity = 1,                             1
        blandChromatin = 7,                              1
        maginalAdhesion = 1,                             1
        mitosis = 3,                                     1
        normalNucleoli = 3,                              1
        clumpThickness = 6,                              1
        singleEpithelialCellSize = 3                     1
    )
```

```
> predict(fit.forest, alex, type="prob")                                    (2)

  benign malignant
1  0.278     0.722

set.seed(1234)                                                              (3)
explainer_rf_malignant <-                                                   (3)
  explain(fit.forest, data = train,  y = train$class == "malignant",        (3)
  predict_function = function(m, x) predict(m, x, type = "prob")[,2])       (3)

rf_pparts <- predict_parts(explainer=explainer_rf_malignant,                (4)
                           new_observation = alex,                          (4)
                           type = "break_down")                             (4)

plot(rf_pparts)                                                             (4)
```

(1) **Копирование наблюдения.**

(2) **Прогноз для этого наблюдения.**

(3) **Сборка объясняющего объекта.**

(4) **Создание графика разбивки.**

После загрузки пакета `DALEX` интересующий случай копируется в таблицу данных (1). Функция `predict()` вычисляет прогноз для этого случая, используя случайный лес. Поскольку `type="prob"`, в прогнозе возвращаются вероятности доброкачественного и злокачественного исходов (2). Затем создается объясняющий объект `explainer` (3). Конструктору объекта передаются модель случайного леса, обучающие данные, переменная с результатом и функция, используемая для прогнозирования. Здесь указывается, что нам нужно предсказать вероятность злокачественного исхода. Использование `[,2]` в функции прогнозирования возвращает только вероятность злокачественного новообразования. В заключение создается график разбивки на основе объекта `explainer` и наблюдения, которое нужно объяснить (4).

На рис. 17.6 можно видеть, что для Алекса `bareNuclei = 9` и `clumpThickness = 6` вносят наибольший вклад в диагностику злокачественности новообразования. Оценки `sizeUniformity = 1` и `shapeUniformity = 1` снижают вероятность злокачественности. Принимая во внимание все девять предикторов, вероятность злокачественности новообразования составила 0,658. Поскольку 0,658 > 0,50, модель случайного леса классифицировала результаты анализов Алекса как злокачественную опухоль.

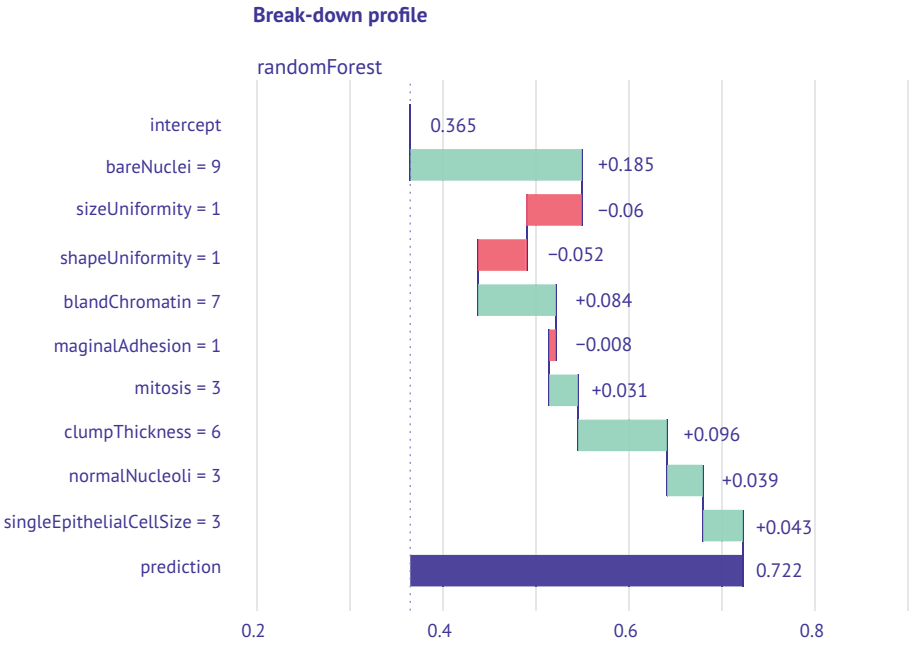


Рис. 17.6. График разбивки прогноза, полученного с помощью случайного леса для одного человека. Средняя прогнозируемая вероятность злокачественности равна 0,365, а с учетом оценок для этого человека прогнозируемая вероятность составляет 0,722. Поскольку эта вероятность больше 0,5, то прогнозируется злокачественный характер новообразования. Также можно увидеть индивидуальный вклад каждого предиктора для этого человека. Зеленые столбцы представляют положительный вклад в прогнозируемый результат, а красные столбцы – отрицательный

17.7.2. График значений Шепли

Графики разбивки нередко очень помогают интерпретировать причины данного прогноза, полученного из модели черного ящика. Но имейте в виду, что разбивка зависит от порядка. Если между переменными-предикторами имеются взаимозависимости, то для разных порядков переменных можно получить разные результаты.

Аддитивные объяснения Шепли, или *SHAP*, решают проблему зависимости от порядка, вычисляя вклад разбивки для нескольких порядков переменных-предикторов и усредняя результаты для каждой переменной. Продолжая пример из листинга 17.10, следующий код:

```
set.seed(1234)
rf_pparts = predict_parts(explainer = explainer_rf_malignant,
                          new_observation = alex,
                          type = "shap")
plot(rf_pparts)
```

выведет диаграмму, изображенную на рис. 17.7.

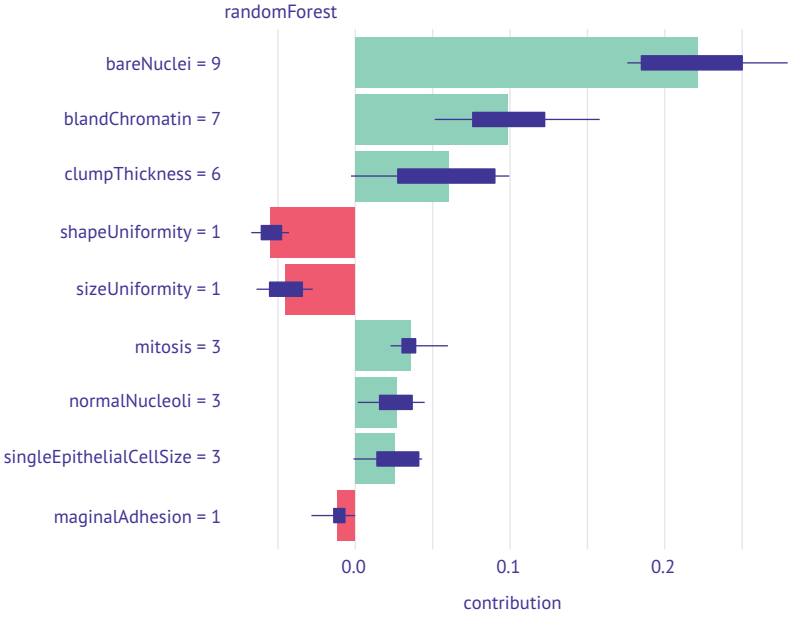


Рис. 17.7. График значений SHAP для предсказания, полученного из модели случайного леса для одного человека

Коробчатые диаграммы показывают диапазон вкладов разбивки для различных порядков переменных. Столбики представляют среднее значение разбивки. Большие столбики соответствуют большому влиянию на прогноз для этого человека. Положительные (зеленые) столбики соответствуют вкладу в злокачественный диагноз, а отрицательные (красные) – в доброкачественный.

Обсуждаемые методы XAI не зависят от модели – их можно использовать с любым подходом к машинному обучению, включая те, что рассматриваются в этой главе. Пакет `DALEX` предлагает множество других подобных статистических инструментов, и с ними определенно стоит познакомиться поближе.

17.8. Дополнительная информация

Создание прогнозной модели с использованием методов машинного обучения – сложный итеративный процесс, в общем случае включающий следующие шаги:

1. *Разделение данных.* Доступные данные делятся на обучающую и контрольную выборки.
2. *Предварительная обработка данных.* Выбираются и, возможно, преобразуются переменные-предикторы. Например, метод опорных векторов часто лучше всего работает со стандартизированными предикторами. Сильно коррелированные перемен-

ные можно объединить в составные переменные или исключить из анализа. Отсутствующие значения должны быть заменены или удалены.

3 *Построение модели.* Создание нескольких возможных моделей прогнозирования. Большинство из них будут иметь гиперпараметры, требующие настройки. Гиперпараметры – это параметры модели, управляющие процессом обучения и подбираемые методом проб и ошибок. Параметр сложности в деревьях классификации, количество переменных-кандидатов для разбиения в случайных лесах, а также параметры `cost` и `gamma` в методе опорных векторов – все это примеры таких гиперпараметров. Вы должны попробовать разные значения для гиперпараметров и выбрать те, которые приводят к наиболее надежным моделям.

4 *Сравнение моделей.* После подгонки (обучения) модели сравниваются, и выбирается окончательная модель.

5 *Выпуск модели.* После выбора окончательной модели ее можно использовать для прогнозирования будущего. Поскольку окружение может меняться, важно постоянно оценивать эффективность модели с течением времени.

Задача еще более усложняется тем фактом, что методы машинного обучения реализованы во множестве разных пакетов и используют разный синтаксис. Для упрощения рабочего процесса были разработаны *комплексные пакеты*, или *метапакеты*, служащие обертками для других пакетов и обеспечивающие упрощенный и согласованный интерфейс для создания прогнозных моделей. Изучение одного из этих подходов может значительно упростить задачу построения эффективной прогностической системы.

Три самых популярных подхода – пакет `Caret` и фреймворки `mlr3` и `tidymodels` (фреймворки состоят из нескольких взаимосвязанных пакетов). Пакет `Caret` (http://topepo.github.io/caret), возможно, является наиболее зрелым пакетом и поддерживает более 230 алгоритмов машинного обучения. Фреймворк mlr3 (https://mlr3.mlr-org.com/) хорошо структурирован и понравится объектно-ориентированным программистам. Фреймворк tidymodels (https://www.tidymodels.org/) – самый новый и, пожалуй, самый гибкий. Он создавался теми же разработчиками, что отвечали за пакет `Caret`, поэтому я подозреваю, что со временем он заменит пакет `Caret`. Если вы собираетесь использовать R для машинного обучения, я предлагаю выбрать один из этих фреймворков и подробно изучить его. Какой из них выбрать, зависит от ваших личных предпочтений.

Чтобы узнать больше о функциях R, поддерживающих прогнозирование и классификацию, можно обратиться к обзору задач по машинному и статистическому обучению «CRAN Task View for Ma-

chine Learning and Statistical Learning» (http://mng.bz/I1Lm). В числе других источников информации можно порекомендовать Kuhn & Johnson (2013), Forte (2015), Lanz (2015) и Torgo (2017).

Итоги

- Предсказание бинарного категориального результата (пройдено / не пройдено, успешно/неудачно, жив/мертв, доброкачественное/злокачественное) – распространенная задача в науке о данных, и такие прогнозы могут иметь реальные последствия для людей.
- Прогнозные модели могут быть относительно простыми (логистическая регрессия, деревья классификации) и чрезвычайно сложными (случайные леса, машины опорных векторов, искусственные нейронные сети).
- Модели часто имеют гиперпараметры (параметры, управляющие процессом обучения), которые необходимо настраивать методом проб и ошибок.
- Эффективность прогнозных моделей оценивается с использованием таких показателей, как достоверность (accuracy), чувствительность, специфичность, положительная прогностическая сила и отрицательная прогностическая сила.
- В последние годы разработаны новые методы исследования очень сложных моделей черного ящика.
- Процесс разработки прогнозной модели состоит из множества повторяющихся шагов, которые можно упростить, используя комплексный пакет или фреймворк, такие как `Caret`, `mlr2` или `tidymodels`.

18 Продвинутые методы работы с пропущенными данными

В этой главе:

- идентификация пропущенных данных;
- визуализация закономерностей в пропущенных данных;
- удаление пропущенных значений;
- восстановление пропущенных значений.

В предыдущих главах мы занимались анализом полных наборов данных (без пропущенных значений). Это упрощало описание статистических и графических методов, но в реальном мире пропущенные данные неизбежны.

В некотором смысле влияние пропущенных данных – это то, чего каждый из нас хочет избежать. В книгах по статистике эта тема может вообще не затрагиваться, или же ее обсуждение сводится к нескольким абзацам. В статистических пакетах реализованы не всегда оптимальные методы работы с пропущенными данными. Даже учитывая, что обычно анализ данных (по крайней мере, в социологии) всегда

включает работу с пропущенными наблюдениями, эта тема редко обсуждается при описании методов и результатов в научных статьях. Учитывая, насколько часто встречаются пропущенные данные и насколько сильно их наличие может сделать несостоятельными результаты исследований, можно сказать, что эта тема получила недостаточно внимания за пределами специализированных книг и курсов.

Данные могут отсутствовать по многим причинам. Участники анкетирования могут забыть ответить на один или несколько вопросов, отказаться отвечать на щекотливые вопросы или же утомиться и не закончить заполнение длинной анкеты. Испытуемые могут не выполнить инструкции или преждевременно выбыть из исследования. Записывающее оборудование может выйти из строя, подключение к интернету – пропасть, а также данные могут быть неправильно закодированы. Наличие пропущенных данных даже может быть запланировано. Например, для увеличения эффективности исследования или снижения расходов вы можете решить не собирать полные данные обо всех участниках исследования. Наконец, данные могут быть потеряны по причинам, о которых вы никогда не узнаете.

К сожалению, большинство статистических методов рассчитаны на работу с полными матрицами, векторами и таблицами данных. В большинстве случаев вам придется удалить пропущенные данные, перед тем как искать ответы на вопросы, побудившие вас начать исследование. От пропущенных данных можно избавиться, (1) удаляя наблюдения с такими данными или (2) заменяя пропущенные данные подходящими расчетными значениями. В любом случае, вы получите набор данных без пропущенных значений.

В этой главе мы рассмотрим как традиционные, так и новые подходы к работе с пропущенными данными. Мы будем в основном работать с пакетами `VIM`, `mice` и `missForest`. Установить оба этих пакета можно вызовом `install.packages(c("VIM", "mice", "missForest"))`.

Чтобы сделать обсуждение более интересным, мы рассмотрим набор данных о сне млекопитающих (`sleep`) из пакета `VIM` (не перепутайте его с набором данных `sleep`, описывающим влияние лекарств на сон, из базовой версии R). Данные описаны в работе Allison & Chichetti (1976), которая посвящена взаимосвязям между сном, экологией и морфологией 62 видов млекопитающих. Авторы исследовали, почему необходимая продолжительность сна различается от вида к виду. Параметры сна служили зависимыми переменными, а экологические и морфологические характеристики – независимыми переменными.

К параметрам сна относились продолжительность сна со сновидениями (`Dream`) и без сновидений (`NonD`), а также их сумма (`Sleep`). Морфологические характеристики – это вес тела в килограммах (`BodyWgt`), вес мозга в граммах (`BrainWgt`), продолжительность жиз-

ни в годах (`Span`) и продолжительность беременности в днях (`Gest`). Экологические характеристики – пресс хищников (`Pred`), степень уязвимости во время сна (`Exp`) и общая степень опасности, которой подвергается животное (`Danger`). Экологические характеристики оценивались по пятибалльной шкале от 1 (низкий) до 5 (высокий).

В оригинальной статье анализ ограничивался только видами, для которых имелись полные данные. Мы же пойдем дальше и проанализируем все 62 вида, используя прием множественного восстановления пропущенных данных.

18.1. Этапы работы с пропущенными данными

Новички в области исследования пропущенных значений обнаружат обескураживающее многообразие подходов, критических отзывов и методологий. Классическое руководство в этой области – это публикация Little & Rubin (2002). Прекрасные доступные обзоры написаны Allison (2001), Schafer & Graham (2002), Enders (2010) и Schlomer, Bauman & Card (2010). Типичный алгоритм работы с пропущенными данными, как правило, включает следующие этапы:

1. выявить пропущенные данные;
2. исследовать причины отсутствия данных;
3. удалить наблюдения с пропущенными значениями или заменить пропущенные данные подходящими расчетными значениями.

К сожалению, идентификация пропущенных данных – это единственный простой этап. Узнать, почему данные пропущены, можно, только если полностью понимать процесс их появления. Решить, как поступить с пропущенными данными, можно, если оценить, какие методы дадут наиболее надежный и аккуратный результат.

Классификация типов пропущенных данных

В статистике обычно выделяют три типа пропущенных данных. Они, как правило, описываются в терминах вероятностей, но идеи, лежащие в основе выделения этих типов, очевидны. Мы рассмотрим их на примере продолжительности сна со сновидениями из набора данных `sleep` (где для 12 животных есть пропущенные значения).

- *Полностью случайный пропуск. Если наличие пропущенных значений в переменной не зависит от значений любой другой наблюдаемой или ненаблюдаемой переменной, тогда данные являются отсутствующими полностью случайно. Если бы пропуск данных по продолжительности сна со сновидениями не был бы ничем обусловлен, то данные подходили бы под эту категорию. Отметим, что если пропуски во всех переменных полностью случайны, то наблюдения без пропусков – это просто случайная выборка из общего массива данных.*

- *Случайный пропуск. Если наличие пропущенных значений в переменной зависит от других переменных, но не от самих неотмеченных значений, то данные являются отсутствующими случайно. Например, если пропуски значений продолжительности сна со сновидениями выше у животных с меньшим весом тела (возможно, потому, что за мелкими животными сложнее наблюдать) и если пропуски не зависят от продолжительности сновидений, то пропуски можно назвать случайными. В данном случае наличие или отсутствие данных по продолжительности сна со сновидениями будет случайным, если избавиться от влияния веса животного.*
- *Неслучайный пропуск. Под эту категорию попадают пропущенные значения, которые не относятся к первым двум категориям. Например, если животные с меньшей продолжительностью сновидений с большей вероятностью имеют пропущенные значения этой величины (возможно, из-за трудностей регистрации более кратких событий), пропуски будут неслучайными.*

Большинство приемов обработки пропущенных данных предполагают, что такие данные являются полностью случайно пропущенными или случайно пропущенными. В этом случае можно проигнорировать механизм появления пропущенных данных и (после их замены или удаления) напрямую моделировать интересующие отношения.

Данные, которые пропущены неслучайно, сложнее поддаются анализу. Когда данные пропущены неслучайно, необходимо смоделировать механизмы, породившие пропущенные значения, а также интересующие отношения. (Существующие подходы к анализу неслучайно пропущенных данных включают использование моделей отбора и комбинирование шаблонов. Анализ неслучайно пропущенных данных невероятно сложен, и его обсуждение выходит за рамки этой книги.)

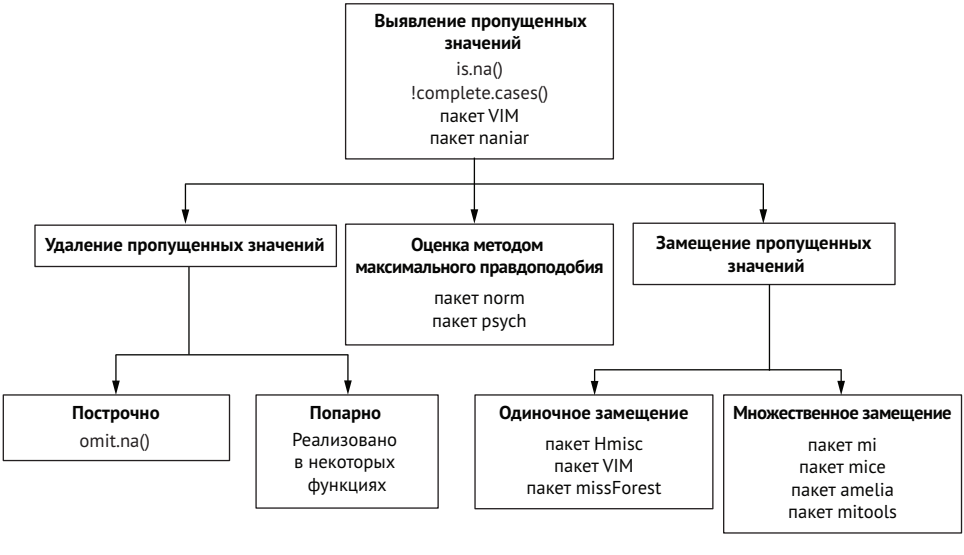


Рис. 18.1. Методы, используемые для работы с пропущенными данными, и пакеты R, в которых реализованы эти методы

Существует *много* методов для работы с пропущенными данными, и нет гарантии, что они дадут один и тот же результат. На рис. 18.1 показано все разнообразие методов, которые используются для работы с пропущенными данными, наряду с пакетами, в которых реализованы эти методы.

Для полного обзора методов работы с пропущенными данными понадобилась бы отдельная книга (отличным примером может служить Enders [2010]). В этой главе мы рассмотрим методы изучения закономерностей в пропущенных данных и сосредоточимся на четырех наиболее распространенных методах работы с неполными наблюдениями: вычислительный подход, удаление неполных наблюдений, одиночное и множественное восстановление пропущенных данных. Мы закончим главу кратким обсуждением прочих методов, включая те, которые могут пригодиться в отдельных ситуациях.

18.2. Идентификация пропущенных значений

Для начала повторим сведения, изложенные в разделе 3.5, и будем отталкиваться от них. В R пропущенные значения обозначаются как `NA` (not available – нет в наличии), а недопустимые значения – как `NaN` (not a number – не число). В дополнение символы `Inf` и `-Inf` обозначают плюс и минус бесконечность соответственно. Для идентификации пропущенных, недопустимых и бесконечных значений используются функции `is.na()`, `is.nan()` и `is.infinite()` соответственно. Каждая из них возвращает значения `TRUE` (истина) и `FALSE` (ложь). Примеры приведены в табл. 18.1.

Таблица 18.1. Примеры значений, возвращаемых функциями is.na(), is.nan() и is.infi nite()

x	is.na()	is.nan()	is.infinite()
`x <- NA`	TRUE	FALSE	FALSE
`x <- 0 / 0`	TRUE	TRUE	FALSE
`x <- 1 / 0`	FALSE	FALSE	TRUE

Эти функции возвращают объект того же размера, что и исходный, но при этом каждый элемент заменяется значением `TRUE`, если это элемент проверяемого типа, и `FALSE` – в противном случае. К примеру, пусть `y <- c(1, 2, 3, NA)`. Тогда функция `is.na(y)` вернет вектор `c(FALSE, FALSE, FALSE, TRUE)`.

Функцию `complete.cases()` можно использовать для идентификации строк в матрице или таблице данных, которые не содержат пропущенных значений. Эта функция возвращает логический вектор со значениями `TRUE` для всех полных строк и `FALSE` для строк с одним и более пропущенными значениями.

Давайте применим эту функцию к нашему набору данных:

```
# загрузить набор данных
data(sleep, package="VIM")

# вывести строки без пропущенных значений
sleep[complete.cases(sleep),]

# вывести строки, в которых хотя бы одно значение пропущено
sleep[!complete.cases(sleep),]
```

Изучение результатов показывает, что 42 строки не имеют пропущенных значений, а в 20 строках есть хотя бы одно пропущенное значение.

Поскольку логические значения `TRUE` и `FALSE` эквивалентны числам `1` и `0`, для получения полезной информации о пропущенных данных можно использовать функции `sum()` и `mean()`. Например:

```
> sum(is.na(sleep$Dream))
[1] 12
> mean(is.na(sleep$Dream))
[1] 0.19
> mean(!complete.cases(sleep))
[1] 0.32
```

Полученный результат свидетельствует о том, что в переменной `Dream` пропущено 12 значений. Пропущенные значения этой переменной содержатся в 19 % строк, а во всем наборе данных 32 % строк имеют хотя бы одно пропущенное значение.

Выявляя пропущенные значения, важно помнить две вещи. Во-первых, функция `complete.cases()` «воспринимает» как отсутствующие только значения `NA` и `NaN`. Бесконечные значения (`Inf` и `-Inf`) пропущенными не считаются. Во-вторых, упомянутые выше функции можно использовать лишь для идентификации пропущенных значений в объектах R. Логические выражения, такие как `myvar == NA`, никогда не вернут `TRUE`.

Теперь, зная, как идентифицировать пропущенные значения, рассмотрим способы исследования возможной структуры пропущенных данных.

18.3. Исследование структуры пропущенных данных

Перед тем как решить, что делать с пропущенными данными, необходимо выяснить, в каких переменных они содержатся, в каких объемах и комбинациях. В этом разделе мы рассмотрим табличные, графические и корреляционные методы исследования структуры пропущенных значений. С их помощью можно понять, почему данные отсутствуют, и скорректировать дальнейший ход статистического анализа.

18.3.1. Представление пропущенных значений в виде таблицы

Вы уже видели простейший подход к идентификации пропущенных значений. Для этого можно использовать описанную в разделе 18.2 функцию `complete.cases()` и с ее помощью создать список полных строк или, наоборот, строк, имеющих хотя бы одно пропущенное значение, но с увеличением объема данных этот подход становится не особенно привлекательным. В таком случае можно обратиться к другим функциям R.

Функция `md.pattern()` из пакета mice представляет информацию о пропущенных значениях в табличной форме. В листинге 18.1 приводится результат применения этой функции к набору данных `sleep`:

Листинг 18.1. Получение информации о пропущенных значениях с помощью md.pattern()

```
> library(mice)
> data(sleep, package="VIM")
> md.pattern(sleep, rotate.names=TRUE)
   BodyWgt BrainWgt Pred Exp Danger Sleep Span Gest Dream NonD
42       1        1    1   1      1     1    1    1     1    1  0
 2       1        1    1   1      1     1    0    1     1    1  1
 3       1        1    1   1      1     1    1    0     1    1  1
 9       1        1    1   1      1     1    1    1     0    0  2
 2       1        1    1   1      1     0    1    1     1    0  2
 1       1        1    1   1      1     1    0    0     1    1  2
 2       1        1    1   1      1     0    1    1     0    0  3
 1       1        1    1   1      1     1    0    1     0    0  3
         0        0    0   0      0     4    4    4    12   14 38
```

Единицы и нули в ячейках таблицы дают представление о структуре пропущенных значений, `0` обозначает пропущенное значение для данной переменной, а `1` – имеющееся значение. Первая строка содержит информацию *о полных наблюдениях* (все значения в ячейках равны `1`). Вторая строка показывает число наблюдений, в которых *нет пропущенных значений во всех переменных, кроме* `Span`. В первом столбце приведено число наблюдений с данным типом структуры пропущенных данных, а последний столбец содержит данные о числе столбцов с пропущенными значениями для каждого типа структуры. В этом примере видно, что есть 42 наблюдения без пропущенных данных и два наблюдения, в которых пропущены только значения переменной `Span`. В девяти наблюдениях пропущены значения переменных `NonD` и `Dream`. Всего в наборе данных содержится (42 × 0) + (2 × 1) + ... + (1 × 3) = 38 пропущенных значений. В последней строке таблицы приведено общее число пропущенных значений для каждой переменной.

Функция `md.pattern()` выводит довольно компактное табличное представление результатов, но часто удобнее исследовать

закономерности визуально. На рис. 18.2 каждая строка представляет закономерность. Темно-синий цвет соответствует присутствующим значениям, а светлый – отсутствующим. Цифры слева указывают количество наблюдений в шаблоне, числа справа – количество переменных с пропущенными значениями, а числа внизу – количество пропущенных значений в каждой переменной.

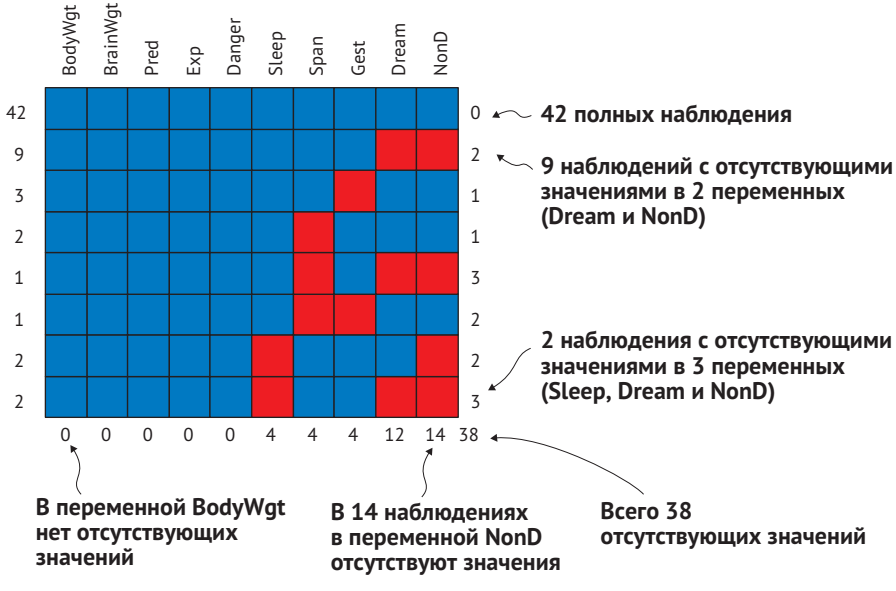


Рис. 18.2. Закономерности отсутствующих значений, полученные с помощью функции md.pattern(). Каждая строка представляет шаблон (закономерность) отсутствующих (красный цвет) и присутствующих (темно-синий цвет) данных

В пакете `VIM` имеется множество функций для визуализации структуры пропущенных данных. В этом разделе мы познакомимся с некоторыми из них, включая `aggr()`, `matrixplot()` и `marginplot()`.

Функция `aggr()` отображает диаграмму с числом наблюдений для каждой отдельной переменной и каждой комбинации переменных. Например, следующий код:

```
library("VIM")
aggr(sleep, prop=FALSE, numbers=TRUE)
```

отобразит диаграмму, изображенную на рис. 18.3.

Как видите, больше всего пропущено значений в переменной `NonD` (14), и есть два наблюдения, не имеющих значений в переменных `NonD`, `Dream` и `Sleep`. 42 наблюдения имеют полные данные.

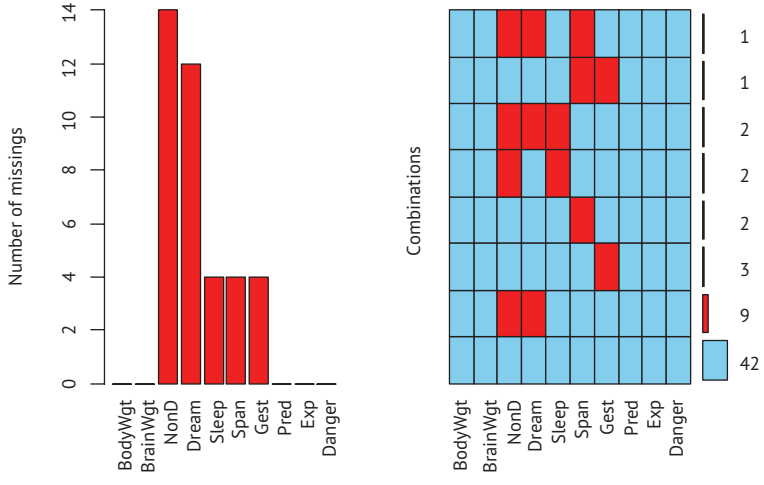


Рис. 18.3. Функция aggr() отображает в графическом виде структуру пропущенных значений в наборе данных sleep

Вызов `aggr(sleep, prop=TRUE, numbers=TRUE)` создаст такую же диаграмму, только в ней вместо чисел будут указаны проценты. Параметр `numbers=FALSE` (по умолчанию) подавляет вывод числовых меток. Обратите внимание, что с увеличением количества переменных в наборе данных метки на диаграмме могут накладываться друг на друга. Эту проблему можно исправить, вручную уменьшив размер меток. Добавив параметры `cex.lab=`*n*, `cex.axis=`*n* и `cex.number=`*n* (где *n* – число меньше 1), можно уменьшить размер меток на осях, имена переменных и числа соответственно.

Функция `matrixplot()` отображает данные по наблюдениям. Диаграмма, полученная вызовом `matrixplot(sleep, sort="BodyWgt")`, представлена на рис. 18.4. Здесь числовые данные нормализованы так, что они все находятся в интервале [0, 1] и представлены оттенками серого. Чем светлее оттенок, тем более низкому значению он соответствует. По умолчанию пропущенные значения обозначены красным. Обратите внимание, что на рис. 18.4 красный цвет был заменен штриховкой вручную, чтобы отсутствующие значения были хорошо видны в черно-белой книге. Строки на рис. 18.4 отсортированы в порядке убывания значений переменной `BodyWgt`.

Функция `marginplot()` создает диаграмму рассеяния для двух переменных, где информация о пропущенных значениях представлена на полях. Рассмотрим связь между длительностью сна со сновидениями и продолжительностью беременности. Инструкция

```
marginplot(sleep[c("Gest","Dream")], pch=20,
           col=c("darkgray", "red", "blue"))
```

создает диаграмму, представленную на рис. 18.5. Необязательные параметры `pch` и `col` определяют тип и цвет отображаемых символов.

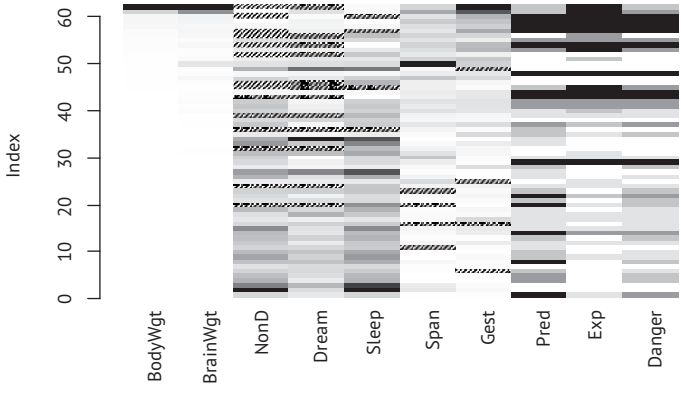


Рис. 18.4. Матричная диаграмма присутствующих и отсутствующих значений в каждом наблюдении в наборе данных sleep. Матрица отсортирована по значениям переменной BodyWgt

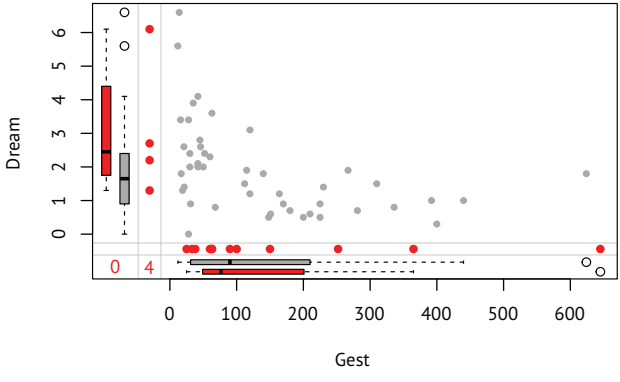


Рис. 18.5. Диаграмма рассеяния длительности сна со сновидениями и продолжительности беременности с информацией о пропущенных значениях на полях

В центре на рис. 18.5 представлена диаграмма рассеяния для переменных `Gest` и `Dream` (на основании наблюдений, где эти две переменные имеют значения). В левом поле расположены диаграммы размахов значений переменной `Dream`, которым соответствуют имеющиеся значения переменной `Gest` (темно-серый) и отсутствующие (красный). (Обратите внимание, что в черно-белой книге красный цвет выглядит темнее.) Четыре красные точки соответствуют значениям переменной `Dream` в четырех наблюдениях, где значения переменной `Gest` отсутствуют. В нижнем поле эти две переменные «поменялись местами». Видно, что существует отрицательная связь между продолжительностями сна и беременности и что сведения о продолжительности беременности, как правило, отсутствуют у животных с более длительным сном. Число наблюдений с пропущенными значениями в обеих переменных дано синим цветом на пересечении полей (в нижнем левом углу).

Пакет VIM позволяет построить множество диаграмм, которые помогут понять роль пропущенных значений в наборе данных и с которыми стоит познакомиться, в том числе функции для создания диаграмм рассеяния, диаграмм размахов, гистограмм, матриц диаграмм рассеяния, параллельных диаграмм, графиков-щеток и пузырьковых диаграмм.

18.3.2. Использование корреляции для исследования пропущенных значений

Перед тем как двигаться дальше, нужно познакомиться с еще одним заслуживающим внимания подходом. Значения в наборе данных можно заменить условными значениями: 1 – пропущенные значения, 0 – имеющиеся. Полученную таблицу иногда называют матрицей теней (shadow matrix). Вычисление корреляций между этими преобразованными переменными и между исходными переменными поможет узнать, какие пары переменных склонны к отсутствию значений, а также выявить связи между отсутствием значений в одной переменной и значениями других переменных.

Рассмотрим следующую инструкцию:

```
x <- as.data.frame(abs(is.na(sleep)))
```

Элементы таблицы данных x равны 1, если соответствующий элемент в наборе данных sleep отсутствует, и 0 – в противном случае. В этом можно убедиться, рассмотрев первые несколько строк каждой таблицы:

```
> head(sleep, n=5)

   BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger
1 6654.000   5712.0   NA    NA   3.3 38.6  645    3   5      3
2    1.000      6.6  6.3   2.0   8.3  4.5   42    3   1      3
3    3.385     44.5   NA    NA  12.5 14.0   60    1   1      1
4    0.920      5.7   NA    NA  16.5   NA   25    5   2      3
5 2547.000   4603.0  2.1   1.8   3.9 69.0  624    3   5      4

> head(x, n=5)
  BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger
1       0        0    1     1     0    0    0    0   0      0
2       0        0    0     0     0    0    0    0   0      0
3       0        0    1     1     0    0    0    0   0      0
4       0        0    1     1     0    1    0    0   0      0
5       0        0    0     0     0    0    0    0   0      0
```

Инструкция

```
y <- x[which(apply(x,2,sum)>0)]
```

извлечет переменные, в которых пропущено несколько значений (но не все), а инструкция

```
cor(y)
```

вычислит корреляции между этими преобразованными переменными:

```
        NonD  Dream  Sleep   Span   Gest
NonD   1.000  0.907  0.486  0.015 -0.142
Dream  0.907  1.000  0.204  0.038 -0.129
Sleep  0.486  0.204  1.000 -0.069 -0.069
Span   0.015  0.038 -0.069  1.000  0.198
Gest  -0.142 -0.129 -0.069  0.198  1.000
```

Здесь видно, что переменные `Dream` и `NonD` чаще имеют пропущенные значения в одних и тех же наблюдениях (`r = 0.91`). Это в меньшей степени относится к переменным `Sleep` и `NonD` (`r = 0.49`) или `Sleep` и `Dream` (`r = 0.20`).

Наконец, можно рассмотреть связь между наличием пропущенных значений одной переменной и наблюдаемыми значениями других переменных:

```
> cor(sleep, y, use="pairwise.complete.obs")
           NonD  Dream   Sleep   Span   Gest
BodyWgt   0.227  0.223  0.0017 -0.058 -0.054
BrainWgt  0.179  0.163  0.0079 -0.079 -0.073
NonD         NA     NA      NA -0.043 -0.046
Dream    -0.189     NA -0.1890  0.117  0.228
Sleep    -0.080 -0.080      NA  0.096  0.040
Span      0.083  0.060  0.0052     NA -0.065
Gest      0.202  0.051  0.1597 -0.175     NA
Pred      0.048 -0.068  0.2025  0.023 -0.201
Exp       0.245  0.127  0.2608 -0.193 -0.193
Danger    0.065 -0.067  0.2089 -0.067 -0.204
Warning message:
In cor(sleep, y, use = "pairwise.complete.obs") :
  the standard deviation is zero
```

В этой корреляционной матрице строки соответствуют наблюдаемым переменным, а столбцы – преобразованным, несущим информацию о наличии пропущенных значений. Вы можете не обращать внимания на предупреждения и на значения `NA` – это побочные эффекты нашего метода.

В первом столбце матрицы видно, что значения продолжительности сна без сновидений чаще отсутствуют у животных с большими значениями веса (`r = 0.227`), продолжительности беременности (`r = 0.202`) и уязвимости во время сна (`0.245`). Другие столбцы можно интерпретировать сходным образом. Все коэффициенты корреляции в этой матрице не слишком велики, поэтому можно предположить, что пропуски несильно отклоняются от полностью случайных и могут считаться случайными.

Обратите внимание, что никогда нельзя исключать возможность неслучайности отсутствия значений, потому что неизвестно, какие значения в реальности принимали пропущенные данные. Напри-

мер, мы не знаем, есть ли связь между продолжительностью сновидений и вероятностью пропуска значений этой переменной. В отсутствие убедительных свидетельств в пользу обратного мы обычно считаем пропуски полностью случайными или случайными.

18.4. Определение причин отсутствия данных и их влияния

Мы исследуем количество, распределение и структуру пропущенных данных, чтобы получить представление о возможных механизмах возникновения пропущенных данных и о влиянии пропущенных данных на качество ответов на интересующие нас вопросы. В частности, нам нужно знать:

- Какая доля данных пропущена?
- Сосредоточены ли пропущенные данные в нескольких переменных или они широко распределены по всему набору данных?
- Можно ли их считать случайными?
- Позволяет ли ковариация пропущенных данных друг с другом или с наблюдаемыми данными обнаружить возможный механизм, лежащий в основе пропущенных значений?

Ответы на эти вопросы позволят определить статистические методы, которые лучше всего подходят для анализа данных. Например, если пропущенные данные сосредоточены в нескольких второстепенных переменных, то можно удалить эти переменные и спокойно продолжить анализ. Если частота пропущенных значений невелика (скажем, меньше 10 %) и они случайным образом разбросаны по всему набору данных (полностью случайные пропуски), то можно ограничиться полными строками и при этом получить надежные и обоснованные результаты. Если предполагается, что данные пропущены случайно или полностью случайно, то можно выполнить множественное восстановление пропущенных данных, чтобы получить обоснованные выводы. Если значения пропущены неслучайно, то следует обратиться к специализированным методам, собрать новые данные или сменить профессию на более простую и благодарную.

Вот несколько примеров:

- в недавнем исследовании бумажных анкет я обнаружил, что некоторые вопросы пропускаются совместно, потому что многие респонденты не поняли, что у третьей страницы анкеты была оборотная сторона (с вопросами). В данном случае это полностью случайные пропуски;
- в другом исследовании стилей руководства по всему миру выяснилось, что много данных отсутствует в графе «образо-

вание». Оказалось, что европейцы чаще оставляли это поле анкеты пустым. Это происходило потому, что предложенные значения не имели смысла для участников опроса из определенных стран. В этом случае пропуски, скорее всего, были случайными;

- наконец, я участвовал в исследованиях депрессии, где пожилые пациенты чаще пропускали вопросы, посвященные подавленному настроению, по сравнению с молодыми. Выяснилось, что пожилые пациенты неохотно признавали такие симптомы, поскольку это шло вразрез с их стремлением не пасовать перед трудностями. К сожалению, было также выявлено, что пациенты с наиболее тяжелыми формами депрессии чаще пропускали эти вопросы из-за чувства безнадежности и проблем с концентрацией внимания. Так что эти значения были пропущены неслучайно.

Как вы могли понять из приведенных примеров, обнаружение закономерностей в пропущенных данных – только первый шаг. Чтобы определить причины пропуска данных, необходимо использовать ваши знания о предмете исследования и процессе сбора данных.

Теперь, рассмотрев причины пропусков и их влияние, разберемся, как в этом случае скорректировать статистические подходы. Мы сосредоточимся на самых распространенных методах: восстановлении пропущенных данных, традиционном способе, который заключается в удалении пропущенных данных, и современном подходе с использованием моделирования. Попутно мы кратко рассмотрим устаревшие методы, которые должны выйти из употребления. Наша цель остается неизменной: ответить настолько корректно, насколько это возможно, на основные вопросы, ради ответа на которые мы и собирали данные, учитывая, что полная информация нам недоступна.

18.5. Рациональный подход к обработке отсутствующих данных

При так называемом рациональном подходе, когда предпринимаются попытки заменить или восстановить пропущенные значения, используются математические или логические связи между переменными. Несколько примеров помогут прояснить ситуацию.

В наборе данных `sleep` переменная `Sleep` является суммой переменных `Dream` и `NonD`. Если известны значения любых двух переменных в данном наблюдении, то можно вычислить третью. Таким образом, если есть какие-то наблюдения, где пропущены значения только одной из трех переменных, можно восстановить пропущенную информацию при помощи сложения или вычитания.

Во втором примере мы рассмотрим исследование различий в стиле жизни людей разных поколений, где возрастная группа определялась годом рождения. Участников исследования спрашивали и об их дате рождения, и о возрасте. Если дата рождения отсутствует, вы можете восстановить год рождения (и, таким образом, возрастную группу), зная возраст опрашиваемых и дату проведения исследования.

В качестве примера использования логических связей для восстановления пропущенных значений рассмотрим исследование стиля руководства. Во время опросов участникам нужно было ответить, являются ли они руководителями, и указать число непосредственных подчиненных. Если опрашиваемые игнорировали вопрос про руководство, но указывали, что у них есть один или более непосредственных подчиненных, было бы логичным предположить, что они являлись руководителями.

В качестве последнего примера я расскажу о сравнительных исследованиях стиля и эффективности руководства мужчин и женщин, в которые я часто бываю вовлечен. Респонденты заполняют анкеты, где они указывают свои имя и фамилию, пол и дают детальную оценку своего стиля руководства и его эффективности. Если опрашиваемый игнорирует вопрос о поле, мне приходится восстанавливать это значение, чтобы можно было включить этого человека в исследование. В одном из последних опросов из 66 000 руководителей 11 000 (17 %) оставили вопрос о поле без ответа.

Для улучшения ситуации я пользуюсь следующим алгоритмом. Для начала я выясняю соответствие имени и пола. Некоторые имена свойственны только мужчинам, некоторые – только женщинам, а некоторые – обоим полам. Например, имя *Вильям* встретилось 417 раз, и это всегда были мужчины. Напротив, имя Крис было у 237 человек, часть из них – мужчины (86 %), а часть – женщины (14 %). Если имя встречалось больше, чем 20 раз, и всегда принадлежало либо мужчинам, либо женщинам (но никогда – и тем, и другим), я предполагал, что это имя ассоциировано только с одним полом. Я использовал это допущение для составления таблицы соответствий полов и имен, характерных только для одного пола. Используя эту таблицу для участников с неизвестным полом, я смог восстановить 7000 пропущенных значений (63 % всех пропущенных ответов).

Для применения рационального подхода обычно требуется творческое мышление наряду с должными навыками управления данными. Восстановление данных может быть точным (как в примере со сном) или приблизительным (как в примере с полами). В следующем разделе мы рассмотрим подход, избавляющий от отсутствующих значений удалением строк.

18.6. Удаление пропущенных данных

Самый распространенный подход к обработке отсутствующих данных – простое удаление. Обычно под этим подразумевается удаление переменных (столбцов) с чрезмерным количеством отсутствующих данных, и наблюдений (строк), содержащих отсутствующие данные по любой из оставшихся переменных (построчное удаление). Менее распространенной альтернативой является удаление только отсутствующих данных, участвующих в конкретном анализе (например, попарное удаление). Все они описаны далее.

18.6. Анализ полных строк (построчное удаление)

В анализе полных строк (complete-case analysis) участвуют только строки без пропущенных значений. Многие широко используемые статистические пакеты по умолчанию используют *построчное удаление* при работе с пропущенными данными. Такой подход настолько распространен, что многие аналитики, проводя регрессионный или дисперсионный анализ, могут даже не осознавать, что существует *проблема пропущенных данных*, которую нужно как-то решать!

Извлечь полные строки из матрицы или таблицы данных можно с помощью функции `complete.cases()`:

```
newdata <- mydata[complete.cases(mydata),]
```

Этого же результата можно добиться при помощи функции `na.omit`:

```
newdata <- na.omit(mydata)
```

Обе функции удалят все строки с пропущенными значениями из объекта `mydata` перед его сохранением в виде объекта `newdata`.

Предположим, что нас интересуют корреляции между переменными в исследовании сна. Применив построчное удаление, исключим из анализа всех животных, для которых часть данных пропущена:

```
> options(digits=1)
> cor(na.omit(sleep))
         BodyWgt BrainWgt NonD Dream Sleep  Span  Gest  Pred  Exp Danger
BodyWgt      1.0      0.9 -0.4 -0.07  -0.3  0.47  0.71  0.10  0.4   0.26
BrainWgt     0.9      1.0 -0.4 -0.07  -0.3  0.63  0.73 -0.02  0.3   0.15
NonD        -0.4     -0.4  1.0  0.52   1.0 -0.37 -0.61 -0.35 -0.6  -0.53
Dream       -0.1     -0.1  0.5  1.00   0.7 -0.27 -0.41 -0.40 -0.5  -0.57
Sleep       -0.3     -0.3  1.0  0.72   1.0 -0.38 -0.61 -0.40 -0.6  -0.60
Span         0.5      0.6 -0.4 -0.27  -0.4  1.00  0.65 -0.17  0.3   0.01
Gest         0.7      0.7 -0.6 -0.41  -0.6  0.65  1.00  0.09  0.6   0.31
Pred         0.1     -0.0 -0.4 -0.40  -0.4 -0.17  0.09  1.00  0.6   0.93
Exp          0.4      0.3 -0.6 -0.50  -0.6  0.32  0.57  0.63  1.0   0.79
Danger       0.3      0.2 -0.5 -0.57  -0.6  0.01  0.31  0.93  0.8   1.00
The correlations in this table are based solely on the 42 mammals that have
complete data on all variables. (Note that the statement cor(sleep,
```

```
use="complete.obs")
would have produced the same results.)
```

Для исследования влияния продолжительности жизни и срока беременности на продолжительность сновидений можно применить линейную регрессию с построчным удалением пропущенных значений:

```
> fit <- lm(Dream ~ Span + Gest, data=na.omit(sleep))
> summary(fit)

Call:
lm(formula = Dream ~ Span + Gest, data = na.omit(sleep))

Residuals:
   Min     1Q Median     3Q    Max
-2.333 -0.915 -0.221  0.382  4.183

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.480122   0.298476    8.31  3.7e-10 ***
Span        -0.000472   0.013130   -0.04    0.971
Gest        -0.004394   0.002081   -2.11    0.041 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1 on 39 degrees of freedom
Multiple R-squared: 0.167,      Adjusted R-squared: 0.125
F-statistic: 3.92 on 2 and 39 DF,  p-value: 0.0282
```

Здесь видно, что млекопитающие с более короткой продолжительностью беременности дольше видят сны (при постоянной продолжительности жизни) и продолжительность жизни не связана с продолжительностью сна со сновидениями при постоянных значениях срока беременности. Анализ был основан на 42 наблюдениях с полными данными.

Что будет, если в предыдущем примере выражение `data=na.omit(sleep)` заменить на `data=sleep`? Как многие функции R, `lm()` использует узкое определение построчного удаления значений. В данном случае будут удалены строки, в которых есть пропущенные значения в любой из переменных, используемых функцией (`Dream`, `Span` и `Gest`). Анализ будет основан на 42 строках.

При построчном удалении данных предполагается, что пропуски полностью случайны (то есть полные строки – это случайная выборка из всего набора данных). В данном примере предполагалось, что 42 проанализированных животных – это случайная выборка из всех 62 видов. Полученные параметры регрессии исказятся настолько, насколько будет нарушено предположение о полной случайности пропусков. Удаление всех строк с пропущенными данными также может снизить статистическую мощность, уменьшив

объем выборки. В приведенном примере построчное удаление сократило размер выборки на 32 %. Далее мы рассмотрим подход, который позволяет использовать весь набор данных целиком (включая строки с пропущенными данными).

18.6.2. Анализ доступных наблюдений (попарное удаление)

Попарное удаление часто считается альтернативой построчному удалению при работе с наборами данных, имеющими пропущенные значения. При попарном удалении наблюдения удаляются только в том случае, если в них отсутствуют значения переменных, участвующих в конкретном анализе. Рассмотрим следующий код:

```
> cor(sleep, use="pairwise.complete.obs")
         BodyWgt BrainWgt NonD Dream Sleep  Span Gest  Pred  Exp Danger
BodyWgt     1.00     0.93 -0.4  -0.1  -0.3  0.30  0.7  0.06  0.3   0.13
BrainWgt    0.93     1.00 -0.4  -0.1  -0.4  0.51  0.7  0.03  0.4   0.15
NonD       -0.38    -0.37  1.0   0.5   1.0 -0.38 -0.6 -0.32 -0.5  -0.48
Dream      -0.11    -0.11  0.5   1.0   0.7 -0.30 -0.5 -0.45 -0.5  -0.58
Sleep      -0.31    -0.36  1.0   0.7   1.0 -0.41 -0.6 -0.40 -0.6  -0.59
Span        0.30     0.51 -0.4  -0.3  -0.4  1.00  0.6 -0.10  0.4   0.06
Gest        0.65     0.75 -0.6  -0.5  -0.6  0.61  1.0  0.20  0.6   0.38
Pred        0.06     0.03 -0.3  -0.4  -0.4 -0.10  0.2  1.00  0.6   0.92
Exp         0.34     0.37 -0.5  -0.5  -0.6  0.36  0.6  0.62  1.0   0.79
Danger      0.13     0.15 -0.5  -0.6  -0.6  0.06  0.4  0.92  0.8   1.00
```

В этом примере корреляции между любыми двумя переменными вычислены на основе всех имеющихся наблюдений для этих двух переменных (не принимая во внимание все остальные). Корреляция между переменными `BodyWgt` и `BrainWgt` рассчитана для всех 62 животных (число животных, для которых имеются данные в обеих переменных). Корреляция между переменными `Dream` и `BodyWgt` основана на 50 животных, а между переменными `BodyWgt` и `NonD` – на 48.

Кажется, что попарное удаление позволяет использовать все имеющиеся данные, но на самом деле расчет каждого коэффициента производится для разных выборок из данных. Это может привести к искаженным и сложно интерпретируемым результатам. Я рекомендую держаться подальше от этого подхода.

Далее мы рассмотрим подход, использующий весь набор данных (включая случаи с отсутствующими данными).

18.7. Одиночное восстановление пропущенных данных

При одиночном восстановлении каждое отсутствующее значение замещается разумным альтернативным значением (т. е. правдоподобным предположением). В этом разделе мы рассмотрим три подхода и опишем, когда использовать (или не использовать) каждый из них.

18.7.1. Простое восстановление

При *простом восстановлении* (simple imputation) пропущенные значения переменной заменяются одним и тем же числом (например, средним, медианой или модой). Используя замену средним значением, можно заменить пропущенные значения переменной `Dream` числом 1,97 (среднее имеющихся значений).

Преимущество простого восстановления заключается в том, что оно «решает проблему пропущенных значений» без уменьшения размера выборки. Ну что же, простое восстановление данных – это действительно просто, однако оно дает искаженные результаты для неслучайных пропусков. Если пропущенных значений достаточно много, их простое замещение, скорее всего, приведет к занижению среднеквадратичной ошибки, исказит корреляции между переменными и вызовет появление некорректных значений статистической ошибки первого рода. Как и в случае попарного удаления, я рекомендую избегать этого подхода.

18.7.2. Восстановление методом k-ближайших соседей

Идея *восстановления методом k-ближайших соседей* проста. Для наблюдения с одним или несколькими отсутствующими значениями нужно найти наиболее похожие наблюдения, но имеющие значения, и использовать эти наблюдения для восстановления.

Например, рассмотрим следующее наблюдение из таблицы данных `sleep`:

```
BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger
   1.41     17.5  4.8   1.3   6.1   34   NA    1   2      1
```

В этом наблюдении отсутствует значение переменной, определяющей продолжительность беременности (`Gest`). Чтобы восстановить это отсутствующее значение, можно:

1. найти в таблице данных *k* наблюдений, наиболее близких (похожих) к этому случаю на основе других девяти переменных;
2. агрегировать значения `Gest` в этих *k* наблюдениях. Например, вычислить среднее значение `Gest`;
3. заменить отсутствующее значение этим агрегированным значением.

Шаги должны повторяться для каждого наблюдения с пропущенными значениями.

Чтобы использовать этот подход, необходимо ответить на три вопроса. Как определить ближайшего соседа? Сколько ближайших соседей использовать? Как должны агрегировать значения?

В пакете `VIM` имеется функция `kNN()`, реализующая восстановление методом *k*-ближайших соседей, которая по умолчанию:

- определяет ближайших соседей как случаи с наименьшим расстоянием Гауэра (Kowarik & Templ, 2016) до целевого наблюдения. В отличие от евклидовых расстояний, описанных в главе 16, расстояние Гауэра можно вычислить для данных, содержащих как количественные, так и категориальные переменные. При определении ближайших соседей используются все доступные переменные;
- выявляет пять ближайших соседей для каждого наблюдения с пропущенными значениями;
- при агрегировании данных для количественных пропущенных значений используется медиана, а для категориальных – мода, наиболее часто встречающаяся категория.

Каждую из этих настроек по умолчанию можно изменить. Подробности ищите в `help(kNN)`.

В листинге 18.2 показан пример применения функции `kNN()` к таблице данных `sleep`.

Листинг 18.2. Восстановление пропущенных значений методом *k*-ближайших соседей

```
> library(VIM)
> head(sleep)

   BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger
1 6654.000   5712.0   NA    NA   3.3 38.6  645    3   5      3
2    1.000      6.6  6.3   2.0   8.3  4.5   42    3   1      3
3    3.385     44.5   NA    NA  12.5 14.0   60    1   1      1
4    0.920      5.7   NA    NA  16.5   NA   25    5   2      3
5 2547.000   4603.0  2.1   1.8   3.9 69.0  624    3   5      4
6   10.550    179.5  9.1   0.7   9.8 27.0  180    4   4      4

> sleep_imp <- kNN(sleep, imp_var=FALSE)

> head(sleep_imp)
   BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger
1 6654.000   5712.0  3.2   0.8   3.3 38.6  645    3   5      3
2    1.000      6.6  6.3   2.0   8.3  4.5   42    3   1      3
3    3.385     44.5 12.8   2.4  12.5 14.0   60    1   1      1
4    0.920      5.7 10.4   2.4  16.5  3.2   25    5   2      3
5 2547.000   4603.0  2.1   1.8   3.9 69.0  624    3   5      4
6   10.550    179.5  9.1   0.7   9.8 27.0  180    4   4      4
```

В таблице данных `sleep_imp` все отсутствующие значения были замещены восстановленными значениями. По умолчанию функция `kNN()` добавит в таблицу 10 новых переменных (`BodyWgt_imp`, `BrainWgt_imp`, ..., `Danger_imp`) со значениями `TRUE`/`FALSE`, указывающими, какие фактические значения были восстановлены. Добавив параметр `var_imp=FALSE`, можно подавить добавление этих дополнительных переменных.

Восстановление с помощью `kNN()` хорошо подходит для наборов данных небольшого или среднего размера (скажем, < 1000 наблюдений). Поскольку расстояние вычисляется между каждым наблюдением с отсутствующими данными и всеми другими наблюдениями, этот прием плохо масштабируется для обработки больших объемов данных. В таких случаях рекомендуется использовать подход, описанный далее.

18.7.3. missForest

Для восстановления пропущенных значений в больших наборах данных можно использовать случайные леса (глава 17). Для набора p переменных X_1, X_2, ..., X_p следует выполнить следующие шаги:

1. заменить пропущенные значения в каждой количественной переменной средним значением. Заменить пропущенные значения в каждой категориальной переменной, используя моду – наиболее часто встречающуюся категорию. Запомнить наблюдения, где была выполнена подстановка пропущенных значений;
2. вернуть недостающие данные для переменной X_1. Создать обучающий набор наблюдений без пропущенных значений в этой переменной. Используя обучающую выборку, построить модель случайного леса (глава 17), чтобы предсказать X_1. Использовать модель, чтобы восстановить X_1 в наблюдениях с отсутствующими значениями X_1;
3. повторить шаг 2 для переменных от X_2 до X_p;
4. повторять шаги 2 и 3, пока восстановленные значения не будут изменяться не более чем на заданную величину.

Эти операции проще выполнить, чем описать! Для этого можно использовать функцию `missForest()` из пакета `missForest`. В листинге представлен пример применения этой функции к таблице данных `sleep`.

Листинг 18.3. Восстановление пропущенных значений с помощью случайного леса

```
> library(missForest)
> set.seed(1234)
> sleep_imp <- missForest(sleep)$ximp

  missForest iteration 1 in progress...done!
  missForest iteration 2 in progress...done!
  missForest iteration 3 in progress...done!
  missForest iteration 4 in progress...done!
  missForest iteration 5 in progress...done!
  missForest iteration 6 in progress...done!

> head(sleep_imp)
   BodyWgt BrainWgt      NonD  Dream Sleep   Span Gest Pred Exp Danger
```

```
1 6654.000   5712.0  3.391857 1.1825   3.3 38.600  645    3   5       3
2    1.000      6.6  6.300000 2.0000   8.3  4.500   42    3   1       3
3    3.385     44.5 10.758000 2.4300  12.5 14.000   60    1   1       1
4    0.920      5.7 11.572000 2.7020  16.5  7.843   25    5   2       3
5 2547.000   4603.0  2.100000 1.8000   3.9 69.000  624    3   5       4
6   10.550    179.5  9.100000 0.7000   9.8 27.000  180    4   4       4
```

Здесь намеренно выполняется установка начального случайного числа, чтобы обеспечить воспроизводимость результатов. В этом случае потребовалось шесть итераций, чтобы восстановленные значения стабилизировались.

Так же как функция `kNN()`, функция `missForest()` может работать и с количественными, и с категориальными данными. Подход на основе случайного леса лучше подходит для наборов данных от среднего до большого размера (скажем, более 500 случаев), так как в этом случае не так остро проявляется проблема переобучения. Для небольших наборов данных эффективнее использовать подход на основе *k*-ближайших соседей.

Если в центре внимания анализа находится проверка гипотез, то рекомендуется использовать метод восстановления, учитывающий неопределенность, вызванную пропущенными значениями. Это обеспечивает подход множественного восстановления.

18.8. Множественное восстановление пропущенных данных

Метод *множественного восстановления пропущенных данных* (multiple imputation, MI) – это способ заполнения пропусков при помощи повторного моделирования. Множественное восстановление часто применяется для работы с пропущенными данными в сложных ситуациях. При этом подходе из существующего набора данных с пропущенными значениями создается несколько полных наборов данных (обычно от трех до десяти). Для замещения пропущенных значений в производных наборах данных используются методы Монте-Карло. К каждому из производных наборов данных применяются стандартные статистические методы, а на основании их результатов формируются оценки окончательных результатов и доверительные интервалы, которые учитывают неопределенность, созданную пропущенными значениями. Эта идея хорошо реализована в таких пакетах R, как `Amelia`, `mice` и `mi`. В этом разделе мы сосредоточим свое внимание на подходе, реализованном в пакете `mice` (multivariate imputation by chained equations – многомерное восстановление данных при помощи связанных уравнений). Чтобы понять, как работает этот пакет, рассмотрите схему на рис. 18.6.

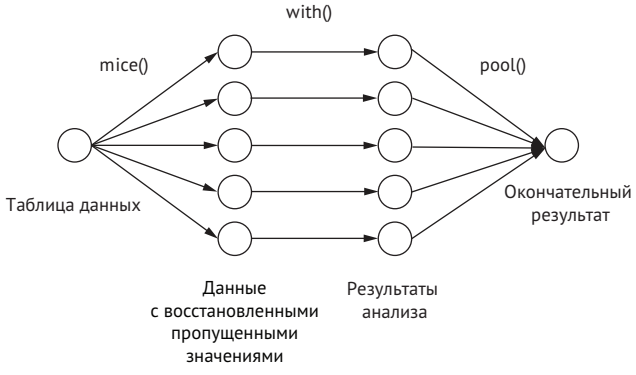


Рис. 18.6. Этапы множественного восстановления пропущенных данных при помощи подхода, реализованного в пакете mice

Функция `mice()` принимает исходную таблицу данных с пропущенными значениями и возвращает объект, содержащий несколько полных наборов данных (пять по умолчанию). Каждый такой полный набор данных получается в результате восстановления пропущенных данных в исходной таблице. В алгоритме восстановления данных есть случайная составляющая, поэтому все производные полные наборы данных немного отличаются друг от друга. Затем вызовом функции `with()` к наборам данных применяется статистическая модель (например, линейная или обобщенная линейная). И в заключение функция `pool()` объединяет результаты, полученные для отдельных производных наборов данных. Стандартные ошибки и значения статистической ошибки первого рода в этой окончательной модели корректно отражают неопределенность, обусловленную как наличием пропущенных значений, так и алгоритмами множественной замены.

Как функция mice() замещает пропущенные значения?

Пропущенные значения замещаются при помощи *выборок Гиббса*. По умолчанию значения каждой переменной, содержащей пропущенные значения, предсказываются по значениям остальных переменных. Полученные уравнения используются для замещения пропущенных данных подходящими значениями. Этот процесс повторяется, пока значения для пропущенных данных не сойдутся. Пользователь может выбирать вид прогнозной модели (называемой *простейшим методом замещения*) для каждой переменной и определять переменные, которые в нее войдут.

По умолчанию для замены пропущенных значений непрерывных переменных используется метод соответствия предсказанного среднего (predictive mean matching), а логистическая или полиномиальная логистическая регрессия применяются для *дихотомических* (фактор с двумя уровнями) или *политомических* (фактор с более чем двумя

уровнями) переменных соответственно. К другим простейшим методам замещения относятся байесовская линейная регрессия, дискриминантный анализ, двухуровневое нормальное замещение и составление случайных выборок из наблюдаемых значений. Также есть возможность использовать свои методы.

Анализ данных в пакете `mice` обычно имеет такую структуру:

```
library(mice)
imp <- mice(data, m)
fi t <- with(imp, analysis)
pooled <- pool(fi t)
summary(pooled),
```

где

- *data* – матрица или таблица данных с пропущенными значениями;
- `imp` – список, содержащий *m* наборов данных с восстановленными пропущенными значениями вместе с информацией о том, как это восстановление было проведено. По умолчанию *m* = 5;
- *analysis* – формула, определяющая тип статистического метода, который должен быть применен к каждому из *m* восстановленных наборов данных. К таким методам относятся `lm()` – линейная регрессия, `glm()` – обобщенная линейная регрессия и `gam()` – обобщенные аддитивные модели. В формулах внутри скобок зависимая переменная указывается слева от знака ~, а независимые (разделенные знаком +) – справа;
- `fit` – список, содержащий результаты *m* отдельных статистических анализов;
- `pooled` – список, содержащий усредненные результаты этих *m* отдельных статистических анализов.

Давайте применим метод множественного восстановления пропущенных данных к нашему набору данных `sleep`. Мы повторим анализ, описанный в разделе 18.6, только на этот раз используем данные обо всех 62 животных. Установите начальное значение для генератора случайных чисел, равное `1234`, чтобы ваши результаты совпали с моими.

```
> library(mice)
> data(sleep, package="VIM")
> imp <- mice(sleep, seed=1234)

 [...output deleted to save space...]

> fit <- with(imp, lm(Dream ~ Span + Gest))
> pooled <- pool(fit)
> summary(pooled)
```

```
          term estimate std.error statistic   df  p.value
1 (Intercept)  2.59669   0.24861    10.445 52.0 2.29e-14
2         Span -0.00399   0.01169    -0.342 55.6 7.34e-01
3         Gest -0.00432   0.00146    -2.961 55.2 4.52e-03
```

Здесь видно, что коэффициент регрессии для переменной `Span` незначим ($p \cong 0{,}08$), а коэффициент для переменной `Gest` значим на уровне $p < 0{,}01$. Если сравнить эти результаты с полученными при помощи анализа полных строк (раздел 18.6), то можно заметить, что по данному вопросу мы пришли к таким же заключениям. Срок беременности (статистически) имеет значимую отрицательную связь с продолжительностью сновидений при постоянных значениях длины жизни. Несмотря на то что анализ полных строк был основан на 42 наблюдениях с полными данными, текущий анализ основан на информации, полученной из полного набора с 62 наблюдениями.

Получить больше информации о замещении пропущенных значений можно, исследовав объекты, созданные в ходе анализа. Давайте посмотрим на общую информацию об объекте `imp`. Например, инструкция

```
> imp$imp$Dream
     1   2   3   4   5
1  0.0 0.5 0.5 0.5 0.3
3  0.5 1.4 1.5 1.5 1.3
4  3.6 4.1 3.1 4.1 2.7
14 0.3 1.0 0.5 0.0 0.0
24 3.6 0.8 1.4 1.4 0.9
26 2.4 0.5 3.9 3.4 1.2
30 2.6 0.8 2.4 2.2 3.1
31 0.6 1.3 1.2 1.8 2.1
47 1.3 1.8 1.8 1.8 3.9
53 0.5 0.5 0.6 0.5 0.3
55 2.6 3.6 2.4 1.8 0.5
62 1.5 3.4 3.9 3.4 2.2
```

позволяет отобразить пять вариантов восстановленных значений для каждого из 12 животных, для которых отсутствовали значения в переменной `Dream`. Обзор этих матриц позволит понять, имеют ли восстановленные значения смысл. Отрицательные значения продолжительности сна могут заставить вас остановиться (или станут причиной ночных кошмаров).

Каждый из *m* восстановленных наборов данных можно вывести при помощи функции `complete()`. Она имеет следующий синтаксис:

```
complete(imp, action=#)
```

где # – это номер одного из *m* искусственных наборов данных с замещенными значениями. Например, следующий код:

```
> dataset3 <- complete(imp, action=3)
> dataset3
```

```
    BodyWgt BrainWgt NonD Dream Sleep  Span  Gest Pred Exp Danger
1  6654.000  5712.00  3.2   0.5   3.3  38.6 645.0    3   5      3
2     1.000     6.60  6.3   2.0   8.3   4.5  42.0    3   1      3
3     3.385    44.50 11.0   1.5  12.5  14.0  60.0    1   1      1
4     0.920     5.70 13.2   3.1  16.5   7.0  25.0    5   2      3
5  2547.000  4603.00  2.1   1.8   3.9  69.0 624.0    3   5      4
6    10.550   179.50  9.1   0.7   9.8  27.0 180.0    4   4      4
[...вывод удален для экономии места...]
```

выведет третий (из пяти) полный набор данных, созданный методом множественного восстановления пропущенных данных.

Из-за экономии места мы лишь коротко рассмотрели алгоритм восстановления пропущенных данных, реализованный в пакете `mice`. Пакеты `mi` и `Amelia` тоже содержат методы, достойные вашего внимания. Если вы интересуетесь множественным восстановлением пропущенных данных, рекомендую следующие источники информации:

- электронное руководство «Applied Missing Data Analysis with SPSS and (R)Studio» Мартина У. Хейманса (Martijn W. Heymans) и Ириса Экхута (Iris Eekhout), доступное по адресу: https://bookdown.org/mwheymans/bookmi/;
- статьи Van Buuren & Groothuis-Oudshoorn (2010) и Yu-Sung, Gelman, Hill & Yajima (2010);
- электронное руководство «Amelia II: A Program for Missing Data» (http://gking.harvard.edu/amelia/).

Каждый из этих источников информации поможет углубить и расширить ваши представления об этой важной, но недостаточно широко используемой методологии.

18.9. Другие подходы обработки пропущенных данных

В R реализовано и несколько других подходов к обработке пропущенных данных. Пакеты, перечисленные в табл. 18.2, хотя и не так широко применимы, как описанные выше методы, но содержат функции, которые могут быть весьма полезными при определенных обстоятельствах.

Таблица 18.2. Специализированные методы работы с пропущенными данными

Пакет	Описание
`norm`	Оценка методом максимального правдоподобия пропущенных значений в многомерных данных с нормальным распределением
`cat`	Анализ категориальных данных с отсутствующими значениями
`longitudinalData`	Содержит полезные функции, включая алгоритмы интерполяции, для восстановления пропущенных данных во временных рядах

Окончание табл. 18.2

Пакет	Описание
`kmi`	Множественное восстановление пропущенных данных методом Каплана–Мейера при анализе выживаемости
`mix`	Множественное восстановление пропущенных значений для смешанных категориальных и непрерывных данных при помощи общей модели положения (general location model)
`pan`	Множественное восстановление пропущенных значений в многомерных сгруппированных данных

Также можно обратиться к обзору задач по восстановлению пропущенных данных «CRAN Task View on Missing Data» (https://cran.r-project.org/web/views/MissingData.html), где перечислены все подходы, поддерживаемые в R.

Итоги

- Статистические методы требуют передачи для анализа полных наборов данных, но на практике многие наборы данных имеют пропущенные значения.
- Для изучения распределения пропущенных значений в наборе данных можно с успехом использовать функции из пакетов `VIM` и `mice`.
- Удаление строк является наиболее популярным методом обработки пропущенных значений и стандартным подходом для многих статистических процедур. Но этот подход может привести к потере мощности, если будут удалены значительные объемы данных.
- `kNN()` и `missForest()` – отличные инструменты для восстановления пропущенных значений. Первая хорошо работает с наборами данных малого и среднего размеров, а вторая – с большими наборами данных.
- Множественное восстановление основано на моделировании и учитывает неопределенность, которую добавляют пропущенные значения.
- Если количество отсутствующих данных не слишком мало, обычно следует избегать простого восстановления (например, замены средним) и попарного удаления.

Часть V

Расширение возможностей

В этой заключительной части мы рассмотрим дополнительные темы, которые помогут вам расширить ваши возможности в программировании на языке R. Глава 19 завершает обсуждение приемов создания диаграмм и приводит подробную информацию о том, как настраивать диаграммы, созданные с помощью пакета `ggplot2`. В ней вы научитесь изменять заголовки, метки, надписи на осях, цвета, шрифты, легенды и многое другое, а также объединять несколько графиков в одну общую диаграмму и превращать статические графики в интерактивные веб-формы.

Глава 20 рассматривает язык R на более глубоком уровне. Здесь вас ждет обсуждение возможностей объектно-ориентированного программирования на R, работы со средами и разработки расширенных функций. В этой главе также даются советы по написанию эффективного кода и отладке программ. Глава 20 носит более технический характер, чем другие главы в этой книге, и содержит множество практических советов по разработке программ.

Глава 21 посвящена написанию отчетов. В R имеется богатый набор средств для динамического создания привлекательных отчетов из данных. В этой главе вы узнаете, как создавать отчеты в виде веб-страниц, документов PDF и документов в формате текстовых процессоров (включая Microsoft Word).

На протяжении всей этой книги мы использовали пакеты. В главе 22 вы научитесь писать *свои* пакеты. Это поможет вам организовывать и документировать свою работу, создавать сложные и всеобъемлющие программные решения и делиться своими творениями с другими. Совместное использование пакетов функций с другими также может быть прекрасным способом отблагодарить сообщество R (при этом распространяя свою известность повсюду).

После завершения части V вы получите более глубокое представление о работе R и о том, какие инструменты он предлагает для создания сложной графики, программного обеспечения и отчетов.

19 Продвинутые методы работы с диаграммами

В этой главе:

- настройка диаграмм `ggplot2`;
- добавление аннотаций;
- объединение нескольких диаграмм в одну;
- интерактивная графика.

Создавать диаграммы в R можно множеством способов, но я решил сосредоточиться в этой книге на использовании пакета `ggplot2` как наиболее гибкого и полного инструмента, имеющего последовательную грамматику. Пакет `ggplot2` был представлен в главе 4, где были показаны примеры создания простых геометрических фигур, диаграмм, заголовков и способы управления отображением осей. В главе 6 мы создавали гистограммы, круговые, древовидные и столбиковые диаграммы, графики ядерной плотности, коробчатые и скрипичные диаграммы, а также точечные диаграммы. В главах 8 и 9 демонстрировались способы создания графиков для регрессионного и дисперсионного анализов. В главе 11 обсуждались точечные диаграммы, матрицы точечных диаграмм, пузырьковые диаграммы, линейные диаграммы, коррелограммы и мозаичные

диаграммы. И в других главах тоже были показаны графики, служащие иллюстрациями для рассматриваемых тем.

В этой главе мы продолжим знакомство с пакетом `ggplot2`, но сосредоточимся на настройке – создании диаграмм, точно соответствующих нашим потребностям. Графики помогают выявлять закономерности и описывать тенденции, взаимосвязи, различия, состав и распределение данных. Основная причина, почему может понадобиться настроить диаграмму, заключается в желании расширить возможности в исследовании данных или передать результаты в другом аккуратно оформленном виде. Второстепенная цель – соответствовать требованиям организации или издателя к внешнему виду.

В этой главе мы рассмотрим функции из пакета `ggplot2` для настройки осей и цветов. Используем функцию `theme()` для настройки общего внешнего вида диаграммы, включая внешний вид текста, легенды, линий сетки и фона. Для добавления аннотаций будут использоваться геометрические объекты, опорные линии и метки. Также мы посмотрим, как с помощью пакета `patchwork` объединить несколько диаграмм в одну. Наконец, мы познакомимся с пакетом `plotly` и используем его для преобразования статических графиков `ggplot2` в интерактивную веб-графику, помогающую полнее исследовать данные.

Пакет `ggplot2` имеет огромное количество параметров, доступных для настройки элементов диаграмм. Одна только функция `theme()` имеет более 90 аргументов! Здесь мы сосредоточимся на наиболее часто используемых функциях и аргументах. Если вы читаете эту главу в черно-белом формате, то я рекомендую запускать примеры кода у себя, чтобы увидеть диаграммы в цвете. А чтобы было легче сосредоточиться на самом коде, мы используем очень простые примеры.

К настоящему моменту у вас уже должны быть установлены пакеты `ggplot2` и `dplyr`. Для опробования примеров вам понадобится установить еще несколько пакетов, в том числе `ISLR` и `gapminder` (содержащие примеры данных, с которыми мы будем работать), а также `ggrepel`, `showtext`, `patchwork` и `plotly`, содержащие функции для создания продвинутых диаграмм. Установить их все можно вызовом `install.packages(c("ISLR", "gapminder", "scales", "showtext", "ggrepel", "patchwork", "plotly"))`.

19.1. Управление отображением осей

В `ggplot2` есть функции, управляющие отображением значений переменных в конкретные характеристики диаграмм. Например, функция `scale_x_continuous()` отображает значения количественной переменной в позиции на оси *x*. Функция `scale_color_discrete()` отображает значения категориальной переменной в значения цвета. В данном разделе мы используем эти функции для настройки осей и цветов графика.

19.1.1. Настройка осей

В ggplot2 управление осями *x* и *y* на диаграмме осуществляется с помощью функций scale_x_* и scale_y_*, где * задает тип оси. В табл. 19.1 перечислены наиболее часто используемые функции. Обычно оси настраиваются с целью упростить чтение данных или сделать тенденции более очевидными.

Таблица 19.1. Функции управления отображением осей

Функция	Описание
scale_x_continuous, scale_y_continuous	Определяют порядок отображения осей для непрерывных данных
scale_x_binned, scale_y_binned	Распределяют непрерывные данные по поддиапазонам
scale_x_discrete, scale_y_discrete	Определяют порядок отображения осей для дискретных (категориальных) данных
scale_x_log10, scale_y_log10	Определяют порядок отображения осей в логарифмическом масштабе (по основанию 10) для непрерывных данных
scale_x_date, scale_y_date	Определяют порядок отображения осей для дат и времени

Настройка осей для непрерывных переменных

В первом примере мы используем набор данных mtcars, содержащий характеристики 32 автомобилей. Этот набор данных входит в состав стандартного дистрибутива R. Давайте построим график зависимости величины пробега автомобиля на одном галлоне топлива (mpg) от его массы (wt) в тысячах фунтах[1]:

```
library(ggplot2)
ggplot(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  labs(title = "Fuel efficiency by car weight")
```

Получившаяся диаграмма показана на рис. 19.1. По умолчанию подписываются только основные отметки на осях. Для оси mpg подписи добавлены в позициях с 10 по 35 с интервалом 5. Второстепенные отметки размещаются равномерно между основными и не подписываются.

Какой вес имеет самый тяжелый автомобиль на этом графике? Как далеко уедет на одном галлоне топлива третий из самых легких автомобилей? Чтобы ответить на эти вопросы, придется немного повозиться. Однако мы можем настроить оси *x* и *y*, чтобы упростить эту работу.

[1] 1000 фунтов ≈ 453 килограмма. – *Прим. перев.*

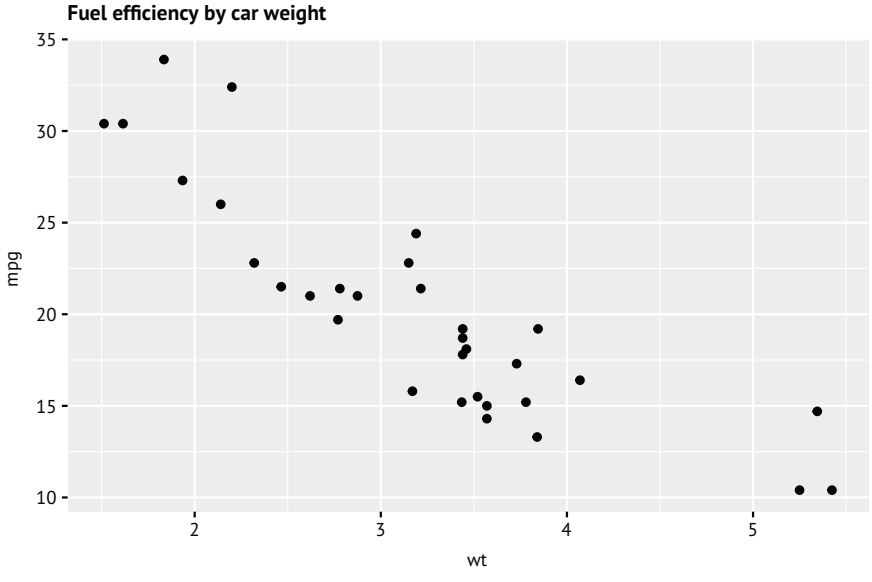


Рис. 19.1. Диаграмма рассеяния ggplot2 с оформлением по умолчанию, отражающая зависимость пробега на одном галлоне топлива (в милях) от массы (в тысячах фунтов) для 32 автомобилей в наборе данных mtcars

Поскольку `wt` и `mpg` – непрерывные переменные, используем функции `scale_x_continuous()` и `scale_y_continuous()` для настройки осей. В табл. 19.2 перечислены наиболее часто используемые параметры этих функций.

Таблица 19.2. Наиболее часто используемые параметры функций scale_*_continuous

Параметр	Описание
`name`	Название масштаба. То же, что используется в функции `labs(x = , y = )`
`breaks`	Числовой вектор с позициями основных отметок. Основные отметки снабжаются подписями автоматически, если иное не определено параметром `labels`. Используйте `NULL` для подавления вывода отметок
`minor_breaks`	Числовой вектор с позициями второстепенных отметок. Второстепенные отметки не снабжаются подписями. Используйте `NULL` для подавления вывода второстепенных отметок
`n.breaks`	Целое число, определяющее количество основных отметок. Число используется только как рекомендация. Функция может изменить это число, чтобы придать больше привлекательности отметкам
`labels`	Строковый вектор, определяющий альтернативные подписи для отметок (должен иметь ту же длину, что и вектор `breaks`)
`limits`	Числовой вектор с двумя элементами, определяющими минимальное и максимальное значения
`position`	Размещение осей (left/right для оси *y*, top/bottom для оси *x*)

Внесем следующие изменения. Для веса:

- добавим подпись «Weight (1000 pounds)» (Вес (1000 фунтов));
- установим диапазон от 1,5 до 5,5;
- потребуем вывести 10 основных отметок;
- подавим вывод второстепенных отметок.

Для пробега:

- добавим подпись «Miles per gallon» (Миль на галлон);
- установим диапазон от 10 до 35;
- потребуем вывести основные отметки на значениях 10, 15, 20, 25, 30 и 35;
- потребуем вывести второстепенные отметки с шагом в одну милю на галлон.

Эти изменения показаны в листинге 19.1.

Листинг 19.1. Диаграмма зависимости пробега автомобиля от веса с настроенными осями

```
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  scale_x_continuous(name = "Weight (1000 lbs.)",     1
                     n.breaks = 10,                   1
                     minor_breaks = NULL,             1
                     limits = c(1.5, 5.5)) +          1
  scale_y_continuous(name = "Miles per gallon",       2
                     breaks = seq(10, 35, 5),         2
                     minor_breaks = seq(10, 35, 1),   2
                     limits = c(10, 35)) +            2
  labs(title = "Fuel efficiency by car weight")
```

1 **Настройка оси *x*.**

2 **Настройка оси *y*.**

Получившаяся диаграмма показана на рис. 19.2. Теперь с легкостью можно определить, что самая тяжелая машина весит почти 5,5 т, а третья машина среди самых легких сможет проехать 34 мили на одном галлоне топлива. Обратите внимание, что здесь мы потребовали вывести 10 основных отметок на оси `wt`, но на графике их только 9. Параметр `n.breaks` воспринимается лишь как рекомендация. Его значение может быть заменено другим близким значением, если в результате получится более удобный для восприятия набор отметок. Позже мы продолжим работу с этой диаграммой.

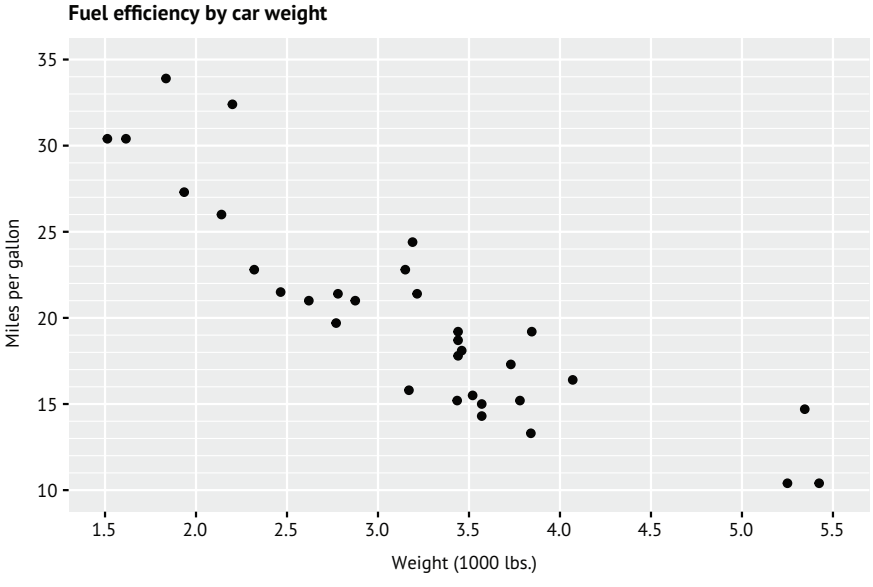


Рис. 19.2. Диаграмма рассеяния ggplot2, отражающая зависимость пробега на одном галлоне топлива (в милях) от массы (в тысячах фунтов) с настроенными осями *x* и *y*. Теперь значения точек проще определить

Настройка осей для категориальных переменных

В предыдущем примере было показано, как настроить оси для непрерывных переменных. В следующем примере мы настроим оси для категориальных переменных. Основой для этого упражнения нам послужит набор данных `Wage` из пакета `ISLR`, содержащий размеры заработной платы и демографическую информацию для 3000 рабочих-мужчин на северо-востоке США, собранную в 2011 году. Построим график зависимости между расой и образованием в этой выборке:

```
library(ISLR)
library(ggplot2)
ggplot(Wage, aes(race, fill = education)) +
  geom_bar(position = "fill") +
  labs(title = "Participant Education by Race")
```

Этот код выведет диаграмму, изображенную на рис. 19.3.

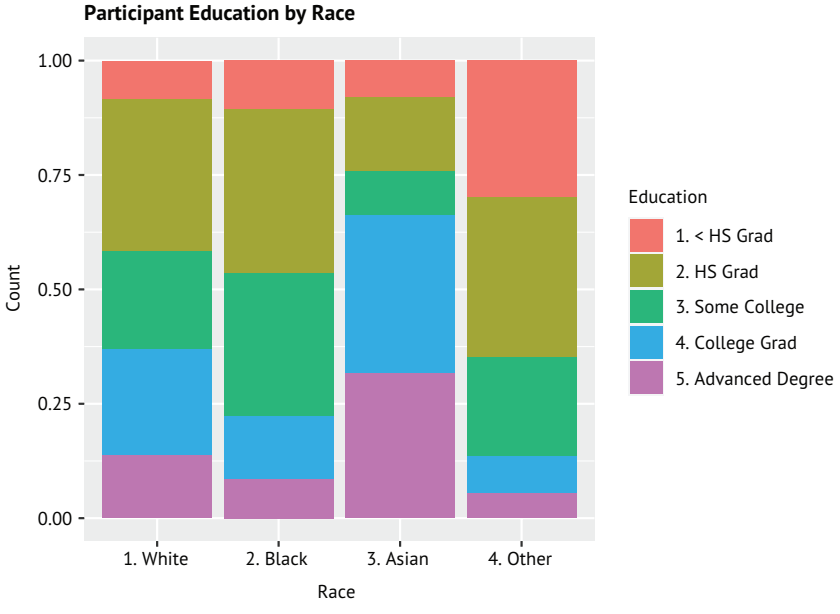


Рис. 19.3. Уровень образования в зависимости от расовой принадлежности по результатам опроса, проводившегося на северо-востоке США среди 3000 рабочих-мужчин в 2011 году

Обратите внимание, что нумерация расовой принадлежности и уровней образования фактически закодирована в наборе данных:

```
> head(Wage[c("race", "education")], 4)
           race       education
231655 1. White    1. < HS Grad
86582  1. White 4. College Grad
161300 1. White 3. Some College
155159 3. Asian 4. College Grad
```

Мы могли бы улучшить график, удалив числа из названий рас (они не являются порядковыми категориями), используя процентное форматирование по оси *y* и переупорядочив расовые категории в соответствии с процентными долями. Также можно подавить вывод категории «Other» (Другие), потому что состав этой группы неизвестен.

Изменение порядка отображения осей для категориальных переменных включает использование функций `scale_*_discrete()`. В табл. 19.3 перечислены наиболее часто используемые параметры. Переупорядочить (и/или опустить) дискретные значения можно с помощью аргумента `limit`, а изменить подписи – с помощью аргумента `labels`.

Таблица 19.3. Наиболее часто используемые параметры функций scale_*_discrete

Параметр	Описание
`name`	Название масштаба. То же, что используется в функции `labs(x = , y = )`
`breaks`	Строковый вектор с категориями для основных отметок
`limits`	Строковый вектор, определяющий категории и порядок их следования
`labels`	Строковый вектор, определяющий альтернативные подписи для отметок (должен иметь ту же длину, что и вектор `breaks`). Передача значения `labels=abbreviate` обеспечит вывод сокращенных подписей
`position`	Размещение осей (left/right для оси *y*, top/bottom для оси *x*)

Код в листинге 19.2 построит диаграмму, изображенную на рис. 19.4.

Листинг 19.4. Формирование диаграммы, отображающей уровень образования в зависимости от расовой принадлежности, с настроенными осями

```
library(ISLR)
library(ggplot2)
library(scales)
ggplot(Wage, aes(race, fill=education)) +
  geom_bar(position="fill") +
  scale_x_discrete(name = "",                                          1
                   limits = c("3. Asian", "1. White", "2. Black"),     1
                   labels = c("Asian", "White", "Black")) +            1
  scale_y_continuous(name = "Percent",                                 2
                     label = percent_format(accuracy=2),               2
                     n.breaks=10) +                                    2
  labs(title="Participant Education by Race")
```

1 **Настройка оси *x*.**

2 **Настройка оси *y*.**

Горизонтальная ось представляет категориальную переменную, поэтому она настраивается с помощью функции `scale_x_discrete()`. Категории рас переупорядочены с использованием параметра `limits`, а их подписи изменены с помощью параметра `labels`. Категория «Other» на диаграмме отсутствует, потому что она не указана в настройках. Заголовок оси удаляется путем передачи пустой строки (`""`) в параметре `name`.

Вертикальная ось представляет числовую переменную, поэтому она настраивается с помощью функции `scale_y_continuous()`: изменено название оси и подписи отметок. Функция `percent_format()` из пакета `scales` переформатирует подписи отметок на оси и отображает их в виде процентов. Аргумент `accuracy=2` задает количество значащих цифр.

Пакет `scales` часто применяется для форматирования осей. Существуют варианты форматирования денежных значений, дат, процентов, запятых, экспоненциального представления и многого другого. Подробности ищите по адресу: https://scales.r-lib.org/. Пакеты `ggh4x` и `ggprism` предоставляют дополнительные возможности для настройки осей, включая расширенную настройку основных и второстепенных отметок.

В предыдущем примере уровень образования был представлен дискретной цветовой шкалой. Поэтому далее мы рассмотрим настройку цветов.

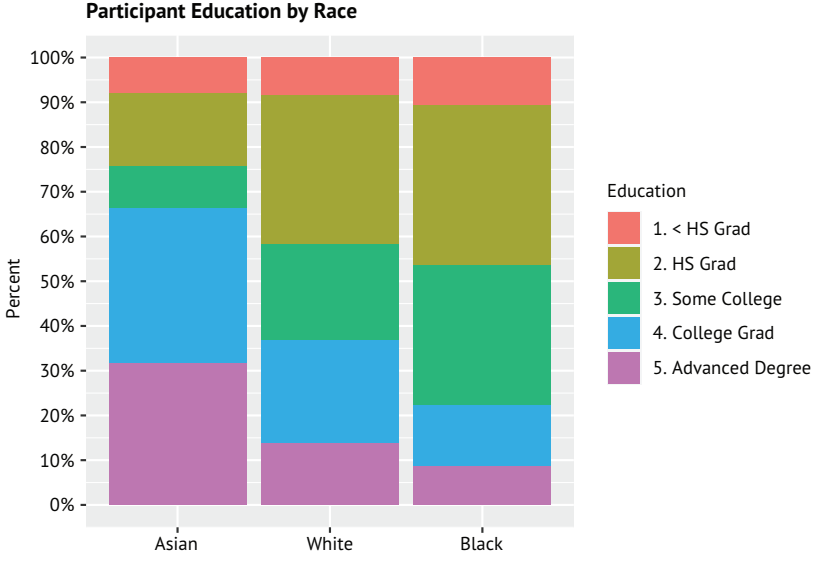


Рис. 19.4. Уровень образования в зависимости от расовой принадлежности по результатам опроса, проводившегося на северо-востоке США среди 3000 рабочих-мужчин в 2011 году. Категории, представляющие расы, были переупорядочены. Категория Other опущена. Подписи вдоль оси *x* были изменены, а значения на оси *y* изменены на проценты

19.1.2. Настройка цветов

В пакете `ggplot2` имеются функции для назначения цветовых схем категориальным и числовым переменным. Они перечислены в табл. 19.4. Функции `scale_color_*()` используются для настройки цветов точек, линий, границ и текста. Функции `scale_fill_*()` применяются для определения цвета заливки площадных объектов, таких как прямоугольники и овалы.

Цветовые схемы могут быть *последовательными*, *расходящимися* или *качественными*. Последовательные схемы используются для сопоставления цветов с монотонной числовой переменной. Расходящиеся схемы используются для числовых переменных, имеющих значимую среднюю или нулевую точку, и состоят из двух последо-

вательных схем, имеющих общую конечную точку в центральном значении. Например, расходящаяся палитра часто используется для представления значений коэффициентов корреляции (раздел 11.3). Качественная цветовая схема преобразует значения категориальной переменной в дискретные цвета.

Таблица 19.4. Функции управления цветовыми схемами

Параметр	Описание
`scale_color_gradient()` `scale_fill_gradient()`	Градиентная цветовая схема для непрерывной переменной. Принимает два параметра, определяющих начальный и конечный цвета. Имеются версии `*_gradient2()`, позволяющие задать начальный, средний и конечный цвета
`scale_color_steps()` `scale_fill_steps()`	Ступенчатая градиентная цветовая схема для непрерывной переменной. Принимает два параметра, определяющих начальный и конечный цвета. Имеются версии `*_steps2()`, позволяющие задать начальный, средний и конечный цвета
`scale_color_brewer()` `scale_fill_brewer()`	Последовательные, расходящиеся и качественные цветовые схемы от ColorBrewer (https://colorbrewer2.org). Основной аргумент `palette=`. Список цветовых схем можно получить вводом инструкции `?scale_color_brewer`
`scale_color_grey()` `scale_fill_gray()`	Последовательная черно-белая схема. Принимает необязательные параметры `start` (начальный цвет) и `end` (конечный цвет), определяющие интенсивность серого цвета. По умолчанию используются значения `0.2` и `0.8` соответственно
`scale_color_manual()` `scale_fill_manual()`	Позволяют создать свою цветовую схему для дискретной переменной, передав вектор цветов в параметре `values`
`scale_color_virdis_*` `scale_fill_virdis_*`	Цветовые схемы Viridis из пакета `viridisLite`. Помогают представить цветовую информацию пользователям с распространенными формами дальтонизма, к тому же хорошо различимую в черно-белом режиме отображения. Функции `*_d` используются для дискретных, `*_c` для непрерывных и `*_b` для категориальных переменных. Например, функция `scale_fill_viridis_d()` обеспечит выбор цветов для дискретной переменной. Параметр `option` поддерживает четыре варианта цветовой схемы (`"inferno"`, `"plasma"`, `"viridis"` (по умолчанию) и `"cividis"`)

Непрерывные цветовые схемы

Рассмотрим примеры использования цветовой схемы для отображения непрерывной количественной переменной. На рис. 19.1 была представлена диаграмма зависимости величины пробега автомобиля на одном галлоне топлива в зависимости от его массы. Давайте добавим в эту диаграмму третью переменную – объем двигателя, значения которой будут отображаться цветом. Поскольку рабочий объем двигателя является числовой переменной, определим цветовой градиент. Код в листинге 19.3 демонстрирует несколько возможностей.

Листинг 19.3. Использование цветового градиента для непрерывных переменных

```
library(ggplot2)
p <- ggplot(mtcars, aes(x=wt, y=mpg, color=disp)) +
  geom_point(shape=19, size=3) +
  scale_x_continuous(name = "Weight (1000 lbs.)",
                     n.breaks = 10,
                     minor_breaks = NULL,
                     limits=c(1.5, 5.5)) +
  scale_y_continuous(name = "Miles per gallon",
                     breaks = seq(10, 35, 5),
                     minor_breaks = seq(10, 35, 1),
                     limits = c(10, 35))

p + ggtitle("A. Default color gradient")

p + scale_color_gradient(low="grey", high="black") +
  ggtitle("B. Greyscale gradient")

p + scale_color_gradient(low="red", high="blue") +
  ggtitle("C. Red-blue color gradient")

p + scale_color_steps(low="red", high="blue") +
  ggtitle("D. Red-blue binned color Gradient")

p + scale_color_steps2(low="red", mid="white", high="blue",
                                    midpoint=median(mtcars$disp)) +
  ggtitle("E. Red-white-blue binned gradient")

p + scale_color_viridis_c(direction = -1) +
  ggtitle("F. Viridis color gradient")
```

Код создает диаграммы, изображенные на рис. 19.5. Функция `ggtitle()` эквивалентна функции `labs(title=)`, используемой в других главах этой книги. Если вы читаете черно-белую версию книги, то обязательно запустите код у себя, чтобы оценить цветовые вариации.

График A использует цветовую схему по умолчанию. График B отображает объем двигателя в оттенках серого. Графики C и D используют градиент от красного к синему. При использовании ступенчатого градиента берется непрерывный градиент и делится на дискретные значения (обычно пять). График E демонстрирует расходящийся цветовой градиент, от красного (начальный цвет) через белый (средняя точка) к синему (конечный цвет). Наконец, на графике F показан цветовой градиент Viridis. Параметр `direction = -1`, использованный при создании графика F, меняет порядок цветов на обратный, благодаря чему двигателям с большим рабочим объемом соответствует более темный цвет.

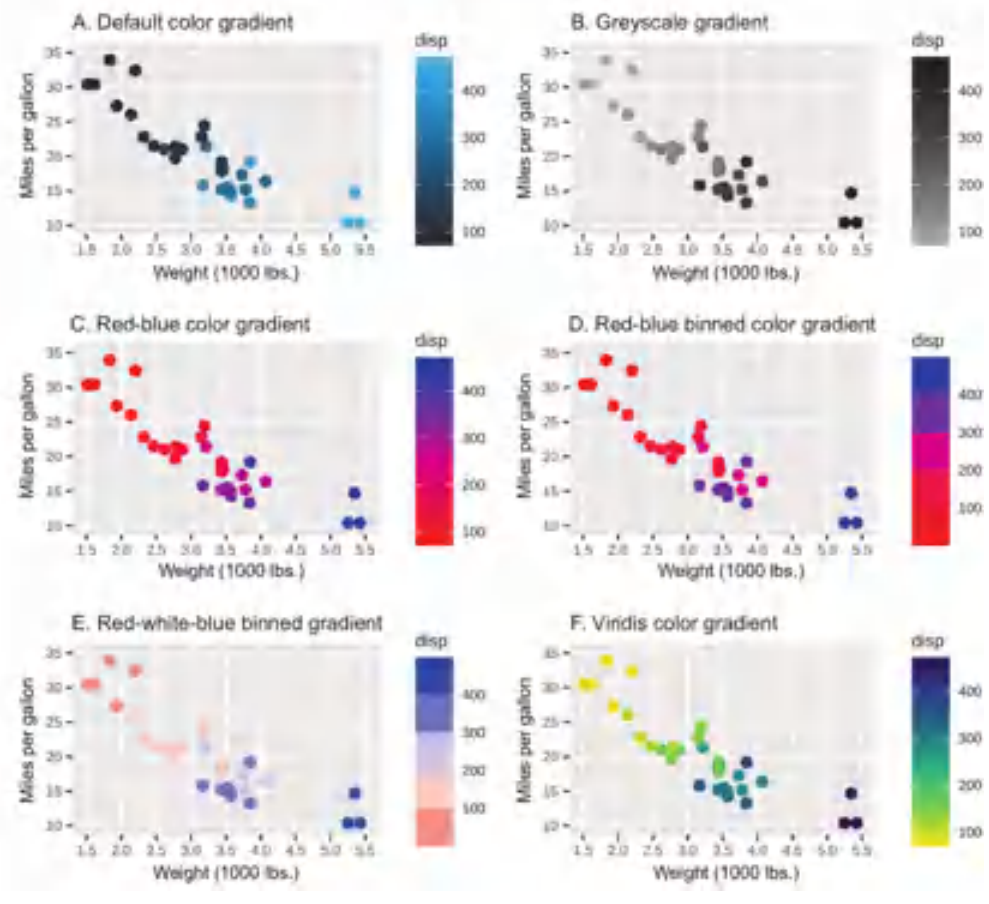


Рис. 19.5. График зависимости величины пробега автомобиля на одном галлоне от его массы. Цвет обозначает объем двигателя. Здесь представлено шесть цветовых схем. A по умолчанию. B – в оттенках серого. C и D – в красно-синей палитре, а D объединяет пять дискретных цветов. E – переход от красного к синему через белый (в середине). F – цветовая схема Viridis. На каждом графике рабочий объем двигателя увеличивается с увеличением массы автомобиля и уменьшением величины пробега

Качественные цветовые схемы

Код в листинге 19.4 демонстрирует использование качественных цветовых схем. Здесь `education` – это категориальная переменная, отображаемая дискретными цветами. На рис. 19.6 показаны получившиеся диаграммы.

Листинг 19.4. Цветовые схемы для категориальных переменных

```
library(ISLR)
library(ggplot2)
p <- ggplot(Wage, aes(race, fill=education)) +
  geom_bar(position="fill") +
  scale_y_continuous("Percent", label=scales::percent_format(accuracy=2),
                     n.breaks=10) +
  scale_x_discrete("",
                   limits=c("3. Asian", "1. White", "2. Black"),
                   labels=c("Asian", "White", "Black"))

p + ggtitle("A. Default colors")

p + scale_fill_brewer(palette="Set2") +
     ggtitle("B. ColorBrewer Set2 palette")

p + scale_fill_viridis_d() +
  ggtitle("C. Viridis color scheme")

p + scale_fill_manual(values=c("gold4", "orange2", "deepskyblue3",
                               "brown2", "yellowgreen")) +
  ggtitle("D. Manual color selection")
```

На графике A использованы цвета по умолчанию. На графике B использована качественная цветовая схема `Set2` от ColorBrewer. Также можно применять другие качественные цветовые схемы от ColorBrewer: `Accent`, `Dark2`, `Paired`, `Pastel1`, `Pastel2`, `Set1` и `Set3`. График C демонстрирует использование дискретной схемы по умолчанию от Viridis. Наконец, на графике D показана цветовая схема, определенная вручную и наглядно показывающая, что я не умею подбирать цвета самостоятельно.

Пакеты R предоставляют широкий выбор цветовых схем для использования в диаграммах `ggplot2`. Эмиль Хвитфельдт (Emil Hvitfeldt) собрал обширную коллекцию цветовых схем в своем репозитории, доступном по адресу: https://github.com/EmilHvitfeldt/r-color-palettes (почти 600 схем по последним подсчетам!). Выберите схему, которая кажется вам самой привлекательной и помогающей наиболее эффективно передать информацию, чтобы читатель мог увидеть отношения, различия, тенденции, композицию или выбросы, которые вы пытаетесь выделить.

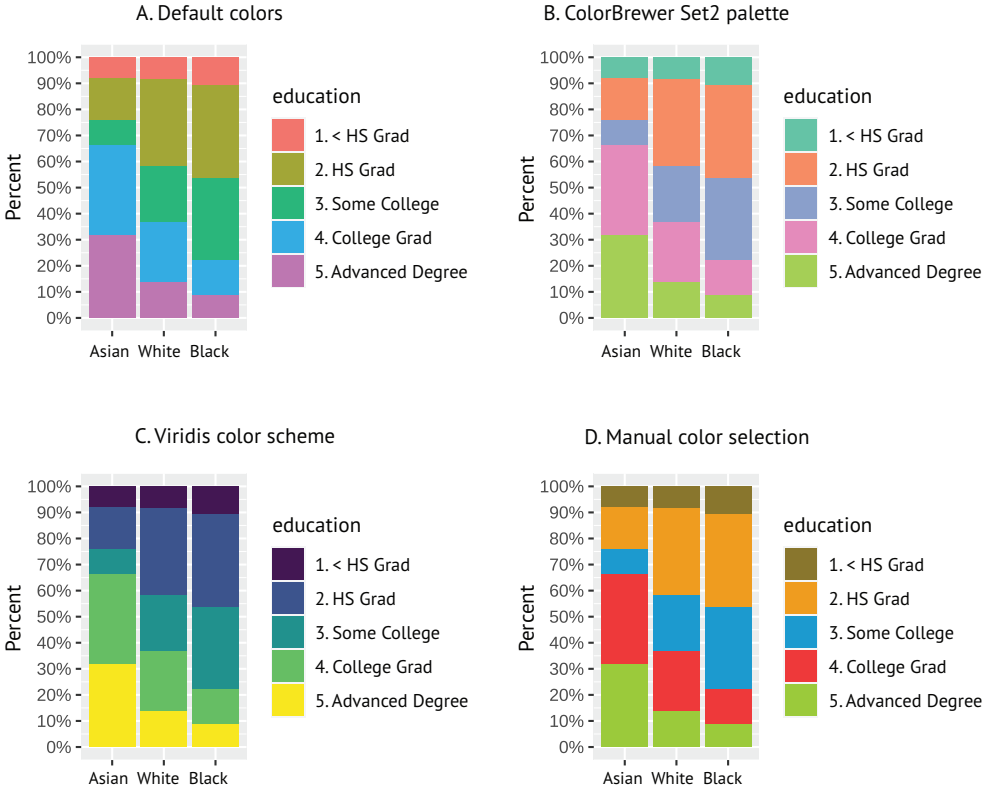


Рис. 19.6. Уровень образования в зависимости от расовой принадлежности по результатам опроса, проводившегося на северо-востоке США среди 3000 рабочих-мужчин в 2011 году. Здесь показаны четыре разные цветовые схемы. A – схема по умолчанию. B и C – предопределенные цветовые схемы. D – цветовая схема, определяемая пользователем

19.2. Изменение темы оформления

Функция `theme()` из пакета `ggplot2` позволяет настраивать визуальное оформление диаграмм. Справка для функции (`?theme`) описывает параметры, позволяющие изменить заголовок графика, подписи, шрифты, цвета фона, линий сетки и легенды.

Например, в следующем коде:

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()+
  theme(axis.title = element_text(size = 14, color = "blue"))
```

функция `theme()` обеспечит отображение заголовков по осям *x* и *y* синим шрифтом с размером в 14 пунктов. Обычно для передачи значений аргументов в `theme()` используются функции, перечисленные в табл. 19.5.

Таблица 19.5. Элементы тем

Функция	Описание
`element_blank()`	Пустой элемент (используется для удаления текста, строк и т. д.)
`element_rect()`	Задает характеристики прямоугольника. Принимает параметры `fill`, `color`, `size` и `linetype`. Последние три относятся к рамке
`element_line()`	Задает характеристики линии. Принимает параметры `color`, `size`, `linetype`, `lineend` (`"round"`, `"butt"`, `"square"`) и `arrow` (создается с помощью функции `grid::arrow()`)
`element_text()`	Задает характеристики текста. Принимает параметры `family` (семейство шрифтов), `face` (`"plain"`, `"italic"`, `"bold"`, `"bold.italic"`), `size` (размер текста в пунктах), `hjust` (выравнивание по горизонтали: `[0,1]`), `vjust` (выравнивание по вертикали: `[0,1]`), `angle` (угол поворота в градусах) и `color`

Сначала рассмотрим некоторые предопределенные темы оформления, которые одновременно изменяют множество элементов, чтобы обеспечить целостный внешний вид, а затем углубимся в настройку отдельных элементов.

19.2.1. Предопределенные темы оформления

Пакет `ggplot2` распространяется с восемью предопределенными темами оформления диаграмм, которые можно применять с помощью функций `theme_*()`. В листинге 19.5 и на рис. 19.7 показаны четыре наиболее популярные из них. Функция `theme_grey()` выбирает тему по умолчанию, а `theme_void()` создает совершенно пустую тему.

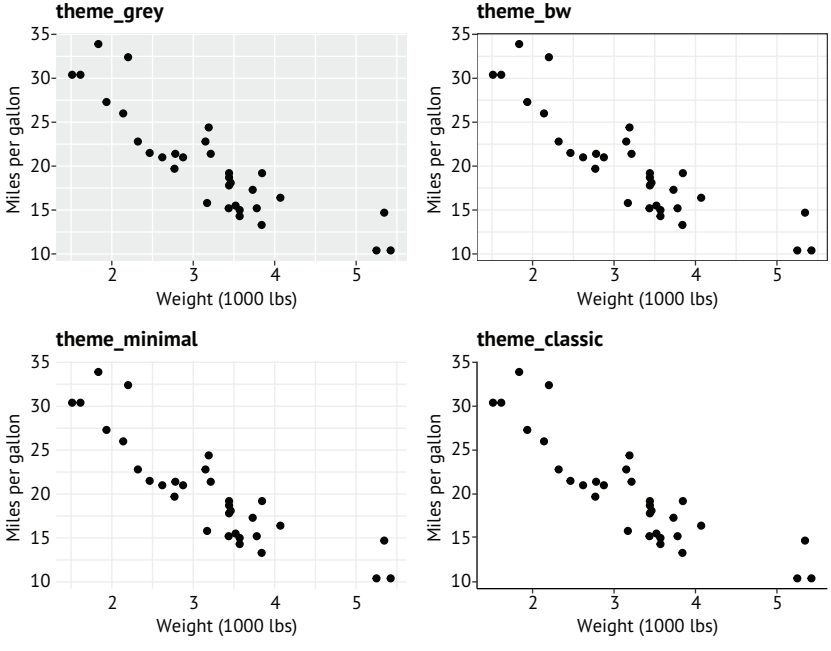


Рис. 19.7. Примеры четырех предопределенных тем оформления. По умолчанию ggplot2 использует theme_grey()

Листинг 19.5. Демонстрация четырех тем оформления, предопределенных в ggplot2

```
library(ggplot2)
p <- ggplot(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  labs(x = "Weight (1000 lbs)",
       y = "Miles per gallon")

p + theme_grey() + labs(title = "theme_grey")
p + theme_bw() + labs(title = "theme_bw")
p + theme_minimal() + labs(title = "theme_minimal")
p + theme_classic() + labs(title = "theme_classic")
```

Другие готовые темы оформления можно найти в пакетах `ggthemes`, `hbrthemes`, `xaringanthemer`, `tgamtheme`, `cowplot`, `tvthemes` и `ggdark`. Все они доступны в CRAN. Кроме того, некоторые организации предоставляют своим сотрудникам предопределенные темы, чтобы обеспечить единообразное оформление диаграмм и графиков в отчетах и презентациях.

Помимо использования предопределенных тем, можно изменять отдельные элементы темы. В следующих разделах мы будем использовать дополнительные аргументы для настройки шрифтов, легенд и иных элементов диаграмм.

19.2.2. Настройка шрифтов

Выбор типографики играет важную роль. Текст должен передавать смысл, не отвлекая и не путая читателя (http://mng.bz/5Z1q). Например, для ясности часто рекомендуются шрифты Google Roboto и Lora. Базовая версия R имеет ограниченные возможности обработки шрифтов. Пакет `showtext` значительно расширяет их, позволяя использовать в диаграммах системные шрифты и шрифты от Google.

Для этого нужно:

1) загрузить локальные шрифты и/или шрифты Google;
2) установить `showtext` в качестве устройства вывода графики;
3) указать шрифты в вызове функции `ggplot2::theme()`.

Заговорив о локальных шрифтах, должен отметить, что их местоположение, количество и типы могут сильно различаться в разных компьютерах. Чтобы использовать локальные шрифты, отличные от шрифтов, используемых в R по умолчанию, нужно знать имена и местоположение файлов шрифтов в системе. В настоящее время поддерживаются следующие форматы шрифтов: TrueType (*.ttf, *.ttc) и OpenType (*.otf).

С помощью функции `font_paths()` можно отыскать каталоги с файлами шрифтов, а с помощью `font_files()` – перечислить сами

файлы шрифтов и узнать их характеристики. В листинге 19.6 показана небольшая функция для поиска файлов шрифтов в локальной системе. В этом примере функция используется для поиска файла шрифта Comic Sans MS. В разных системах (я использую ПК с Windows) результаты могут отличаться, поэтому вы можете увидеть совсем другой вывод.

Листинг 19.6. Поиск локальных файлов шрифтов

```
> findfont <- function(x){
    suppressMessages(require(showtext))
    suppressMessages(require(dplyr))
    filter(font_files(), grepl(x, family, ignore.case=TRUE)) %>%
      select(path, file, family, face)
   }

> findfont("comic")

                path        file        family        face
1 C:/Windows/Fonts   comic.ttf Comic Sans MS     Regular
2 C:/Windows/Fonts comicbd.ttf Comic Sans MS        Bold
3 C:/Windows/Fonts  comici.ttf Comic Sans MS      Italic
4 C:/Windows/Fonts  comicz.ttf Comic Sans MS Bold Italic
```

Отыскав файл шрифта, можно вызвать `font_add()`, чтобы загрузить его. Вот пример для моей машины:

```
font_add("comic", regular = "comic.ttf",
          bold = "comicbd.ttf", italic="comici.ttf")
```

Этот код делает шрифт Comic Sans MS доступным в R под выбранным мною именем `"comic"`.

А так выглядит синтаксис вызова функции для загрузки шрифтов Google (https://fonts.google.com/):

```
font_add_google(name, family)
```

где `name` – имя шрифта Google, а `family` – произвольно выбранное имя, которое будет использоваться для ссылки на шрифт в коде. Например, следующий вызов:

```
font_add_google("Schoolbell", "bell")
```

загрузит шрифт Google Schoolbell и сделает его доступным под именем *bell*.

После загрузки шрифтов вызовите функцию `showtext_auto()`, чтобы установить `showtext` в качестве устройства вывода графики.

Наконец, используйте функцию `theme()`, чтобы задать шрифты для отображения разных элементов диаграмм. В табл. 19.6 перечислены параметры, связанные с настройкой отображения текста. С их помощью можно задать семейство шрифтов, начертание, размер, цвет и ориентацию, используя функцию `element_text()`.

Таблица 19.6. Параметры функции theme() для настройки отображения текста

Параметр	Описание
`axis.title, axis.title.x, axis.title.y`	Заголовки осей
`axis.text.*` (с теми же вариантами, что и параметр axis.title)	Подписи для отметок на осях
`legend.text, legend.title`	Текст и заголовок легенды
`plot.title, plot.subtitle, plot.caption`	Заголовок, подзаголовок и подпись для диаграммы
`strip.text, strip.text.x, strip.text.y`	Метки категорий

Листинг 19.7 демонстрирует настройку отображения текста на диаграмме с использованием двух локальных шрифтов на моей машине (Comic Sans MS и Caveat) и двух шрифтов Google (Schoolbell и Gochi Hand). Получившаяся диаграмма показана на рис. 19.8.

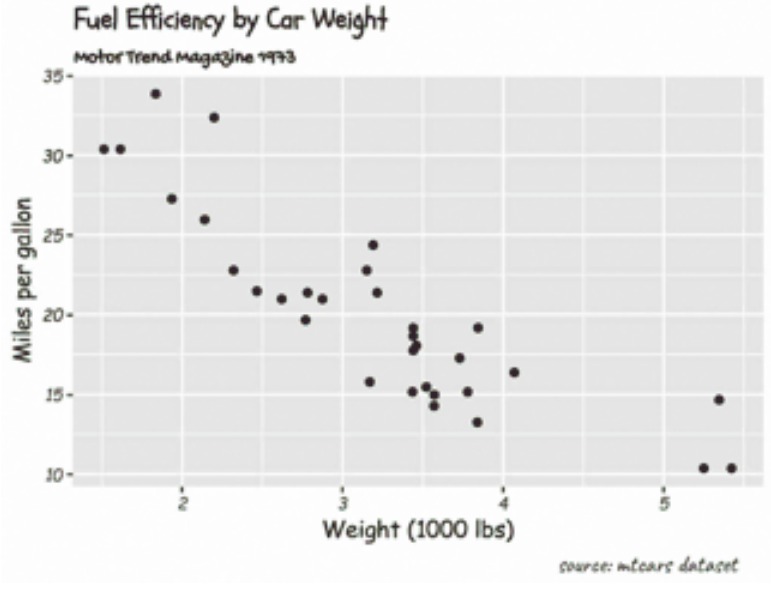


Рис. 19.8. Диаграмма с настроенными шрифтами (Schoolbell для заголовка, Gochi Hand для подзаголовка, Caveat для подписи внизу и Comic Sans MS для заголовков и подписей на осях)

Листинг 19.7. Настройка шрифтов в диаграмме ggplot2

```
library(ggplot2)
library(showtext)

font_add("comic", regular = "comic.ttf",                    1
         bold = "comicbd.ttf", italic="comici.ttf")         1
```

```
font_add("caveat", regular = "caveat-regular.ttf",                    (1)
         bold = "caveat-bold.ttf")                                    (1)

font_add_google("Schoolbell", "bell")                                 (2)
font_add_google("Gochi Hand", "gochi")                                (2)

showtext_auto()                                                       (3)

ggplot(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  labs(title = "Fuel Efficiency by Car Weight",
       subtitle = "Motor Trend Magazine 1973",
       caption = "source: mtcars dataset",
       x = "Weight (1000 lbs)",
       y = "Miles per gallon") +

   theme(plot.title    = element_text(family = "bell", size=14),      (4)
         plot.subtitle = element_text(family = "gochi"),              (4)
         plot.caption  = element_text(family = "caveat", size=15),    (4)
         axis.title    = element_text(family = "comic"),              (4)
         axis.text     = element_text(family = "comic",               (4)
                                      face="italic", size=8))         (4)
```

(1) **Загрузка локальных шрифтов.**

(2) **Загрузка шрифтов Google.**

(3) **Установка `showtext` как устройства вывода графики.**

(4) **Настройка шрифтов, используемых в диаграмме.**

Этот пример я привел исключительно в демонстрационных целях; только чтобы показать имеющиеся возможности. Использование нескольких шрифтов на одной диаграмме часто отвлекает и уводит от основной информации. Выберите один или два шрифта, которые лучше всего выделяют информацию, и придерживайтесь их. Полезным начальным руководством может послужить руководство от Тиффани Франс (Tiffany France) «Choose Fonts for Your Data Visualization» (http://mng.bz/nrY5).

19.2.3. Настройка легенды

Пакет `ggplot2` создает легенды всегда, когда для отображения переменных используются настройки цвета, заливки, формы, типов линий или размеров. При желании оформление легенды можно изменить, используя аргументы `theme()`, перечисленные в табл. 19.7.

Чаще всего используется параметр `legend.position`. Передача в этом параметре аргументов `top`, `right` (по умолчанию), `bottom` или `left` позволяет расположить легенду на любой стороне диаграммы. В качестве альтернативы можно передать двухэлементный числовой вектор (x, y), определяющий координаты легенды по осям x и y, где координата x меняется в диапазоне от 0 (слева) до 1 (справа) и координата y – в диапазоне от 0 (внизу) до 1 (вверху).

Таблица 19.7. Параметры theme() для настройки легенды диаграммы

Параметр	Описание
`legend.background`, `legend.key`	Фон и условные обозначения (символы) легенды. Задаются с помощью `element_rect()`
`legend.title`, `legend.text`	Параметры текста и заголовка легенды
`legend.position`	Позиция легенды. Допустимые значения: `"none"`, `"left"`, `"right"`, `"bottom"`, `"top"` или двухэлементный числовой вектор (с координатами от 0 (*x* – слева, *y* – внизу) до 1 (*x* – справа, *y* – вверху)
`legend.justification`	Если в `legend.postion` передается двухэлементный числовой вектор, то `legend.justification` задает *точку привязки внутри легенды* в форме двухэлементного числового вектора. Например, если `legend.position = c(1, 1)` и `legend.justification = c(1, 1)`, то точка привязки будет находиться в правом верхнем углу легенды. Внутри диаграммы точка привязки находится в правом верхнем углу
`legend.direction`	Ориентация легенды, задается одним из двух значений: `"horizontal"` или `"vertical"`
`legend.title.align`, `legend.text.align`	Выравнивание текста и заголовка легенды: число от 0 (по левому краю) до 1 (по правому краю)

Давайте создадим диаграмму рассеяния на основе таблицы данных `mtcars`. Поместим вес (`wt`) по оси *x* и пробег в милях на галлон (`mpg`) по оси *y* и раскрасим точки разными цветами, в зависимости от количества цилиндров в двигателе. Используя табл. 19.7, настроим диаграмму:

- разместив легенду в правом верхнем углу диаграммы;
- озаглавив легенду «Cylinders» (Цилиндры);
- перечислив категории легенды по горизонтали, а не по вертикали;
- установив светло-серый фон легенды и удалив фон вокруг условных обозначений (цветных символов);
- добавив белую рамку вокруг легенды.

В листинге 19.8 приводится необходимый код, а на рис. 19.9 показана получившаяся диаграмма.

Листинг 19.8. Настройка легенды диаграммы

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg, color = factor(cyl))) +
         geom_point(size=3) +
  scale_color_discrete(name="Cylinders") +
  labs(title = "Fuel Efficiency for 32 Automobiles",
       x = "Weight (1000 lbs)",
       y = "Miles per gallon") +
```

```
theme(legend.position = c(.95, .95),
      legend.justification = c(1, 1),
      legend.background = element_rect(fill = "lightgrey",
                                       color = "white",
                                       size = 1),
      legend.key = element_blank(),
      legend.direction = "horizontal")
```

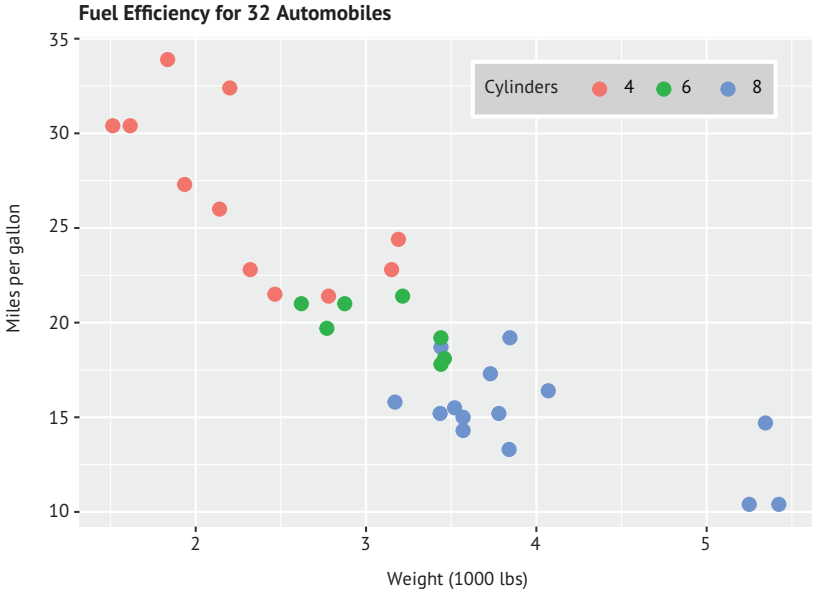


Рис. 19.9. Диаграмма с настроенной легендой. Верхний правый угол легенды помещается в верхний правый угол диаграммы. Легенда выводится по горизонтали, на сером фоне, со сплошной белой рамкой и заголовком

И снова этот пример я привел исключительно в демонстрационных целях. На самом деле легенда легче читается, когда она размещена справа и по вертикали (по умолчанию). Рекомендации по оформлению легенд можно найти в статье «Data Visualization Standards» (http://mng.bz/6m15).

19.2.4. Настройка оформления области диаграммы

Параметры функции `theme()`, перечисленные в табл. 19.8, позволяют настроить область диаграммы. Чаще всего настраиваются цвет фона и основные и второстепенные линии сетки. Код в листинге 19.9 демонстрирует настройку многих характеристик отображения области диаграммы на примере категориальной диаграммы рассеяния, а получившийся результат показан на рис. 19.10.

Таблица 19.8. Аргументы theme() для настройки оформления области диаграммы

Параметр	Описание
`plot.background`	Фон для всей диаграммы. Задается с помощью функции `element_rect()`
`plot.margin`	Поле вокруг диаграммы. Чтобы задать ширину верхнего правого, нижнего и/или левого поля, используйте функцию `units()`
`panel.background`	Фон для области с координатной сеткой. Задается с помощью функции `element_rect()`
`strip.background`	Фон для меток категорий
`panel.grid`, `panel.grid.major`, `panel.grid.minor`, `panel.grid.major.x` `panel.grid.major.y` `panel.grid.minor.x` `panel.grid.minor.y`	Линии сетки, основные (major) линии сетки, второстепенные (minor) линии сетки и основные или второстепенные линии сетки вдоль определенной оси. Задаются с помощью функции `element_line()`
`axis.line`, `axis.line.x`, `axis.line.y`, `axis.line.x.top`, `axis.line.x.bottom`, `axis.line.y.left`, `axis.line.y.right`	Линии обеих осей (`axis.line`), линии для осей по отдельности (`axis.line.x`, `axis.line.y`) или отдельные линии для каждой оси (`axis.line.x.bottom` и т. д.). Задаются с помощью функции `element_line()`

Листинг 19.9. Настройка области диаграммы

```
library(ggplot2)
mtcars$am <- factor(mtcars$am, labels = c("Automatic", "Manual"))
ggplot(data=mtcars, aes(x = disp, y = mpg)) +
  geom_point(aes(color=factor(cyl)), size=2) +                    (1)
  geom_smooth(method="lm", formula = y ~ x + I(x^2),              (2)
              linetype="dotted", se=FALSE) +
  scale_color_discrete("Number of cylinders") +
  facet_wrap(~am, ncol=2) +                                       (3)
  labs(title = "Mileage, transmission type, and number of cylinders",
       x = "Engine displacement (cu. in.)",
       y = "Miles per gallon") +
  theme_bw() +                                                    (4)
  theme(strip.background = element_rect(fill = "white"),          (5)
        panel.grid.major = element_line(color="lightgrey"),
        panel.grid.minor = element_line(color="lightgrey",
                                        linetype="dashed"),
        axis.ticks = element_blank(),
        legend.position = "top",
        legend.key = element_blank())
```

(1) **Сгруппированная диаграмма рассеяния.**

(2) **Линия аппроксимации.**

(3) **Деление на категории**

(4) **Установка черно-белой темы оформления.**

(5) **Изменение темы оформления.**

Этот код создает диаграмму с рабочим объемом двигателя (`disp`) по оси *x* и пробегом в милях на галлон (`mpg`) по оси *y*. Количество цилиндров (`cyl`) и тип трансмиссии (`am`) изначально закодированы как числовые, но для построения графика преобразованы в факторы. Для цилиндров преобразование в факторы гарантирует окрашивание точек одним цветом для каждого количества цилиндров.

Для точек на диаграмме рассеяния задаются увеличенные размеры и окраска в соответствии с количеством цилиндров (1). Затем добавляется квадратичная аппроксимирующая линия (2), имеющая один изгиб (раздел 8.2.3). Потом для каждого типа коробки передач добавляется свой график (3).

Настройка оформления начинается с вызова `theme_bw()` (4), за которым следует вызов функции `theme()` (5), изменяющий отдельные характеристики. Он устанавливает белый цвет фона, светло-серые сплошные линии для основных линий сетки и светло-серые пунктирные линии для второстепенных линий. Удаляет отметки на осях. И наконец, помещает легенду над диаграммой, а для условных обозначений в легенде устанавливает пустой фон.

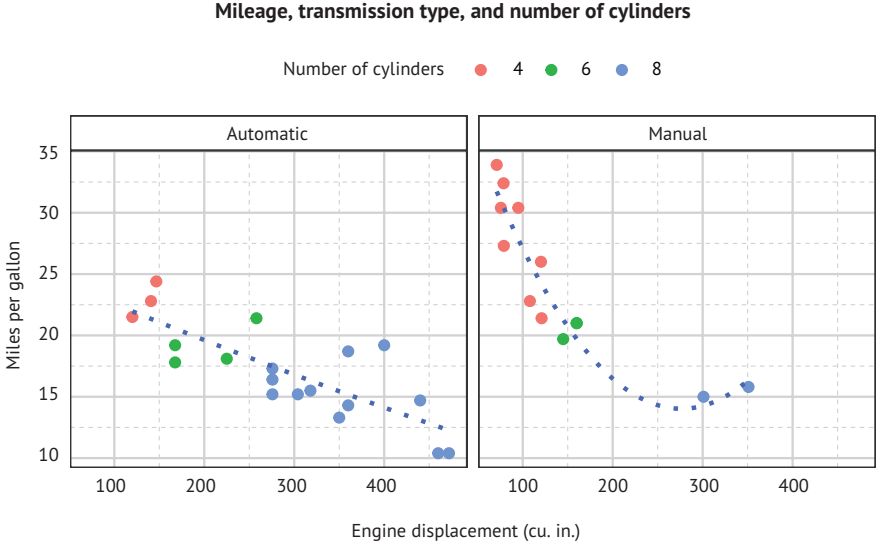


Рис. 19.10. Категориальная диаграмма рассеяния с аппроксимирующими линиями. Окончательное оформление основано на измененной версии черно-белой темы

19.3. Добавление аннотаций

Аннотации позволяют добавлять в диаграммы дополнительную информацию, упрощая распознавание взаимосвязей, распределений или необычных наблюдений. Наиболее распространенными аннотациями являются опорные линии и текстовые метки. Функции для добавления этих аннотаций перечислены в табл. 19.9.

Таблица 19.9. Функции добавления аннотаций

Функция	Описание
`geom_text`, `geom_label`	`geom_text()` добавляет текст в диаграмму; `geom_label()` действует аналогично, но дополнительно заключает текст в прямоугольную рамку
`geom_text_repel`, `geom_label_repel`	Эти функции доступны в пакете `ggrepel`. Они действуют подобно функциям `geom_text()` и `geom_label()`, но стараются исключить перекрытия текста
`geom_hline`, `geom_vline`, `geom_abline`	Добавляют дополнительные горизонтальные, вертикальные и диагональные опорные линии
`geom_rect`	Добавляет прямоугольную рамку вокруг диаграммы. Удобно использовать для выделения областей на диаграмме, на которые следует обратить внимание

Добавление подписей к точкам

Диаграмма на рис. 19.1 изображает взаимосвязь между весом автомобиля (`wt`) и потреблением топлива (`mpg`). Однако, глядя на эту диаграмму, трудно определить, какие автомобили представлены теми или иными точками, не обращаясь к исходному набору данных. Код в листинге добавляет эту информацию в диаграмму, как показано на рис. 19.11.

Листинг 19.10. Диаграмма рассеяния с подписанными точками

```
library(ggplot2)
ggplot(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point(color = "steelblue") +
  geom_text(label = row.names(mtcars)) +
  labs(title = "Fuel efficiency by car weight",
       x = "Weight (1000 lbs)",
       y = "Miles per gallon")
```

Полученную диаграмму трудно читать из-за перекрывающихся надписей. Пакет `ggrepel` устраняет этот недостаток, смещая текстовые метки, чтобы избежать наложения. Давайте построим диаграмму снова, используя этот пакет для добавления подписей точек, а также добавим контрольную линию и подпись, соответствующую медиане `MPG`. Соответствующий код показан в листинге 19.11, а сама диаграмма – на рис. 19.12.

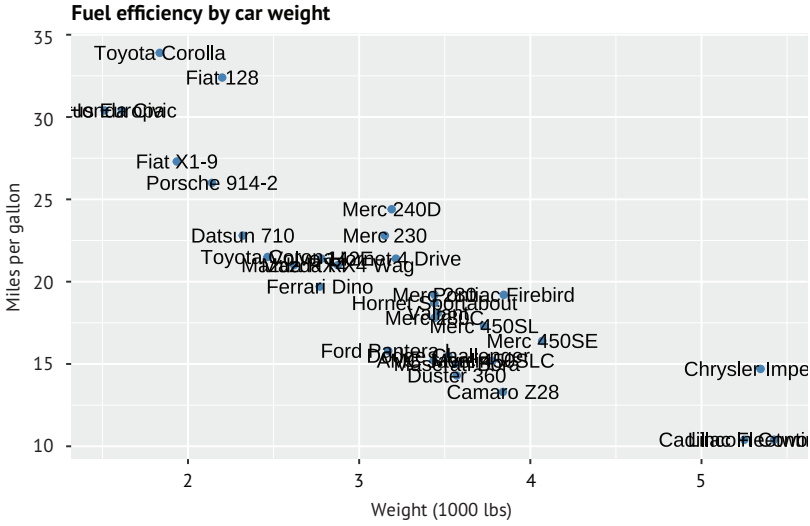


Рис. 19.11. Диаграмма рассеяния, отображающая зависимость потребления топлива от веса автомобиля. Точки подписаны названиями автомобилей

Листинг 19.11. Диаграмма рассеяния с подписями, созданными с помощью пакета ggrepel

```
library(ggplot2)
library(ggrepel)
ggplot(data = mtcars, aes(x= wt, y = mpg)) +
  geom_point(color = "steelblue") +
  geom_hline(yintercept = median(mtcars$mpg),                    (1)
             linetype = "dashed",                                (1)
             color = "steelblue") +                              (1)
  geom_label(x = 5.2, y = 20.5,                                  (2)
             label = "median MPG",                               (2)
             color = "white",                                    (2)
             fill = "steelblue",                                 (2)
             size = 3) +                                         (2)
  geom_text_repel(label = row.names(mtcars), size = 3) +         (3)
  labs(title = "Fuel efficiency by car weight",
       x = "Weight (1000 lbs)",
       y = "Miles per gallon")
```

(1) **Опорная линия.**

(2) **Подпись опорной линии.**

(3) **Подписи для точек.**

Опорная линия помогает увидеть, какие автомобили находятся выше или ниже медианы в плане экономичности (1). Подпись для линии выводится с помощью `geom_label()` (2). Для правильного размещения этой подписи (x, y) мне пришлось немного поэкспериментировать. Наконец, функция `geom_text_repel()` используется для вывода подписей точек (3). Дополнительно размер подписей

уменьшен с 4 пунктов по умолчанию до 3. Эту диаграмму намного проще читать и интерпретировать.

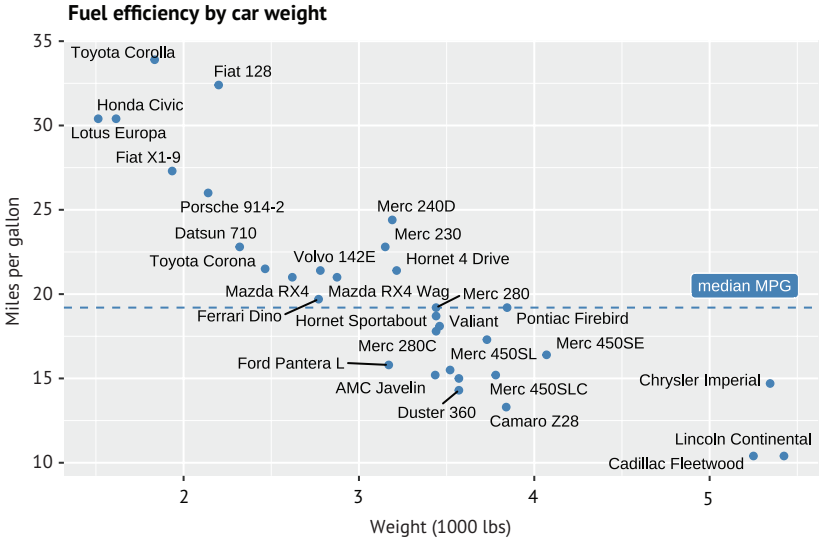


Рис. 19.12. Диаграмма рассеяния, отображающая зависимость потребления топлива от веса автомобиля. Точки подписаны названиями автомобилей. Пакет ggrepel изменил местоположения подписей так, чтобы они не перекрывались. Дополнительно в диаграмму были добавлены опорная линия и ее подпись

Подписи в столбиковых диаграммах

В столбиковые диаграммы тоже можно добавлять дополнительные метки, поясняющие распределение категориальной переменной или состав в составных столбиковых диаграммах. Добавление процентных меток к каждому столбику выполняется в два этапа. Сначала нужно вычислить проценты для каждого столбика, а затем создать столбиковую диаграмму, напрямую отображая проценты в виде меток с помощью функции `geom_text()`. Этот процесс показан в листинге 19.12, а получившаяся диаграмма – на рис. 19.13.

Листинг 19.12. Добавление процентных меток в столбиковую диаграмму

```
library(ggplot2)
library(dplyr)
library(ISLR)

plotdata <- Wage %>%                                              (1)
  group_by(race) %>%                                              (1)
  summarize(n = n()) %>%                                          (1)
  mutate(pct = n / sum(n),                                        (1)
         lbls = scales::percent(pct),                             (1)
         race = factor(race, labels = c("White", "Black",         (1)
                                        "Asian", "Other")))       (1)
```

```
plotdata
```

```
## # A tibble: 4 x 4
##   race          n    pct lbl
##   <fct>     <int>  <dbl> <chr>
## 1 1. White   2480 0.827  82.7%
## 2 2. Black    293 0.0977 9.8%
## 3 3. Asian    190 0.0633 6.3%
## 4 4. Other     37 0.0123 1.2%
```

```
ggplot(data=plotdata, aes(x=race, y=pct)) +
  geom_bar(stat = "identity", fill="steelblue") +                    (2)
  geom_text(aes(label = lbls),                                       (3)
            vjust = -0.5,                                            (3)
            size = 3) +                                              (3)
  labs(title = "Participants by Race",
       x = "",
       y="Percent") +
  theme_minimal()
```

(1) Вычисление процентов.

(2) Добавление столбиков.

(3) Добавление подписей.

Сначала вычисляются проценты для каждой расы (1), а затем создаются форматированные метки (`lbls`) с помощью функции `percent()` из пакета `scales`. Далее на основе этих сводных данных создается столбиковая диаграмма (2). Параметр `stat = "identity"` в вызове функции `geom_bar()` требует от `ggplot2` использовать предоставленные значения *y* (высоты столбиков), а не вычислять их. После этого вызывается функция `geom_text()` для вывода подписей над столбиками (3). Параметр `vjust=-0.5` немного приподнимает текст над столбиками.

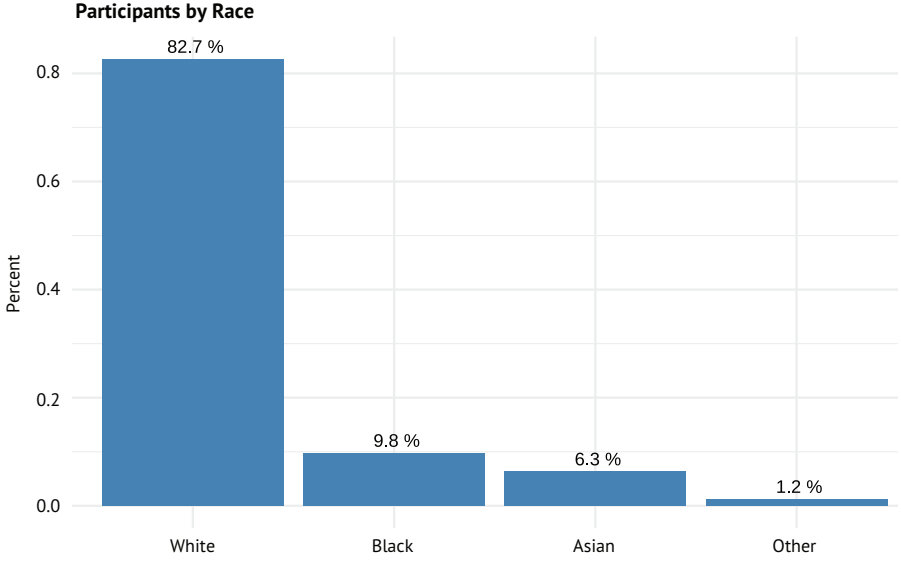


Рис. 19.13. Простая столбиковая диаграмма с процентными метками

Аналогично можно добавлять процентные метки в составные столбиковые диаграммы. Код в листинге 19.13 воссоздает диаграмму, показанную на рис. 19.4, добавляя процентные метки. Результат показан на рис. 19.14.

Листинг 19.13. Добавление процентных меток в составную столбиковую диаграмму

```
library(ggplot2)
library(dplyr)
library(ISLR)

plotdata <- Wage %>%                                                    1
  group_by(race, education) %>%                                         1
  summarize(n = n()) %>%                                                1
  mutate(pct = n/sum(n),                                                1
         lbl = scales::percent(pct))                                    1

ggplot(plotdata, aes(x=race, y=pct, fill=education)) +
  geom_bar(stat = "identity",
           position="fill",
           color="lightgrey") +
  scale_y_continuous("Percent",                                         2
                     label=scales::percent_format(accuracy=2),          2
                     n.breaks=10) +                                     2
  scale_x_discrete("",                                                  2
                   limits=c("3. Asian", "1. White", "2. Black"),        2
                   labels=c("Asian", "White", "Black")) +               2
  geom_text(aes(label = lbl),                                           3
            size=3,                                                     3
            position = position_stack(vjust = 0.5)) +                   3
  labs(title="Participant Education by Race",
       fill = "Education") +
  theme_minimal() +                                                     4
  theme(panel.grid.major.x=element_blank())                             4
```

1 **Вычисление процентов.**

2 **Настройка осей *y* и *x*.**

3 **Добавление процентных меток.**

4 **Настройка оформления.**

Этот подобен предыдущему. Он вычисляет проценты для каждой расы по комбинации образования 1 и строит столбиковые диаграммы на основе этих процентов. Оси *x* и *y* настраиваются по аналогии с листингом 19.2 2. Затем вызывается функция `geom_text()`, чтобы добавить процентные метки 3. Функция `position_stack()` обеспечивает правильное размещение меток для всех сегментов. В заключение производится настройка оформления и заголовков, а также выбирается минималистская тема 4 без линий сетки по оси *x* (они здесь не нужны).

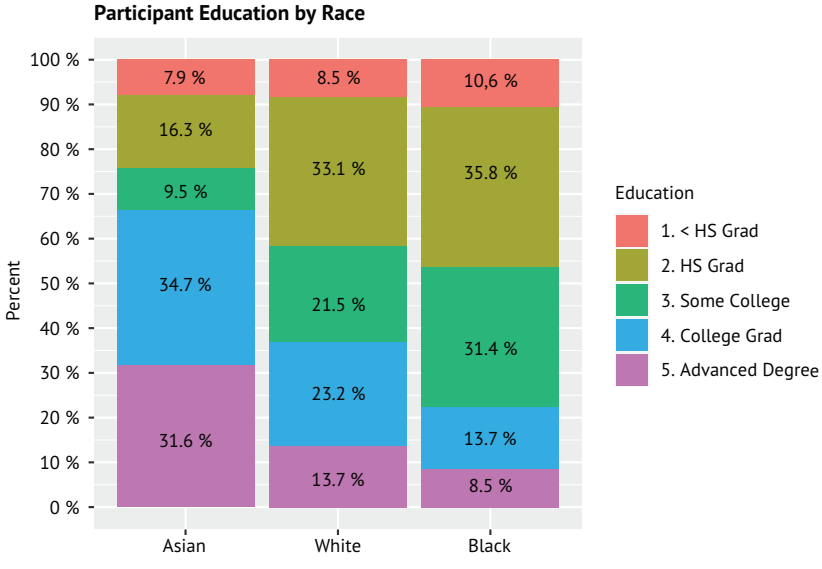


Рис. 19.14. Составная столбиковая диаграмма (с заливкой) с процентными метками

Выделение деталей

В заключительном примере демонстрируется использование аннотаций для выделения информации в сложной диаграмме. Таблица данных `gapminder` в пакете `gapminder` содержит среднегодовую ожидаемую продолжительность жизни для 142 стран, регистрируемую каждые 5 лет с 1952 по 2002 год. Ожидаемая продолжительность жизни в Камбодже сильно отличается от других азиатских стран в этом наборе данных. Давайте нарисуем диаграмму, подчеркивающую это отличие. Необходимый для этого код показан в листинге 19.14, а получившаяся диаграмма – на рис. 19.15.

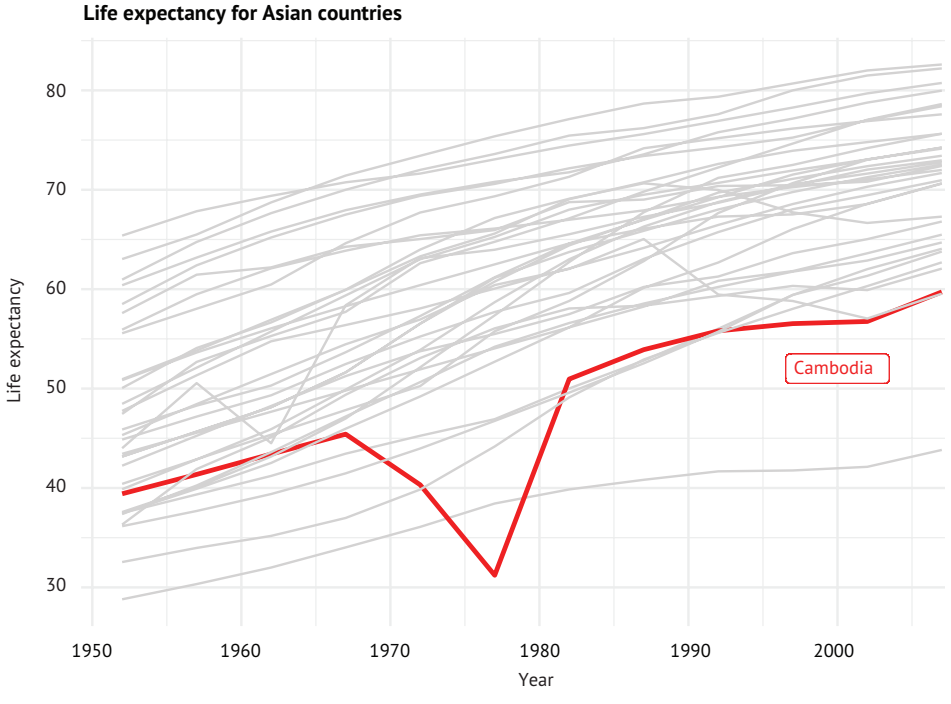


Рис. 19.15. Тенденции изменения средней продолжительности жизни в 33 азиатских странах. Выделена тенденция для Камбоджи. Несмотря на общую положительную направленность для всех стран, в Камбодже с 1967 по 1977 год наблюдался резкий спад

Листинг 19.14. Выделение одной тенденции среди многих

```
library(ggplot2)
library(dplyr)
library(gapminder)
plotdata <- gapminder %>%
  filter(continent == "Asia")                                    1

plotdata$highlight <- ifelse(plotdata$country %in%               2
                             c("Cambodia"), "y", "n")            2

ggplot(plotdata, aes(x = year, y = lifeExp,                      3
                     group = country,                            3
                     size = highlight,                           3
                     color = highlight)) +                       3
  scale_color_manual(values=c("lightgrey", "red")) +             3
  scale_size_manual(values=c(.5, 1)) +                           3
  geom_line() +                                                  3
  geom_label(x=2000, y= 52, label="Cambodia",                    4
             color="red", size=3) +                              4
  labs(title="Life expectancy for Asian countries",
       x="Year",
```

```
      y="Life expectancy") +
  theme_minimal() +
  theme(legend.position="none",
        text=element_text(size=10))
```

1 Выделение подмножества стран Азии.

2 Создание индикаторной переменной для Камбоджи.

3 Визуальное выделение линии тренда, представляющей Камбоджу.

4 Добавление поясняющей метки.

Сначала из общего набора данных выделяется подмножество азиатских стран 1. Затем создается двоичная переменная, служащая признаком Камбоджи 2. Далее для каждой страны строятся линии тренда ожидаемой продолжительности 3. Толщина линии тренда, соответствующей Камбодже, увеличивается, и ей назначается красный цвет. Линии трендов для остальных стран окрашены в светло-серый цвет. Также для линии камбоджийского тренда добавляется метка 4. В заключение выводятся подписи осей и отметок на осях, а также задается минималистская тема оформления. Легенда (размер, цвет) скрыта (она просто не нужна), размер основного текста уменьшен.

Глядя на график, видно, что средняя продолжительность жизни во всех странах последовательно увеличивалась. Но в тренде для Камбоджи наблюдается большой спад между 1967 и 1977 годами. Вероятно, это связано с геноцидом Пол Пота и красных кхмеров, имевшим место в этот период.

19.4. Объединение диаграмм

Объединение диаграмм `ggplot2` в одну общую диаграмму часто помогает подчеркнуть взаимосвязи и различия. Я использовал этот прием в нескольких местах в книге (например, рис. 19.7). Для объединения диаграмм можно использовать простой, но мощный пакет `patchwork`. Чтобы его применить, нужно сохранить каждую диаграмму `ggplot2` в виде объекта, а затем объединить их по горизонтали с помощью оператора вертикальной черты (`|`) и/или по вертикали с помощью оператора косой черты (`/`). Для создания подгрупп диаграмм можно использовать круглые скобки `()`. На рис. 19.16 показаны различные схемы объединения диаграмм.

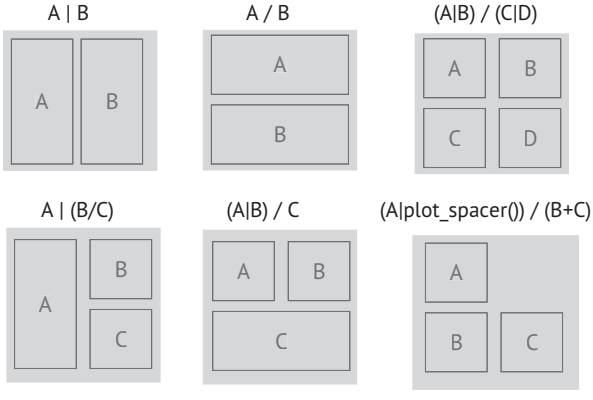


Рис. 19.16. Пакет patchwork реализует поддержку простого набора арифметических символов для объединения нескольких диаграмм

Давайте создадим несколько диаграмм на основе данных из набора `mtcars` и объединим их. Код в листинге 19.15 создает диаграмму, изображенную на рис. 19.17.

Листинг 19.15. Объединение диаграмм с помощью пакета patchwork

```
library(ggplot2)
library(patchwork)

p1 <- ggplot(mtcars, aes(disp, mpg)) +                          1
  geom_point() +                                                1
  labs(x="Engine displacement",                                 1
       y="Miles per gallon")                                    1

p2 <- ggplot(mtcars, aes(factor(cyl), mpg)) +                   1
  geom_boxplot() +                                              1
  labs(x="Number of cylinders",                                 1
       y="Miles per gallon")                                    1

p3 <- ggplot(mtcars, aes(mpg)) +                                1
  geom_histogram(bins=8, fill="darkgrey", color="white") +      1
  labs(x = "Miles per gallon",                                  1
       y = "Frequency")                                         1

(p1 | p2) / p3 +                                                2
  plot_annotation(title = 'Fuel Efficiency Data') &             2
  theme_minimal() +                                             2
  theme(axis.title = element_text(size=8),                      2
        axis.text = element_text(size=8))                       2
```

1 **Создание трех отдельных диаграмм.**

2 **Объединение в одну диаграмму.**

Сначала создаются три отдельные диаграммы и сохраняются в переменных p1, p2 и p3 (1). Код (p1 | p2)/p3 указывает, что первые две диаграммы должны быть размещены в первой строке, а третий график – занимать всю вторую строку (2).

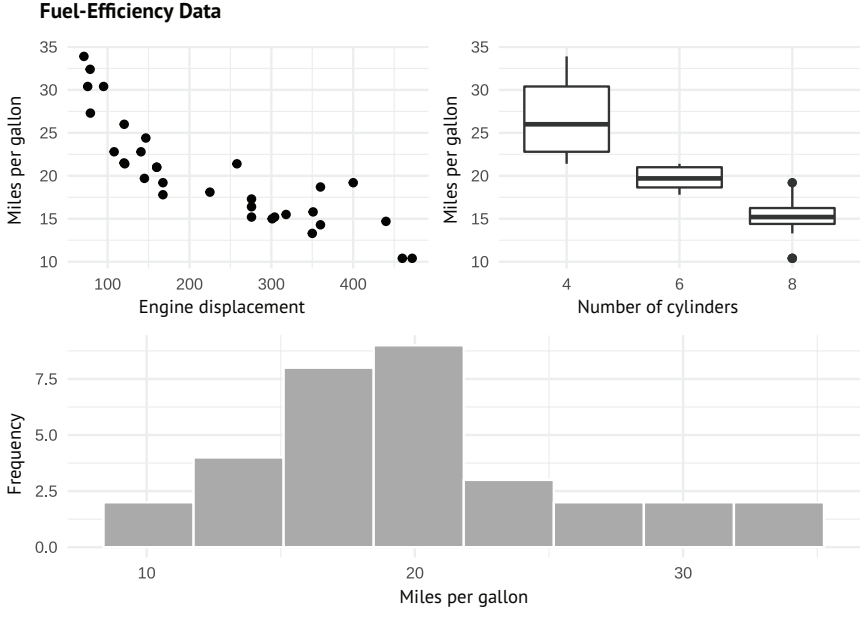


Рис. 19.17. Три диаграммы ggplot2, объединенные в одну с помощью пакета patchwork

Получившаяся диаграмма тоже является диаграммой ggplot2 и может быть отредактирована. Например, вызов функции plot_annotation() в листинге 19.15 добавляет заголовок в объединенную диаграмму, а не в одну из исходных диаграмм. Также можно изменить тему оформления объединенной диаграммы. Обратите внимание на использование амперсанда (&) при добавлении элементов темы. Если использовать знак плюс (+), то изменения будут применяться только к *последней* поддиаграмме (p3). Знак & указывает, что функции изменения оформления должны применяться к каждой поддиаграмме (p1, p2 и p3).

Пакет patchwork предлагает довольно широкий спектр возможностей. Узнать больше об этих возможностях можно на сайте пакета https://patchwork.data-imaginist.com/.

19.5. Создание интерактивных диаграмм

За редкими исключениями, диаграммы в этой книге представляют собой статические изображения. Однако иногда бывает желательно создавать *интерактивные диаграммы*, позволяющие сосредото-

читься на интересных результатах и вызывать дополнительную информацию для изучения закономерностей, тенденций и необычных наблюдений. Кроме того, интерактивные диаграммы часто более привлекательны, чем статические графики.

В R есть несколько пакетов, которые можно использовать для создания интерактивных визуализаций, включая `leaflet`, `rbokeh`, `rCharts`, `highlighter` и `plotly`, но в этом разделе мы сосредоточимся только на `plotly`.

Графическую библиотеку `plotly` *с открытым исходным кодом для R* (https://plotly.com/r/) можно использовать для создания высококачественных интерактивных визуализаций. Ключевым ее преимуществом является способность преобразовывать статические диаграммы `ggplot2` в интерактивную веб-графику.

Создание интерактивной диаграммы с помощью пакета `plotly` – это простой двухэтапный процесс. Сначала нужно сохранить диаграмму `ggplot2` как объект, а затем передать этот объект функции `ggplotly()`.

Код в листинге 19.16 создает диаграмму рассеяния, отражающую зависимость между величиной пробега в милях на галлон от объема двигателя и созданную с помощью `ggplot2`. Точки окрашены в соответствии с количеством цилиндров двигателя. Затем график передается функции `ggplotly()` из пакета `plotly`, чтобы получить интерактивную веб-форму, изображение которой показано на рис. 19.18.

Листинг 19.16. Преобразование диаграммы ggplot2 в интерактивную веб-форму

```
library(ggplot2)
library(plotly)
mtcars$cyl <- factor(mtcars$cyl)
mtcars$name <- row.names(mtcars)

p <- ggplot(mtcars, aes(x = disp, y= mpg, color = cyl)) +
  geom_point()
ggplotly(p)
```

При наведении указателя мыши на диаграмму вверху слева появляется панель инструментов, позволяющая масштабировать, панорамировать, выбирать области, сохранять изображение и т. д. (рис. 19.19). Кроме того, по мере перемещения указателя в области графика будут появляться всплывающие подсказки. По умолчанию в таких подсказках отображаются значения, использованные при построении диаграммы (в данном примере это значения переменных `disp`, `mpg` и `cyl`). Кроме того, щелчок на условном обозначении (значке) в легенде будет включать и выключать соответствующие данные. Это позволяет сосредоточиться на интересующих вас подмножествах данных.

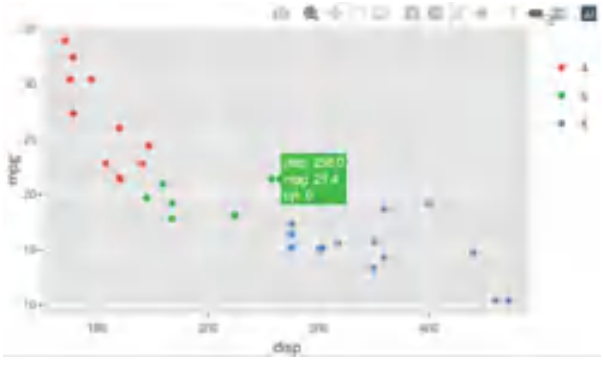


Рис. 19.18. Скриншот интерактивной веб-формы, созданной с помощью пакета plotly из статической диаграммы ggplot2

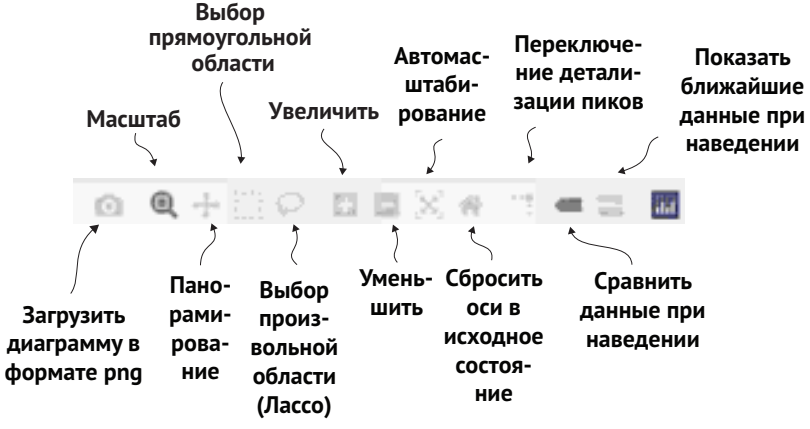


Рис. 19.19. Панель инструментов в диаграмме plotly. Самый эффективный способ освоить эти инструменты – попробовать их

Есть два простых способа настроить всплывающие подсказки. Можно добавить дополнительные переменные, включив `label1 = var1, label2 = var2` и т. д. в вызов функции `ggplot::aes()`. Например, следующий код:

```
p <- ggplot(mtcars, aes(x = disp, y= mpg, color = cyl,
                        label1 = gear, label2 = am))) +
  geom_point()
ggplotly(p)
```

создаст всплывающую подсказку со значениями `disp`, `mpg`, `cyl`, `gear` и `am`.

Или использовать недокументированный аргумент `text` в вызове функции `aes()` для создания всплывающей подсказки из произвольной текстовой строки. Как это сделать, показано в листинге 19.17, а результат изображен на рис. 19.20.

Листинг 19.17. Настройка всплывающей подсказки

```
library(ggplot2)
library(plotly)
mtcars$cyl <- factor(mtcars$cyl)
mtcars$name <- row.names(mtcars)

p <- ggplot(mtcars,
            aes(x = disp, y=mpg, color=cyl,
                text = paste(name, "\n",
                             "mpg:", mpg, "\n",
                             "disp:", disp, "\n",
                             "cyl:", cyl, "\n",
                             "gear:", gear))) +
  geom_point()
ggplotly(p, tooltip=c("text"))
```

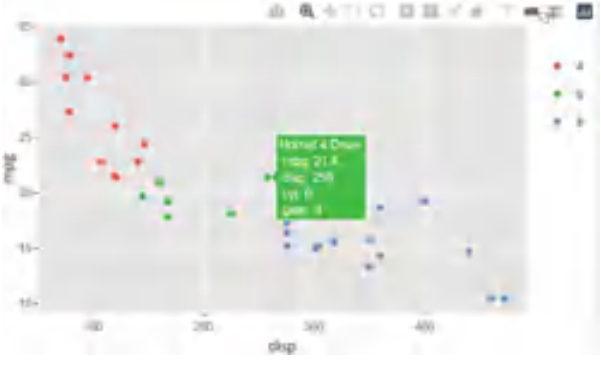


Рис. 19.20. Скриншот интерактивной веб-формы с настроенной всплывающей подсказкой

Прием с аргументом `text` дает больше возможностей в формировании всплывающей подсказки. Он позволяет включить разметку HTML в текстовую строку и дополнительно настроить отображаемый текст.

В этой главе мы рассмотрели различные методы настройки диаграмм `ggplot2`. Помните, что главная цель применения всех этих методов – помощь в исследовании данных и в передаче ваших идей другим. Все дополнения, не подчиненные этой цели, являются ненужной мишурой. Всегда старайтесь избегать мишуры в диаграммах! Дополнительные рекомендации вы найдете в статье «Data Visualization Standards» (https://xdgov.github.io/data-design-standards/).

Итоги

- Функции масштабирования в пакете `ggplot2` сопоставляют значения переменных с визуальными аспектами графика. Они особенно полезны для настройки осей и цветовой палитры.

- Функция `ggplot2::theme()` управляет настройками визуального оформления диаграмм. С ее помощью можно настроить внешний вид шрифтов, легенд, линий сетки и фона диаграммы.
- Функции `geom` удобно использовать для аннотирования диаграмм путем добавления полезной информации, такой как опорные линии и метки.
- Две и более диаграмм можно объединить в одну с помощью пакета `patchwork`.
- Практически любую диаграмму `ggplot2` можно преобразовать из статического изображения в интерактивную веб-форму с помощью пакета `plotly`.

20 Продвинутые приемы программирования

В этой главе:

- погружение в язык R;
- использование средств ООП в R для создания обобщенных функций;
- оптимизация кода;
- поиск и исправление ошибок.

В предыдущих главах мы рассмотрели различные аспекты разработки приложений, включая типы данных (раздел 2.2), управление потоком выполнения (раздел 5.4) и создание функций (раздел 5.5). В этой главе мы вновь вернемся к этим аспектам и рассмотрим их более детально, чтобы получить еще более полное представление о работе языка R. Это поможет вам создавать свои функции и в конечном счете собственные пакеты.

Наше обсуждение мы начнем с обзора объектов, типов данных и конструкций управления потоком выполнения, а затем перейдем к деталям создания функций, включая роль области видимости и окружения. В этой главе представлен подход к объектно-ориентированному программированию, принятый в R, и обсуждается

создание обобщенных функций. Здесь также даются советы, которые помогут вам писать эффективный код и отлаживать его. Эти новые знания помогут вам разбираться в коде приложений, написанных другими, и писать новые приложения. В главе 22 у вас будет возможность применить эти навыки на практике и создать полезный пакет от начала до конца.

20.1. Обзор языка

R – это объектно-ориентированный и одновременно функциональный язык программирования, предназначенный для обработки массивов данных, в котором объекты являются специализированными структурами данных, хранящимися в оперативной памяти и доступными через имена, или символы . Имена объектов состоят из прописных и строчных букв, цифр 0–9, точки и нижнего подчеркивания. Имена чувствительны к регистру и не могут начинаться с цифры, а точка рассматривается как простой символ, не имеющий особого значения.

В отличие от таких языков, как C и C++, R не позволяет напрямую обращаться к ячейкам памяти. Данные, функции и почти все остальное, что можно сохранить под некоторым именем, являются объектами. Кроме того, имена и символы сами являются объектами, которыми можно манипулировать. Во время выполнения программы все объекты сохраняются в оперативной памяти, что имеет большое значение для анализа массивных наборов данных.

Каждый объект имеет атрибуты: метаинформацию, описывающую характеристики объекта. Атрибуты можно получить с помощью функции `attributes()` и изменить с помощью функции `attr()`. Ключевым атрибутом является класс объекта. Функции R используют информацию о классе, чтобы определить, как обрабатывать объект. Класс объекта можно получить и изменить с помощью функции `class()`. Соответствующие примеры вы увидите далее в этой и в следующей главах.

20.1.1. Типы данных

Существует два основных типа данных: атомарные векторы и обобщенные векторы. Атомарные векторы – это массивы, содержащие данные одного типа. Обобщенные векторы, также называемые списками, содержат наборы атомарных векторов. Списки рекурсивны, т. е. могут содержать другие списки. В этом разделе мы подробно рассмотрим оба этих типа.

В отличие от многих других языков программирования, в R не нужно объявлять тип данных объекта или выделять для него место. Тип определяется неявно по содержимому объекта, а размер увеличивается или уменьшается автоматически в зависимости от типов и количества элементов, содержащихся в объекте.

Атомарные векторы

Атомарные векторы – это массивы, содержащие данные одного типа (логические, действительные, комплексные, символьные или двоичные). Например, все следующие массивы являются одномерными атомарными векторами:

```
passed <- c(TRUE, TRUE, FALSE, TRUE)
ages <- c(15, 18, 25, 14, 19)
cmplxNums <- c(1+2i, 0+1i, 39+3i, 12+2i)
names <- c("Bob", "Ted", "Carol", "Alice")
```

Двоичные векторы хранят простые байты и далее обсуждаться не будут.

Многие типы данных в R являются атомарными векторами со специальными атрибутами. Например, в R нет скалярных типов. Скаляр – это атомарный вектор с одним элементом. Соответственно, код `k <- 2` является лишь краткой формой записи `k <- c(2)`.

Матрица – это атомарный вектор с атрибутом размерности `dim`, содержащим два элемента (количество строк и количество столбцов). Например, создадим одномерный числовой вектор `x`:

```
> x <- c(1,2,3,4,5,6,7,8)
> class(x)
[1] "numeric"
> print(x)
{1] 1 2 3 4 5 6 7 8
```

Затем добавим атрибут `dim`:

```
> attr(x, "dim") <- c(2,4)
```

После этого объект `x` превратится в матрицу 2 3 – объект класса `matrix`:

```
> print(x)
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

> class(x)
[1] "matrix" "array"
> attributes(x)
$dim
[1] 2 2
```

Строкам и столбцам матрицы можно дать имена, установив атрибут `dimnames`:

```
> attr(x, "dimnames") <- list(c("A1", "A2"),
                              c("B1", "B2", "B3", "B4"))
> print(x)
   B1 B2 B3 B4
A1  1  3  5  7
A2  2  4  6  8
```

Наконец, матрицу можно преобразовать обратно в одномерный вектор, удалив атрибут `dim`:

```
> attr(x, "dim") <- NULL
> class(x)
[1] "numeric"
> print(x)
[1] 1 2 3 4 5 6 7 8
```

Массив – это атомарный вектор с атрибутом `dim`, содержащим три и более элементов. Так же как и в случае с матрицами, измерениям массива можно давать свои имена, устанавливая атрибут `dimnames`. Матрицы и массивы, как и одномерные векторы, могут быть логическими, числовыми, символьными, комплексными или двоичными типами. Но все элементы в одной матрице или массиве должны быть одного типа.

Функция `attr()` позволяет создавать произвольные атрибуты и связывать их с объектом. Атрибуты хранят дополнительную информацию об объекте и могут использоваться функциями для определения способов их обработки.

Существует ряд специальных функций для установки атрибутов, включая `dim()`, `dimnames()`, `name()`, `row.names()`, `class()` и `tsp()`. Последняя используется для создания объектов временных рядов. Эти специальные функции имеют ограничения на устанавливаемые значения. При добавлении к объектам стандартных атрибутов предпочтительнее использовать эти специальные функции. Встроенные в них проверки и сообщения об ошибках сделают ошибки программирования менее вероятными и более очевидными.

Обобщенные векторы, или списки

Списки – это наборы атомарных векторов и/или других списков. Таблицы данных – это особый тип списка, в котором все атомарные векторы имеют одинаковую длину. Рассмотрим таблицу данных `iris`, входящую в состав дистрибутива R. В ней описываются четыре физических параметра 150 растений вместе с их видами (сетоза, лишай или виргиника):

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Эта таблица данных на самом деле является списком, содержащим пять атомарных векторов. Он имеет атрибут `name` (текстовый вектор с именами переменных), атрибут `row.names` (числовой век-

тор, идентифицирующий отдельные растения) и атрибут `class` со значением `"data.frame"`. Каждый вектор представляет столбец (переменную) в таблице данных. В этом легко убедиться, если вывести содержимое таблицы с помощью функции `unclass()` и получить атрибуты с помощью функции `attribute()`:

```
unclass(iris)
attributes(iris)
```

Вывод этого кода я опустил для экономии места.

Знать и понимать, как работают списки, очень важно, потому что они часто возвращаются функциями R. Давайте рассмотрим пример с использованием метода кластерного анализа из главы 16. Кластерный анализ – это семейство методов для выявления естественных групп наблюдений.

Применим кластерный анализ методом *k*-средних (раздел 16.4.1) к данным `iris`. Предположим, что в данных присутствуют три кластера, и посмотрим, как группируются наблюдения (строки). Мы проигнорируем переменную, определяющую вид, и используем только физические характеристики каждого растения:

Какая информация хранится в объекте `fit`? На странице справки для `kmeans()` указано, что функция возвращает список с семью элементами. Функция `str()` отображает структуру объекта, а функцию `unclass()` можно использовать для непосредственного изучения содержимого объекта. Функция `length()` сообщает число элементов в объекте, а функция `named()` – имена этих компонентов. Также для проверки атрибутов объекта можно использовать функцию `attributes()`. Далее приводится пример исследования содержимого объекта, возвращаемого `kmeans()`:

```
> names(fit)
[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"

> unclass(fit)
$cluster
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [29] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 2 2 2
 [57] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2
 [85] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3 3
[113] 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 2 3 3 3 3 3 2 3 3 3 3 2 3
[141] 3 3 2 3 3 3 2 3 3 2

$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1        5.006       3.428        1.462       0.246
2        5.902       2.748        4.394       1.434
3        6.850       3.074        5.742       2.071
```

```
$totss
[1] 681.4

$withinss
[1] 15.15 39.82 23.88

$tot.withinss
[1] 78.85

$betweenss
[1] 602.5

$size
[1] 50 62 38

$iter
[1] 2

$ifault
[1] 0
```

Вызовом `sapply(fit, class)` можно получить класс каждого компонента объекта:

```
> sapply(fit, class)
    cluster     centers        totss     withinss tot.withinss
  "integer"    "matrix"    "numeric"    "numeric"    "numeric"
  betweenss        size        iter       ifault
  "numeric"   "integer"   "integer"   "integer"
```

В этом примере `cluster` – это целочисленный вектор, определяющий принадлежность к тому или иному кластеру, а `centers` – матрица, содержащая центроиды кластера (средние значения каждой переменной для каждого кластера). Компонент `size` – это целочисленный вектор, содержащий количество растений в каждом из трех кластеров. Чтобы получить информацию о других компонентах, загляните в раздел Value в справке `help(kmeans)`.

Индексирование

Умение извлекать информацию из списка – важный навык программирования на R. Элементы из любого объекта данных можно извлечь с помощью индексов. Прежде чем погрузиться в список, рассмотрим извлечение элементов из атомарного вектора.

Элементы извлекаются с использованием синтаксиса *`object`*`[`*`index`*`]`, где *`object`* – это вектор, а *`index`* – целочисленный вектор. Если элементам атомарного вектора присвоены имена, то *`index`* может также быть текстовым вектором с этими именами. Обратите внимание, что в R индексы начинаются с `1`, а не с `0`, как во многих других языках.

Вот пример извлечения значений из атомарного вектора без именованных элементов:

```
> x <- c(20, 30, 40)
> x[3]
[1] 40
> x[c(2,3)]
[1] 30 40
```

А так можно извлечь значения из атомарного вектора с именованными элементами:

```
> x <- c(A=20, B=30, C=40)
> x[c(2,3)]
 B  C
30 40
> x[c("B", "C")]
 B  C
30 40
```

Из списков их компоненты (атомарные векторы или другие списки) можно извлечь, используя синтаксис *object*[*index*], где *index* – целочисленный вектор. Если компоненты имеют имена, то можно также использовать текстовый вектор с именами.

Продолжим пример с методом *k*-средних. Следующий код:

```
>fit[c(2, 7)]
$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053

$size
[1] 50 62 38
```

вернет список с двумя компонентами (средние значения и размеры кластеров). Каждый компонент содержит матрицу.

Важно отметить, что применение пары квадратных скобок к списку всегда возвращает список. Например:

```
>fit[2]
$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.006000    3.428000     1.462000    0.246000
2     5.901613    2.748387     4.393548    1.433871
3     6.850000    3.073684     5.742105    2.071053
```

Этот код возвращает список с одним компонентом, а не матрицу. Чтобы извлечь содержимое компонента, нужно использовать синтаксис *object*[[*index*]]. Например, следующий код вернет матрицу:

```
> fit[[2]]
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1        5.006       3.428        1.462       0.246
2        5.902       2.748        4.394       1.434
3        6.850       3.074        5.742       2.071
```

Разница может иметь большое значение, в зависимости от того, что дальше предполагается делать с результатами. Если результаты должны передаваться функции, принимающей матрицу, то следует использовать нотацию с двойными квадратными скобками.

Чтобы получить содержимое одного именованного компонента, можно использовать нотацию `$`. В этом случае `object[["name"]]` и `object$name` эквивалентны:

```
> fit$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1        5.006       3.428        1.462       0.246
2        5.902       2.748        4.394       1.434
3        6.850       3.074        5.742       2.071
```

Этот пример также объясняет, почему нотация `$` работает с таблицами данных. Рассмотрим таблицу данных `iris`. Таблица данных – это частный случай списка, в котором каждая переменная представлена как компонент. Например, инструкция `iris$Sepal.Length` эквивалентна инструкции `iris[["Sepal.Length"]]` и возвращает вектор длин чашелистиков со 150 элементами.

Нотации можно комбинировать для получения элементов внутри компонентов. Например, вот как можно получить второй компонент из `fit` (матрицы средних) и вернуть первую строку (средние значения четырех переменных в первом кластере):

```
> fit[[2]][1, ]
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       5.006        3.428        1.462        0.246
```

Пробел после запятой в последней паре квадратных скобок означает, что требуется вернуть все четыре столбца.

После извлечения компонентов и элементов из списков, возвращаемых функциями, можете продолжить их обработку. Например, в листинге 20.1 показано, как построить линейный график центроидов кластеров.

Листинг 20.1. Построение линейного графика центроидов из кластерного анализа методом *k*-средних

```
> set.seed(1234)
> fit <- kmeans(iris[1:4], 3)                          (1)
> means <- fit$center                                  (1)
> means <- as.data.frame(means)                        (1)
> means$cluster <- row.names(means)                    (1)

> library(tidyr)                                       (2)
> plotdata <- gather(means,                            (2)
                     key="variable",                   (2)
                     value="value",                    (2)
                     -cluster)                         (2)
```

```
> names(plotdata) <- c("Cluster", "Measurement", "Centimeters")  (2)
> head(plotdata)

  Cluster  Measurement Centimeters
1       1 Sepal.Length       5.006
2       2 Sepal.Length       5.902
3       3 Sepal.Length       6.850
4       1  Sepal.Width       3.428
5       2  Sepal.Width       2.748
6       3  Sepal.Width       3.074

library(ggplot2)                                                    (3)
ggplot(data=plotdata,                                               (3)
       aes(x=Measurement, y=Centimeters, group=Cluster)) +           (3)
       geom_point(size=3, aes(shape=Cluster, color=Cluster)) +      (3)
       geom_line(size=1, aes(color=Cluster)) +                      (3)
       labs(title="Mean Profiles for Iris Clusters") +               (3)
       theme_minimal()                                              (3)
```

(1) Получение средних значений кластеров.

(2) Преобразование данных в длинный формат.

(3) Построение линейного графика.

Сначала извлекается матрица центроидов кластеров (строки – это кластеры, а столбцы – средние значения переменных) и преобразуется в таблицу данных. Номер кластера добавляется как дополнительная переменная (1). Затем таблица данных преобразуется в длинный формат с помощью пакета `tidyr` (раздел 5.5.2) (2). Наконец, с помощью пакета `ggplot2` строится диаграмма (3). На рис. 20.1 показан получившийся график.

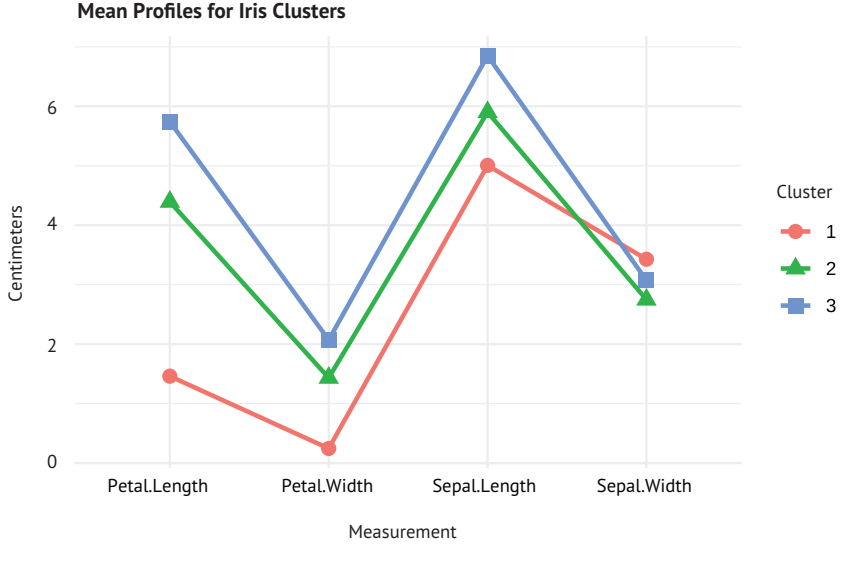


Рис. 20.1. График центроидов (средних) для трех кластеров

У нас получилось построить этот график, потому что все отображаемые переменные имеют одинаковые единицы измерения (сантиметры). Если в кластерном анализе использовались переменные в разных масштабах, то их следует стандартизировать перед построением графика и подписать ось *y* как, например, «стандартизированные оценки». Подробнее об этом рассказывается в разделе 16.1.

Теперь, зная, как представлять данные в структурах и извлекать их, перейдем к управлению потоком выполнения.

20.1.2. Структуры управления потоком выполнения

Когда интерпретатор R обрабатывает код, он читает его последовательно, строка за строкой. Если строка не является полным оператором, он читает следующие строки до тех пор, пока не будет построен правильно сформированный оператор. Например, сумму 3 + 2 + 5 можно получить так:

```
> 3 + 2 + 5
[1] 10
```

или так:

```
> 3 + 2 +
  5
[1] 10
```

Знак + в конце первой строки явно указывает, что инструкция продолжается на следующей строке. Но

```
> 3 + 2
[1] 5
> + 5
[1] 5
```

не сработает, потому что `3 + 2` интерпретируется как законченная инструкция.

Иногда требуется, чтобы код выполнялся непоследовательно, например только если соблюдается определенное условие или код должен повториться несколько раз. В этом разделе описываются три инструкции управления потоком выполнения, которые особенно полезны при написании функций: `for()`, `if()` и `ifelse()`.

Циклы for

Функция `for()` позволяет многократно выполнить некоторую инструкцию. Она имеет следующий синтаксис:

```
for(var in seq){
     statements
}
```

где `var` – имя переменной, а `seq` – выражение, возвращающее вектор. Если многократно выполняться должна только одна инструкция, то фигурные скобки можно опустить:

```
> for(i in 1:5) print(1:i)
[1] 1
[1] 1 2
[1] 1 2 3
[1] 1 2 3 4
[1] 1 2 3 4 5

> for(i in 5:1)print(1:i)
[1] 1 2 3 4 5
[1] 1 2 3 4
[1] 1 2 3
[1] 1 2
[1] 1
```

Обратите внимание, что переменная `var` продолжает существовать после выхода из функции `for()`. Здесь `i` равно `1`.

В предыдущих примерах параметр `seq` был числовым вектором. Но он также может быть текстовым вектором. Например, следующий код создаст четыре гистограммы:

```
> vars <- c("mpg", "disp", "hp", "wt")
> for(i in vars) hist(mtcars[[i]])
```

if() и else

Функция `if()` позволяет выполнять инструкции только при определенном условии. Она имеет следующий синтаксис:

```
if(condition){
    statements
} else {
    statements
}
```

Условие `condition` должно быть одноэлементным логическим вектором (содержащим `TRUE` или `FALSE`) и не может быть отсутствующим значением (`NA`). Часть `else` – необязательная. Если выполняется только одна инструкция, то фигурные скобки можно опустить.

Например, рассмотрим следующий фрагмент:

```
if(interactive()){
    plot(x, y)
} else {
    png("myplot.png")
    plot(x, y)
    dev.off()
}
```

Если код выполняется в интерактивном режиме, то функция `interactive()` вернет значение `TRUE`, и на экране появится график. В противном случае график будет сохранен на диск. Мы часто будем использовать функцию `if()` в главе 22.

ifelse()

Функция `ifelse()` – векторизованная версия функции `if()`. Векторизация позволяет функции обрабатывать объекты без явного применения конструкции цикла. Она имеет следующий синтаксис:

```
ifelse(test, yes, no)
```

где `test` – это объект, приведенный к логическому типу, `yes` возвращает значения для истинных элементов в `test`, а `no` – значения для ложных элементов в `test`.

Допустим, у нас есть вектор *p*-значений, полученный в ходе статистического анализа, возвращающего шесть статистических критериев, и требуется пометить эти значения как значимые на уровне $p < 0{,}05$. Это можно сделать так:

```
> pvalues <- c(.0867, .0018, .0054, .1572, .0183, .5386)
> results <- ifelse(pvalues <.05, "Significant", "Not Significant")
> results

[1] "Not Significant" "Significant"     "Significant"
[4] "Not Significant" "Significant"     "Not Significant"
```

Функция `ifelse()` выполнит обход значений в векторе `pvalues` и вернет текстовый вектор со значениями `"Significant"` (значимый) или `"Not Significant"` (незначимый) в зависимости от значимости соответствующего элемента `pvalues` на уровне 0,05.

Тот же результат можно получить с помощью явного цикла:

```
pvalues <- c(.0867, .0018, .0054, .1572, .0183, .5386)
results <- vector(mode="character", length=length(pvalues))
for(i in 1:length(pvalues)){
  if (pvalues[i] < .05) results[i] <- "Significant"
  else results[i] <- "Not Significant"
}
```

Векторизованная версия выполняется быстрее и эффективнее.

Существуют и другие управляющие структуры, в том числе `while()`, `repeat()` и `switch()`, но представленные здесь используются чаще всего. Теперь, познакомившись со структурами данных и управляющими структурами, можем поговорить о создании функций.

20.1.3. Создание функций

Почти все в R является функциями. Даже такие арифметические операторы, как `+`, `-`, `/` и `*`, на самом деле являются функциями. Например, `2 + 2` можно записать как `"+"(2, 2)`. В этом разделе описывается синтаксис функции, а в разделе 20.2 мы поговорим об областях видимости.

Синтаксис определения функций

Синтаксис определения функции:

```
functionname <- function(parameters){
                        statements
                        return(value)
}
```

Если функция принимает несколько параметров, они перечисляются через запятую.

Параметры можно передавать по именам, позициям или двумя способами одновременно. Кроме того, параметры могут иметь значения по умолчанию. Взгляните на следующую функцию:

```
f <- function(x, y, z=1){
    result <- x + (2*y) + (3*z)
    return(result)
}
```

```
> f(2,3,4)
[1] 20
> f(2,3)
[1] 11
> f(x=2, y=3)
[1] 11
> f(z=4, y=2, 3)
[1] 19
```

В первом случае параметры передаются по позициям (`x = 2`, `y = 3`, `z = 4`). Во втором случае параметры передаются по позициям, но `z` получает значение умолчанию, равное `1`. В третьем случае параметры передаются по именам, а `z` снова получает значение умолчанию, равное `1`. В последнем случае `y` и `z` передаются по именам и предполагается, что `x` является первым параметром, имя которого не указано явно (`x=3`). Этот пример также демонстрирует, что параметры, передаваемые по именам, могут следовать в любом порядке.

Параметры необязательны, но круглые скобки должны добавляться всегда, даже если в вызов функции не передается никаких значений. Функция `return()` возвращает объект, созданный текущей функцией. Вызов этой функции также необязателен, и если он отсутствует, то возвращается результат выполнения последней инструкции.

Для просмотра имен параметров функции можно использовать функцию `args()` для просмотра имен параметров и значений по умолчанию:

```
> args(f)
  function (x, y, z = 1)
  NULL
```

Функция `args()` предназначена для использования в интерактивном режиме. Если вам понадобится получить имена параметров и значения по умолчанию во время выполнения программы, то используйте функцию `formals()`, возвращающую список с необходимой информацией.

Параметры передаются по значению, а не по ссылке. Рассмотрим следующий вызов функции:

```
result <- lm(height ~ weight, data=women)
```

Доступ к набору данных `women` происходит не напрямую. Сначала создается копия, которая и передается в функцию. Если набор данных `women` очень большой, то оперативная память может быстро исчерпаться. Это может стать проблемой при работе с большими объемами данных, и может потребоваться использовать специальные методы (приложение F).

Область видимости объекта

Область видимости объектов в R (определяет, как имена разрешаются в содержимое) имеет сложную организацию. В типичном случае:

- объекты, созданные вне какой-либо функции, являются глобальными (и доступны внутри любой функции). Объекты, созданные внутри функции, являются локальными (доступны только внутри функции);
- локальные объекты исчезают в конце выполнения функции. Только объекты, переданные обратно через функцию `return()` (или присвоенные с помощью оператора `<<-`), доступны после выполнения функции;
- к глобальным объектам можно обращаться (читать) из функций, но их нельзя изменять (опять же, если не используется оператор `<<-`);
- объекты, переданные в функцию через параметры, не могут быть изменены функцией, потому что передаются копии объектов, а не сами объекты.

Вот простой пример:

```
> x <- 2
> y <- 3
> z <- 4
> f <- function(w){
         z <- 2
         x <- w*y*z
         return(x)
 }
> f(x)
[1] 12
```

```
> x
[1] 2
> y
[1] 3
> z
[1] 4
```

В этом примере функция `f()` получает копию `x`, поэтому оригинал не изменяется. Значение `y` получено из окружения. Несмотря на то что `z` существует в окружении, функция использует значение, установленное в ней, и не изменяет значение в окружении.

Чтобы лучше понять правила области видимости, нам нужно обсудить окружения.

20.2. Работа с окружениями

Окружение в R состоит из кадра и вмещающего окружения. *Кадр* – это набор пар символ–значение (имена объектов и их содержимое), а *вмещающее окружение* – это указатель на окружение. Вмещающее окружение также называют *родительским окружением*. R позволяет манипулировать окружениями, давая возможность точного управления областями видимости и разделения функций и данных.

В интерактивном сеансе, сразу после запуска, вы оказываетесь в глобальном окружении. При необходимости можно создать новое окружение с помощью функции `new.env()` и добавить в него значения с помощью функции `assign()`. Значение объекта можно получить из окружения с помощью функции `get()`. Окружения – это структуры данных, позволяющие управлять областью видимости объектов (их также можно рассматривать как хранилища объектов). Например:

```
> x <- 5
> myenv <- new.env()
> assign("x", "Homer", env=myenv)
> ls()
[1] "myenv" "x"
> ls(myenv)
[1] "x"
> x
[1] 5
> get("x", env=myenv)
[1] "Homer"
```

Объект с именем `x` существует в глобальном окружении и имеет значение `5`. Объект с таким же именем `x` существует в окружении `myenv` и имеет значение `"Homer"`. Использование отдельных окружений помогает предотвратить путаницу между объектами.

Помимо функций `assign()` и `get()`, можно использовать нотацию `$`. Например, следующий код даст тот же результат:

```
> myenv <- new.env()
> myenv$x <- "Homer"
> myenv$x
[1] "Homer"
```

Функция `parent.env()` отображает родительское окружение. Продолжая предыдущий пример, родительским окружением для `myenv` является глобальное окружение:

```
> parent.env(myenv)
<environment: R_GlobalEnv>
```

Для глобального окружения родительским является пустое окружение. Подробности ищите в `help(environment)`.

Поскольку функции являются объектами, они тоже имеют окружения. Это важно учитывать при использовании *замыканий функций* (функций, упакованных с состоянием, существовавшим на момент их создания). Рассмотрим функцию, созданную другой функцией:

```
trim <- function(p){
    trimit <- function(x){
      n <- length(x)
      lo <- floor(n*p) + 1
      hi <- n + 1 - lo
      x <- sort.int(x, partial = unique(c(lo, hi)))[lo:hi]
    }
    trimit
}
```

Функция `trim(p)` возвращает функцию, отсекающую `p` процентов наибольших и наименьших значений из вектора `x`:

```
> x <- 1:10
> trim10pct <- trim(.1)
> y <- trim10pct(x)
> y
[1] 2 3 4 5 6 7 8 9
> trim20pct <- trim(.2)
> y <- trim20pct(x)
> y
  [1] 3 4 5 6 7 8
```

Этот код работает просто потому, что *p* находится в окружении, доступном для функции `trimit()` и сохраняемом вместе с функцией:

```
> ls(environment(trim10pct))
[1] "p"      "trimit"
> get("p", env=environment(trim10pct))
[1] 0.1
```

Отсюда следует, что функции в R сохраняют объекты, существовавшие в их окружении в момент создания. Этот факт помогает объяснить следующее поведение:

```
> makeFunction <- function(k){
    f <- function(x){
      print(x + k)
    }
  }

> g <- makeFunction(10)
> g(4)
[1] 14
> k <- 2
> g(5)
[1] 15
```

Функция `g()` использует `k=10` независимо от того, какое значение имеет `k` в глобальном окружении, потому что объект `k` имел значение `10` при создании функции. В этом можно убедиться с помощью следующего кода:

```
> ls(environment(g))
[1] "f" "k"
> environment(g)$k
[1] 10
```

Как правило, значение объекта извлекается из локального окружения. Если искомый объект отсутствует в локальном окружении, R выполняет поиск в родительском окружении, затем в окружении, вмещающем родительское окружение, и так далее, пока объект не будет найден. Если R достигнет пустого окружения и так и не найдет искомый объект, то он сообщит об ошибке. Это называется *лексической областью видимости*.

Чтобы узнать больше об окружениях и лексической области видимости, обращайтесь к статье Кристофера Бэра (Christopher Bare) «Environments in R» (http://mng.bz/uPYM) и к статье Даррена Уилкинсона (Darren Wilkinson) «Lexical Scope and Function Closures in R» (http://mng.bz/R286).

20.3. Нестандартная оценка

R по умолчанию определяет, когда и как вычислять выражения. Рассмотрим следующий пример. Создадим функцию с именем `mystats`:

```
mystats <- function(data, x){
   x <- data[[x]]
   c(n=length(x), mean=mean(x), sd=sd(x))
  }
```

Затем вызовем ее, не заключив в кавычки значение для параметра `x`:

```
> mystats(mtcars, mpg)

 Error in (function(x, i, exact) if (is.matrix(i)) as.matrix(x)[[i]] else
.subset2(x,  : object 'mpg' not found
```

R пытается вычислить выражение (разрешить имя объекта или функции) в аргументе функции сразу же, непосредственно перед вызовом функции. В этом случае R сгенерировал сообщение об ошибке, потому что не смог найти имя `mpg` в рабочем пространстве (в данном случае – в глобальном окружении).

Но если заключить в кавычки значение для параметра `x`, то функция выполнится успешно. Строковые литералы передаются в вызов функции как есть (поиск не выполняется):

```
> mystats(mtcars, "mpg")
        n      mean        sd
32.000000 20.090625  6.026948
```

Это поведение по умолчанию называется *стандартной оценкой* (standard evaluation, SE).

Но как быть, если вам нравится первый способ вызова функции? В этом случае можно изменить определение функции:

```
mystats <- function(data, x){
   x <- as.character(substitute(x))
   x <- data[[x]]
   c(n=length(x), mean=mean(x), sd=sd(x))
  }
```

Функция `substitute()` передает имя объекта без его оценки. Затем имя преобразуется в строку с помощью `as.character()`. Теперь оба варианта вызова:

```
> mystats(mtcars, mpg)
        n      mean        sd
32.000000 20.090625  6.026948
```

и

```
> mystats(mtcars, "mpg")
        n      mean        sd
32.000000 20.090625  6.026948
```

– прекрасно работают! Это называется *нестандартной оценкой* (non-standard evaluation, NSE). Функция `library()` использует NSE, поэтому оба вызова, `library(ggplot2)` и `library("ggplot2")`, будут успешно выполнены.

Когда может понадобиться использовать нестандартную оценку вместо стандартной? Многое зависит от простоты использования. Следующие две инструкции эквивалентны:

```
df <- mtcars[mtcars$mpg < 20 & mtcars$carb==4,
             c("disp", "hp", "drat", "wt")]

df <- subset(mtcars, mpg < 20 & carb == 4, disp:wt)
```

В первой инструкции используется SE, а во второй – NSE. Вторая инструкция проще набирается и читается. Но чтобы облег-

чить жизнь пользователю, программист должен потрудиться над созданием функций.

В R имеется несколько функций, управляющих оценкой выражений. Пакеты из библиотеки `tidyverse`, такие как `dplyr`, `tidyr` и `ggplot2`, используют свою версию NSE, называемую *упорядоченной оценкой* (tidy evaluation). Если вы решите применять функции из `tidyverse` в своих функциях, то вам придется использовать инструменты, предлагаемые пакетом `rlang`.

В частности, вы должны будете использовать `enquo()` и `!!` для заключения аргументов в кавычки и изъятия их из кавычек. Например, следующий код:

```
myscatterplot <- function(data, x, y){
  require(ggplot2)
  x <- enquo(x)
  y <- enquo(y)
  ggplot(data, aes(!!x, !!y)) + geom_point() + geom_smooth()
}

myscatterplot(mtcars, wt, mpg)
```

прекрасно работает. Попробуйте!

Для манипулирования несколькими аргументами, передаваемыми вместо `...`, используйте `enquos()` и `!!!`. Три точки – это заполнитель для одного или нескольких аргументов (имен или выражений), передаваемых в функцию:

```
mycorr <- function(data, ..., digits=2){
  require(dplyr)
  vars <- enquos(...)
  data %>% select(!!!vars) %>% cor() %>% round(digits)
}

mycorr(mtcars, mpg, wt, disp, hp)
```

Последние версии `rlang` предоставляют сокращенные варианты, например предыдущие функции можно записать так:

```
myscatterplot <- function(data, x, y){
  require(ggplot2)
  ggplot(data, aes({{x}}, {{y}})) + geom_point() + geom_smooth()
}

mycorr <- function(data, ..., digits=2){
  require(dplyr)
  data %>% select(...) %>% cor() %>% round(digits)
}
```

Две пары фигурных скобок и нотация с тремя точками избавляют от необходимости явно использовать `enquo()` и `enquos()`, но они не поддерживаются в версиях `rlang` ниже 0.4.0.

R дает программисту значительный контроль над тем, как и когда оценивается код. Но большим возможностям сопутствует большая ответственность (и тупая резь в глазах). NSE – очень сложная тема, и в этом разделе были затронуты только аспекты, наиболее часто встречающиеся при написании собственных функций. Дополнительную информацию можно найти в статьях Броди Гаслама (Brodie Gaslam; http://www.brodieg.com/2020/05/05/on-nse/) и Хэдли Уикхема (Hadley Wickham; http://adv-r.hadley.nz/).

20.4. Объектно-ориентированное программирование

R – это язык *объектно-ориентированного программирования* (ООП), основанный на использовании обобщенных функций. Каждый объект имеет атрибут `class`, посредством которого определяется, какой код запускать, когда копия объекта передается обобщенной функции, такой как `print()`, `plot()` или `summary()`.

R поддерживает несколько моделей ООП, включая S3, S4, RC и R6. Модель S3 старше, проще и менее структурирована. Модель S4 более новая и хорошо структурированная. Обе поддерживаются базовой версией R. Модель S3 проще в использовании, и большинство приложений в R используют именно ее. В этом разделе мы сосредоточимся на модели S3 и закончим кратким обсуждением ее ограничений и попыток их устранить в модели S4. RC и R6 используются значительно реже и здесь не рассматриваются.

20.4.1. Обобщенные функции

Используя информацию о классе объекта, R определяет, какое действие следует предпринять при вызове обобщенной функции. Рассмотрим следующий код:

```
summary(women)
fit <- lm(weight ~ height, data=women)
summary(fit)
```

В первом случае функция `summary()` создает описательную статистику для каждой переменной в таблице данных `women`. Во втором случае `summary()` создает описание модели линейной регрессии. Как это возможно?

Давайте посмотрим на код функции `summary()`:

```
> summary
function (object, ...) UseMethod("summary")
```

А теперь на класс таблицы данных `women` и объекта `fit`:

```
> class(women)
[1] "data.frame"
> class(fit)
[1] "lm"
```

Вызов `summary(women)` выполняет функцию `summary.data.frame(women)`, если она существует, или `summary.default(women)` в противном случае. Аналогично `summary(fit)` выполняет функцию `summary.lm(fit)`, если она существует, или `summary.default(fit)` в противном случае. Функция `UseMethod()` передает управление функции, соответствующей классу объекта.

Получить список всех доступных обобщенных функций в S3 можно с помощью `method()`:

```
> methods(summary)
 [1] summary.aov             summary.aovlist
 [3] summary.aspell*         summary.connection
 [5] summary.data.frame      summary.Date
 [7] summary.default         summary.ecdf*
             ...часть вывода опущена...
[31] summary.table           summary.tukeysmooth*
[33] summary.wmc*

   Non-visible functions are asterisked
```

Количество возвращаемых функций зависит от набора установленных пакетов. На моем компьютере отдельные функции `summary()` определены для 33 классов!

Просмотреть код любой функции из предыдущего примера можно, если ввести ее имя без круглых скобок (`summary.data.frame`, `summary.lm` и `summary.default`). Скрытые функции, за именами которых следуют звездочки, нельзя просмотреть таким способом. В подобных случаях можно использовать функцию `getAnywhere()`. Чтобы увидеть код `summary.ecdf()`, введите `getAnywhere(summary.ecdf)`. Просмотр существующего кода – отличный способ получить идею, как можно реализовать свои функции.

Мы уже видели такие классы, как `numeric`, `matrix`, `data.frame`, `array`, `lm`, `glm` и `table`, но вообще классом объекта может быть любая строка. Кроме того, круг обобщенных функций не ограничивается функциями `print()`, `plot()` и `summary()`. Любая функция может быть обобщенной. Например, в листинге 20.2 определяется обобщенная функция с именем `mymethod()`.

Листинг 20.2. Пример произвольной обобщенной функции

```
> mymethod <- function(x, ...) UseMethod("mymethod")          (1)
> mymethod.a <- function(x) print("Using A")                  (1)
> mymethod.b <- function(x) print("Using B")                  (1)
> mymethod.default <- function(x) print("Using Default")      (1)
```

```
> x <- 1:5
> y <- 6:10
> z <- 10:15
> class(x) <- "a"                                    (2)
> class(y) <- "b"                                    (2)

> mymethod(x)                                        (3)
[1] "Using A"                                        (3)
> mymethod(y)                                        (3)
[1] "Using B"                                        (3)
> mymethod(z)                                        (3)
[1] "Using Default"                                  (3)

> class(z) <- c("a", "b")                            (4)
> mymethod(z)                                        (4)
[1] "Using A"                                        (4)

> class(z) <- c("c", "a", "b")                       (5)
> mymethod(z)                                        (5)
[1] "Using A"                                        (5)
```

(1) Определение обобщенной функции.

(2) Назначение классов объектам.

(3) Применение обобщенной функции к объектам.

(4) Применение обобщенной функции к объекту с двумя классами.

(5) Обобщенная функция не имеет варианта по умолчанию для класса `"c"`.

В этом примере обобщенные функции `mymethod()` определены для классов `a` и `b`. Также определена функция `default()` (1). Далее в листинге определяются объекты `x`, `y` и `z`, после чего объектам `x` и `y` присваивается класс (2). Затем к каждому объекту применяется `mymethod()`, и вызывается соответствующая функция (3). Для объекта `z` вызывается метод по умолчанию, потому что этот объект имеет класс `integer`, а функция `mymethod.integer()` не определена.

Объекту можно назначить несколько классов (например, здание может быть одновременно и жилым, и коммерческим). Как в таком случае R определяет, какую функцию вызвать?

Когда объекту `z` назначается два класса (4), то вызывается метод, соответствующий первому классу. В последнем примере (5) нет функции `mymethod.c()`, поэтому для определения метода используется следующий в списке (`a`). R просматривает список классов слева направо и останавливается на первом, для которого имеется специализированная версия обобщенной функции.

20.4.2. Ограничения модели S3

Основное ограничение объектной модели S3 – отсутствие проверки целостности, т. е. любому объекту может быть назначен любой класс.

В следующем примере таблице данных `women` присвоен класс `lm`, что бессмысленно и приводит к ошибкам:

```
> class(women) <- "lm"
> summary(women)
Error in if (p == 0) { : argument is of length zero
```

Модель S4 является более формальной и строгой и устраняет некоторые сложности, связанные с менее структурированным подходом в модели S3. В модели S4 классы определяются как абстрактные объекты со слотами, содержащими определенные типы информации (то есть типизированные переменные). Конструкции объектов и методов формально определены с соблюдением правил. Но программировать с использованием модели S4 намного сложнее. Чтобы узнать больше о модели S4, обращайтесь к статье Кристофа Дженолини (Chistophe Genolini) «A (Not So) Short Introduction to S4» (http://mng.bz/1VkD).

20.5. Разработка эффективного кода

Среди программистов есть поговорка: «Опытный программист – это тот, кто тратит час на оптимизацию своего кода, чтобы он работал на секунду быстрее». R – язык спрайтов, и большинству пользователей R не нужно беспокоиться об эффективности кода. Самый простой способ ускорить ваш код – увеличить вычислительную мощность аппаратного обеспечения (добавить ОЗУ, поменять процессор на более производительный и т. д.). Как правило, важнее писать простой и понятный код, чем оптимизированный для достижения большей скорости выполнения. Но при работе с большими наборами данных или многократно повторяющимися задачами скорость может стать проблемой.

Вот несколько советов, которые помогут вам сделать ваши программы более эффективными:

- читайте только те данные, которые нужны;
- по возможности используйте векторизацию вместо циклов;
- создавайте объекты сразу правильного размера, чтобы программе не пришлось многократно изменять их размер;
- используйте распараллеливание для повторяющихся, независимых задач.

Рассмотрим каждый по очереди.

20.5.1. Эффективный ввод данных

Импортируя данные, читайте только действительно необходимые данные и передавайте функции импорта как можно больше информации. Кроме того, по возможности выбирайте оптимизированную функцию.

Предположим, вам нужно получить из текстового файла в формате CSV три числовые переменные (возраст, рост, вес), две текстовые переменные (раса, пол) и одну переменную с датой (дата рождения). Для этого можно использовать функции `read.csv` (из стандартной библиотеки R), `fread` (из пакета `data.table`) и `read_csv` (из пакета `readr`), но две последние работают значительно быстрее с большими файлами.

Кроме того, код можно ускорить, выбрав только необходимые переменные и указав их типы (чтобы функции не приходилось тратить время на определение типов). Например, такой код:

```
library(readr)
mydata <- read_csv(mytextfile,
                col_types=cols_only(age="i", height="d", weight="d",
                                    race="c", sex="c", dob="D"))
```

– будет работать быстрее, чем:

```
mydata <- read_csv(mytextfile)
```

Здесь `i=integer`, `d=double`, `c=character` и `D=Date`. Полный список типов переменных смотрите в справке `?read_csv`.

20.5.2. Векторизация

По возможности используйте векторизацию вместо циклов. Под *векторизацией* подразумевается использование функций R, предназначенных для оптимизированной обработки векторов. В стандартной библиотеке R к таковым относятся: `ifelse()`, `colSums()`, `colMeans()`, `rowSums()` и `rowMeans()`. Пакет `matrixStats` предлагает оптимизированные функции для множества вычислений, включая подсчет количества, суммы, произведения, меры центральной тенденции и дисперсии, квантилей, рангов и группировки. Такие пакеты, как `dplyr` и `data.table`, тоже имеют в своем составе высокооптимизированные функции.

Рассмотрим матрицу с 1 млн строк и 10 столбцами. Подсчитаем суммы по столбцам с помощью циклов и затем с помощью функции `colSums()`. Сначала создадим матрицу:

```
set.seed(1234)
mymatrix <- matrix(rnorm(10000000), ncol=10)
```

Затем напишем функцию `accum()`, использующую цикл `for` для вычисления сумм по столбцам:

```
accum <- function(x){
   sums <- numeric(ncol(x))
   for (i in 1:ncol(x)){
        for(j in 1:nrow(x)){
            sums[i] <- sums[i] + x[j,i]
        }
   }
}
```

Для измерения времени, затраченного на выполнение кода, можно использовать функцию `system.time()`:

```
> system.time(accum(mymatrix))
   user  system elapsed
  25.67    0.01   25.75
```

Теперь подсчитаем те же суммы с помощью функции `colSums()`:

```
> system.time(colSums(mymatrix))
     user  system elapsed
   0.02    0.00    0.02
```

На моей машине векторизованной функции потребовалось в 1200 раз меньше времени. На вашей машине результат может отличаться.

20.5.3. Правильный размер объектов

Получить дополнительное ускорение можно, если сразу инициализировать объекты окончательного размера и тут же заполнить их значениями, а не начинать с меньшего объекта и постепенно увеличивать его, добавляя значения. Допустим, у нас есть вектор `x` со 100 000 числовых значений и требуется получить вектор `y` с квадратами этих значений:

```
> set.seed(1234)
> k <- 100000
> x <- rnorm(k)
```

Вот одно из возможных решений:

```
>  y <- 0
>  system.time(for (i in 1:length(x)) y[i] <- x[i]^2)
   user  system elapsed
  10.03    0.00   10.03
```

Здесь сначала создается вектор `y` с одним элементом, размер которого последовательно увеличивается до 100 000. Моей машине потребовалось около 10 секунд, чтобы выполнить этот код.

Если сначала создать вектор `y` со 100 000 элементов:

```
>  y <- numeric(length=k)
>  system.time(for (i in 1:k) y[i] <- x[i]^2)
   user  system elapsed
   0.23    0.00    0.24
```

то те же вычисления займут намного меньше времени, потому что в процессе вычислений не придется тратить значительный объем времени на постоянное изменение размера объекта.

А если вдобавок использовать векторизацию:

```
> y <- numeric(length=k)
> system.time(y <- x^2)
   user  system elapsed
      0       0       0
```

то выигрыш получится еще больше. Обратите внимание, что для таких операций, как возведение в степень, сложение, умножение и т. д., существуют векторизованные функции.

20.5.4. Распараллеливание

Распараллеливание предполагает разбиение задачи на части и одновременное выполнение этих частей на двух или более ядрах процессора с последующим объединением результатов. Выполнение может происходить на ядрах одного и того же процессора или разных процессоров и даже разных машин в кластере. Задачи, требующие многократного выполнения вычислительных функций, скорее всего, выиграют от распараллеливания. К таким функциям относятся практически все методы Монте-Карло.

Многие пакеты для R поддерживают распараллеливание (см. статью Дирка Эддельбюттеля (Dirk Eddelbuettel) «High Performance and Parallel Computing with R», http://mng.bz/65sT). В этом разделе мы используем пакеты `foreach` и `doParallel`, чтобы увидеть эффект распараллеливания на одном компьютере. Пакет `foreach` поддерживает конструкцию цикла `foreach` (перебирает элементы коллекции) и упрощает распараллеливание операций в циклах. Пакет `doParallel` обеспечивает поддержку параллельного выполнения для пакета `foreach`.

Важнейшим шагом в методе главных компонент и факторном анализе является определение соответствующего количества компонентов или факторов (раздел 14.2.1). Один из подходов заключается в многократном вычислении собственных значений и собственных векторов корреляционной матрицы, полученной из случайных данных, которые имеют такое же количество строк и столбцов, что и исходные данные. В листинге 20.3 показан пример, сравнивающий время выполнения параллельной и непараллельной версий. Чтобы выполнить этот код, нужно установить оба пакета.

Листинг 20.3. Распараллеливание с foreach и doParallel

```
> library(foreach)                                          (1)
> library(doParallel)                                       (1)
> registerDoParallel(cores=detectCores())                   (1)

> eig <- function(n, p){                                    (2)
            x <- matrix(rnorm(100000), ncol=100)            (2)
            r <- cor(x)                                     (2)
            eigen(r)$values                                 (2)
  }                                                         (2)

> n <- 1000000
> p <- 100
> k <- 500
```

```
> system.time(                                                    3
    x <- foreach(i=1:k, .combine=rbind) %do% eig(n, p)           3
  )                                                              3

   user  system elapsed
  10.97    0.14   11.11
> system.time(                                                    4
    x <- foreach(i=1:k, .combine=rbind) %dopar% eig(n, p)        4
  )                                                              4

  user  system elapsed
  0.22    0.05    4.24
```

1 Загрузка пакетов и назначение количества ядер для параллельного выполнения.

2 Определение функции.

3 Обычное непараллельное выполнение.

4 Параллельное выполнение.

Этот код сначала загружает пакеты и задает количество ядер (на моей машине их четыре), которые будут использоваться для параллельных вычислений 1. Затем определяется функция для анализа собственных значений и собственных векторов 2. Здесь анализируются матрицы случайных данных размером 100 000×100. Функция `eig()` выполняется 500 раз с использованием `foreach` и `%do%` 3. Оператор `%do%` запускает функцию последовательно, а параметр `.combine=rbind` добавляет результаты в объект `x` в виде строк. В заключение та же функция запускается параллельно с использованием оператора `%dopar%` 4. В этом случае на параллельное выполнение ушло примерно в 2,5 раза меньше времени, чем на последовательное.

В этом примере каждая итерация функции `eig()` требует значительных вычислительных ресурсов и не требует дискового ввода-вывода. Именно в такой ситуации распараллеливание приносит наибольшую пользу. Недостатком распараллеливания является меньшая переносимость кода – нет никакой гарантии, что другие пользователи будут иметь такую же аппаратную конфигурацию.

Четыре показателя эффективности, описанные в этом разделе, могут помочь в решении повседневных задач программирования. Но они помогают только при обработке очень больших наборов данных (например, в диапазоне терабайт). Однако при работе с большими наборами данных рекомендуется использовать методы, подобные описанным в приложении F.

Обнаружение узких мест

«Почему мой код выполняется так долго?» R предоставляет инструменты для профилирования программ и выявления наиболее затратных функций. Добавьте вызовы `Rprof()` и `Rprof(NULL)` перед и после про-

филируемого кода. Затем выполните `summaryRprof()`, чтобы получить сводку о времени, затраченном на выполнение каждой функции. Подробности ищите в справке `?Rprof` и `?summaryRprof`.

Rstudio тоже поддерживает профилирование кода. Выберите пункт Start Profiling (Начать профилирование) в меню Profile (Профилирование), запустите профилируемый код и по завершении щелкните на красной кнопке Stop Profiling (Остановить профилирование). Результаты будут представлены в виде таблицы и графика.

Высокая эффективность – слабое утешение, когда в программе возникают сбои или она возвращает бессмысленные результаты. Поэтому далее мы рассмотрим методы выявления ошибок.

20.6. Отладка

Отладка – это процесс поиска и устранения ошибок или дефектов в программе. Было бы замечательно, если бы программы сразу начинали работать безупречно. А еще было бы здорово, если бы единороги жили по соседству со мной. К сожалению, почти во всех программах, кроме самых простых, возникают ошибки. Выявление причин этих ошибок и их исправление требуют времени. В этом разделе мы рассмотрим распространенные источники ошибок и инструменты, которые могут помочь их обнаруживать.

20.6.1. Распространенные источники ошибок

Ниже перечислены некоторые распространенные причины сбоев функций в R:

- имя объекта записано с ошибкой или объект не существует;
- один или несколько параметров в вызове функции указаны неверно. Это может произойти, если имена параметров написаны с ошибками, пропущены обязательные параметры, значения параметров имеют неправильный тип (например, вектор, когда требуется список) или значения параметров введены в неправильном порядке (когда имена параметров опущены);
- содержимое объекта не соответствует ожиданиям пользователя. В частности, ошибки часто вызываются передачей объектов со значениями `NULL`, `NaN` или `NA` в функцию, которая не может их обработать.

Третья причина встречается чаще, чем вы думаете. Это результат подхода к ошибкам и предупреждениям, реализованного в R.

Рассмотрим следующий пример. Допустим, что мы решили придать переменной `am` (тип коробки передач) в наборе данных `mtcars` более информативное название, а затем сравнить расход бензина автомобилей с автоматической и механической коробками передач:

```
> mtcars$Transmission <- factor(mtcars$a,
                                levels=c(1,2),
                                labels=c("Automatic", "Manual"))
> aov(mpg ~ Transmission, data=mtcars)
Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
  contrasts can be applied only to factors with 2 or more levels
```

Черт! (Мне неудобно признаваться в этом, но я действительно так и сказал.) Что случилось?

Мы не получили сообщение об ошибке «Объект xxx не найден», поэтому, скорее всего, ошибка не в написании имени функции, таблицы данных или переменной. Давайте посмотрим на данные, которые были переданы в функцию `aov()`:

```
> head(mtcars[c("mpg", "Transmission")])
                   mpg Transmission
Mazda RX4         21.0    Automatic
Mazda RX4 Wag     21.0    Automatic
Datsun 710        22.8    Automatic
Hornet 4 Drive    21.4         <NA>
Hornet Sportabout 18.7         <NA>
Valiant           18.1         <NA>

> table(mtcars$Transmission)

Automatic    Manual
       13         0
```

Здесь не нашлось машин с механической коробкой передач. В исходном наборе данных переменная `am` кодируется следующим образом: 0 = автоматическая, 1 = механическая (но не 1 = автоматическая, 2 = механическая).

Функция `factor()` с радостью сделала то, что мы просили, не предупредив об ошибке. Всем автомобилям с механической коробкой передач она установила признак автоматической трансмиссии, а всем автомобилям с автоматической коробкой передач – признак отсутствующей трансмиссии. Поскольку доступной осталась только одна группа, дисперсионный анализ не удался. Проверка правильности входных данных в каждой функции может сэкономить вам часы утомительной работы.

20.6.2. Инструменты отладки

Изучение имен объектов, параметров функций и входных данных функций помогает обнаружить множество источников ошибок, но иногда бывает нужно вникнуть во внутреннюю работу функций и функций, вызывающих другие функции. В таких случаях может пригодиться встроенный в R отладчик. В табл. 20.1 перечислены некоторые полезные функции отладки.

Таблица 20.1. Встроенные функции отладки

Функция	Описание
`debug()`	Помечает функцию для отладки
`undebug()`	Снимает метку с отлаживаемой функции
`browser()`	Позволяет перейти к пошаговому выполнению функции. Во время отладки ввод `n` или нажатие клавиши Enter выполняет текущий оператор и переходит к следующему. Ввод `c` продолжает выполнение функции до конца, отключая пошаговый режим. При вводе `where` отображается стек вызовов, а ввод `Q` останавливает выполнение и производит переход на верхний уровень. Другие команды, такие как `ls()`, `print()` и операторы присваивания, тоже можно вводить в строке приглашения отладчика
`trace()`	Модифицирует функцию, вставляя временный отладочный код для трассировки
`untrace()`	Отменяет трассировку и удаляет временный код
`traceback()`	Выводит последовательность вызовов функций, приведшую к последней необработанной ошибке

Функция `debug()` помечает функцию как отлаживаемую. При выполнении такой функции будет вызвана функция `browser()`, дающая возможность выполнить функцию по шагам. Функция `undebug()` отключает режим отладки, позволяя функции выполняться как обычно. Функция `trace()` позволяет вставить временный отладочный код в отлаживаемую функцию. Это особенно полезно при отладке базовых функций и функций из CRAN, недоступных для редактирования.

Если функция вызывает другие функции, то может быть трудно определить, где произошла ошибка. В этом случае если выполнить функцию `traceback()` сразу после ошибки, то она покажет последовательность вызовов функций, приведшую к этой ошибке. Последний вызов – это вызов, сгенерировавший ошибку.

Давайте рассмотрим пример. Функция `mad()` вычисляет среднее абсолютное отклонение для числового вектора. Используем функцию `debug()`, чтобы изучить работу функции `mad()`. В листинге 20.4 показан сеанс отладки.

Листинг 20.4. Пример сеанса отладки

```
> args(mad)                                                    (1)
function (x, center = median(x), constant = 1.4826,
    na.rm = FALSE, low = FALSE, high = FALSE)
NULL

> debug(mad)                                                   (2)
> mad(1:10)

debugging in: mad(x)
debug: {
    if (na.rm)
        x <- x[!is.na(x)]
```

```
    n <- length(x)
    constant * if ((low || high) && n%%2 == 0) {
        if (low && high)
            stop("'low' and 'high' cannot be both TRUE")
        n2 <- n%/%2 + as.integer(high)
        sort(abs(x - center), partial = n2)[n2]
    }
    else median(abs(x - center))
}

Browse[2]> ls()                                          ❸
[1] "center"   "constant" "high"     "low"      "na.rm"    "x"

Browse[2]> center
[1] 5.5

Browse[2]> constant
[1] 1.4826

Browse[2]> na.rm
[1] FALSE

Browse[2]> x
 [1]  1  2  3  4  5  6  7  8  9 10

Browse[2]> n                                             ❹
debug: if (na.rm) x <- x[!is.na(x)]

Browse[2]> n
debug: n <- length(x)

Browse[2]> n
debug: constant * if ((low || high) && n%%2 == 0) {
    if (low && high)
        stop("'low' and 'high' cannot be both TRUE")
    n2 <- n%/%2 + as.integer(high)
    sort(abs(x - center), partial = n2)[n2]
} else median(abs(x - center))

Browse[2]> print(n)
[1] 10

Browse[2]> where
where 1: mad(x)

Browse[2]> c                                             ❺
exiting from: mad(x)
[1] 3.7065

> undebug(mad)
```

❶ **Просмотр формальных аргументов.**

❷ **Настройка функции для отладки.**

3 Вывод списка объектов.

4 Пошаговое выполнение кода.

5 Продолжение выполнения до конца.

Сначала вызывается функция `arg()`, чтобы вывести имена аргументов и значения по умолчанию для функции `mad()` 1. Затем вызовом `debug(mad)` функция помечается как отлаживаемая 2. Теперь всякий вызов `mad()` будет запускать функцию `browser()`, что позволит выполнить функцию `mad()` в пошаговом режиме.

При вызове `mad()` сеанс переходит в режим `browser()` (пошагового выполнения). Код функции показывается, но не выполняется. Кроме того, меняется приглашение к вводу на `Browse[n]>`, где `n` указывает *уровень вложенности отладки*. Это число увеличивается с каждым рекурсивным вызовом.

В режиме `browser()` могут выполняться другие команды R. Например, вызовом `ls()` можно получить список объектов, существующих в данный момент 3. При вводе имени объекта отображается его содержимое. Если объект имеет имя `n`, `c` или `Q`, то для просмотра его содержимого следует использовать `print(n)`, `print(c)` или `print(Q)`. Содержимое объектов можно изменять с помощью оператора присваивания.

Каждый шаг выполняется вводом буквы `n` или нажатием клавиши Enter 4. Оператор `where` сообщает, где вы находитесь в стеке вызовов функций. В случае с одной функцией это не особенно интересно, но при отладке групп функций, вызывающих друг друга, это может быть полезно.

Ввод `c` завершает пошаговый режим и выполняет оставшуюся часть текущей функции до конца 5. Нажатие `Q` завершает работу функции и возвращает вас в окружение R. Функция `debug()` полезна, когда есть циклы и требуется увидеть, как меняются значения. При желании вы можете сами вставить вызов `browser()` в свой код, чтобы облегчить себе поиск проблемы. Допустим, у вас есть переменная `X`, которая никогда не должна принимать отрицательного значения. Добавив код

```
if (X < 0) browser()
```

вы сможете изучить текущее состояние функции, если возникнет проблема. Когда функция будет достаточно отлажена, этот дополнительный код можно убрать. (Первоначально я написал «полностью отлажена», но это почти недостижимое состояние, поэтому я написал «достаточно отлажена», чтобы отразить реальность с точки зрения программиста.)

20.6.3. Параметры сеанса для поддержки отладки

Если у вас есть функции, вызывающие другие ваши функции, то при отладке могут помочь два параметра сеанса. Обычно, когда в программе встречается ошибка, R выводит сообщение и выходит из

функции. Параметр `options(error=traceback)` выводит стек вызовов (последовательность вызовов функций, которые привели к ошибке). Это может помочь определить, какая функция привела к ошибке.

Параметр `options(error=recover)` тоже выводит стек вызовов в случае ошибки, но при этом предлагает выбрать одну из функций в списке, а затем вызывает `browser()` в соответствующем окружении. Если в этой ситуации ввести `c`, то интерпретатор вновь вернет список функций, а если ввести `0`, то произойдет выход в строку приглашения к вводу R.

Режим `recover()` позволяет исследовать содержимое любого объекта в любой функции, выбранной из последовательности вызываемых функций. Выборочно просматривая содержимое объектов, часто можно определить источник проблемы. Чтобы вернуть интерпретатор R в состояние по умолчанию, установите параметр `options(error=NULL)`. В листинге 20.5 показан пример сеанса отладки.

Листинг 20.5. Пример сеанса отладки с функцией recover()

```
f <- function(x, y){                                    (1)
       z <- x + y                                       (1)
       g(z)                                             (1)
}                                                       (1)
g <- function(x){                                       (1)
       z <- round(x)                                    (1)
       h(z)                                             (1)
}                                                       (1)
h <- function(x){                                       (1)
       set.seed(1234)                                   (1)
       z <- rnorm(x)                                    (1)
       print(z)                                         (1)
}                                                       (1)
> options(error=recover)
> f(2,3)
[1] -1.207  0.277  1.084 -2.346  0.429
> f(2, -3)                                              (2)
Error in rnorm(x) : invalid arguments

Enter a frame number, or 0 to exit

1: f(2, -3)
2: #3: g(z)
3: #3: h(z)
4: #3: rnorm(x)

Selection: 4                                            (3)
Called from: rnorm(x)

Browse[1]> ls()
[1] "mean" "n"    "sd"
```

```
Browse[1]> mean
[1] 0

Browse[1]> print(n)
[1] -1

Browse[1]> c

Enter a frame number, or 0 to exit
1: f(2, -3)
2: #3: g(z)
3: #3: h(z)
4: #3: rnorm(x)

Selection: 3                                    4
Called from: h(z)

Browse[1]> ls()
[1] "x"

Browse[1]> x
[1] -1
Browse[1]> c

Enter a frame number, or 0 to exit

1: f(2, -3)
2: #3: g(z)
3: #3: h(z)
4: #3: rnorm(x)

Selection: 2                                    5
Called from: g(z)

Browse[1]> ls()
[1] "x" "z"

Browse[1]> x
[1] -1

Browse[1]> z
[1] -1

Browse[1]> c

Enter a frame number, or 0 to exit

1: f(2, -3)
2: #3: g(z)
3: #3: h(z)
4: #3: rnorm(x)
```

```
Selection: 1                                                    6
Called from: f(2, -3)

Browse[1]> ls()
[1] "x" "y" "z"

Browse[1]> x
[1] 2

Browse[1]> y
[1] -3

Browse[1]> z
[1] -1

Browse[1]> print(f)
function(x, y){
     z <- x + y
     g(z)
}
Browse[1]> c

Enter a frame number, or 0 to exit

1: f(2, -3)
2: #3: g(z)
3: #3: h(z)
4: #3: rnorm(x)

Selection: 0

> options(error=NULL)
```

1. **Создание функций.**
2. **Ввод значений, вызывающих ошибку.**
3. **Исследование** `rnorm()`.
4. **Исследование** `h(z)`.
5. **Исследование** `g(z)`.
6. **Исследование** `f(2, -3)`.

Здесь сначала определяется несколько функций. Функция `f()` вызывает функцию `g()`. Функция `g()` вызывает функцию `h()`.

Вызов `f(2, 3)` работает нормально, но вызов `f(2, -3)` завершается ошибкой. Благодаря установленному параметру `options(error=recover)` интерактивный сеанс немедленно переходит в режим восстановления. Выводится стек вызовов функций, и дается возможность выбрать функцию для исследования в режиме `browser()`.

Ввод `4` переносит интерпретатор в окружение функции `rnorm()`, где с помощью `ls()` выводится список объектов; здесь можно видеть, что `n = -1`, т. е. имеет недопустимое для `rnorm()` значение. Это

явно проблема, но, чтобы увидеть, как значение `-1` попало в `n`, мы перемещаемся вверх по стеку.

Ввод `c` возвращает нас в меню, а ввод `3` переносит вас в окружение функции `h(z)`, где `x = -1`. Ввод `c` и `2` переносит нас в окружение функции `g(z)`. Здесь значения `x` и `z` равны `-1`.

Наконец, переход в окружение функции `f(2, -3)` показывает, что `z` получает значение `-1`, потому что `x = 2` и `y = -3`.

Обратите внимание, что для просмотра кода функции используется вызов `print()`. Это полезно при отладке кода, написанного не вами. Обычно, чтобы просмотреть код функции, достаточно ввести ее имя без круглых скобок. Но в этом примере имя функции `f` совпадает с зарезервированной командой в режиме `browser()`, означающей «завершить выполнение текущего цикла или функции»; функция `print()` помогает избежать выполнения этой команды и получить код функции `f`.

Наконец, `c` возвращает нас в меню, а ввод `0` – в обычное приглашение к вводу R. Также в приглашение к вводу R можно вернуться вводом команды `Q`. Чтобы узнать больше об отладке в целом и о режиме восстановления в частности, я рекомендую прочитать превосходное руководство Роджера Пенга (Roger Peng) «An Introduction to the Interactive Debugging Tools in R» (http://mng.bz/GPR6).

20.6.4. Визуальный отладчик RStudio

В дополнение к инструментам, описанным выше, RStudio предлагает визуальную поддержку отладки.

Для начала давайте попробуем отладим функцию `mad()` из стандартной библиотеки R (рис. 20.2).

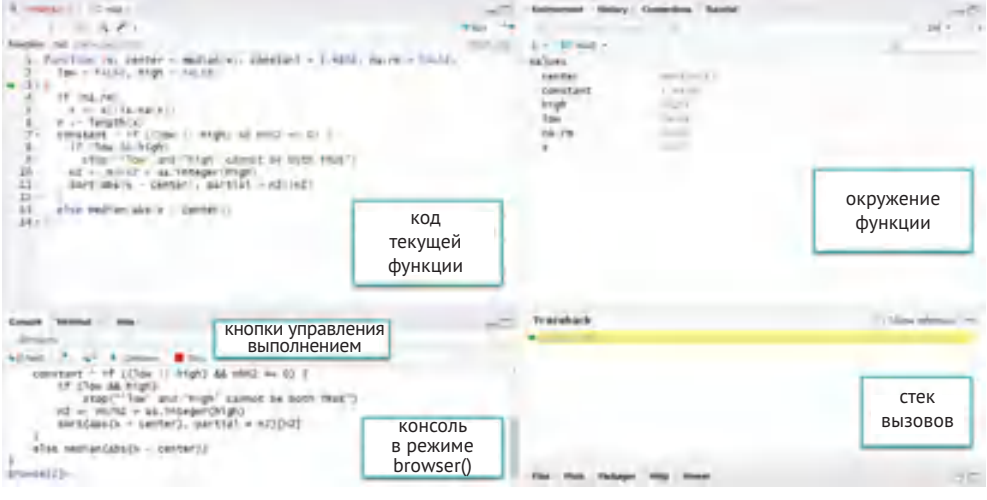


Рис. 20.2. Визуальная отладка функции mad() в RStudio

В левой верхней панели отображается код текущей функции. Зеленая стрелка указывает на текущую строку. В правой верхней панели перечислены объекты, имеющиеся в окружении функции, и их значения в текущей строке. По мере выполнения каждой строки функции эти значения будут меняться. В нижней правой панели показан стек вызовов. Поскольку функция всего одна, то в нем нет ничего интересного. Наконец, в нижней левой панели отображается консоль в режиме `browser()`. Здесь вы будете проводить большую часть своего времени.

В консоли можно вводить любые команды, упоминавшиеся в предыдущих разделах. Кроме того, можно пользоваться кнопками управления выполнения в верхней левой панели (рис. 20.2). Вот их назначение (слева направо):

- выполнить следующую строку кода;
- выполнить вход в функцию, вызываемую в текущей строке;
- выполнить оставшуюся часть текущей функции или цикла;
- продолжить выполнение до следующей точки останова (здесь не рассматривается);
- выйти из режима отладки.

Таким образом, можно по шагам выполнить свою функцию и понаблюдать, что происходит после выполнения каждой строки кода.

Далее давайте отладим код в листинге 20.5.

```
f <- function(x, y){
       z <- x + y
       g(z)
}
g <- function(x){
       z <- round(x)
       h(z)
}
h <- function(x){
       set.seed(1234)
       z <- rnorm(x)
       print(z)
}
> options(error=recover)
> f(2,-3)
```

Интерфейс отладчика изменится и будет выглядеть, как показано на рис. 20.3.

В верхней левой панели отобразится код функции, вызвавшей ошибку. Поскольку в этом примере у нас есть функции, которые вызывают другие функции, стек вызовов представляет определенный интерес. Щелкая на элементах списка в окне со стеком вызовов функций (внизу справа), можно просмотреть код каждой функции, поря-

док их вызова и значения, которые они передают. Когда функция выделена в списке, в правой верхней панели отобразятся значения объектов в ее окружении. Таким образом можно наблюдать за происходящим при вызове функций и видеть передаваемые значения.

Чтобы узнать больше о визуальном отладчике RStudio, я рекомендую прочитать статью «Debugging with RStudio IDE» (http://mng.bz/4M65) и посмотреть видеоурок «Introduction to debugging in R» (http://mng.bz/Q2R1).

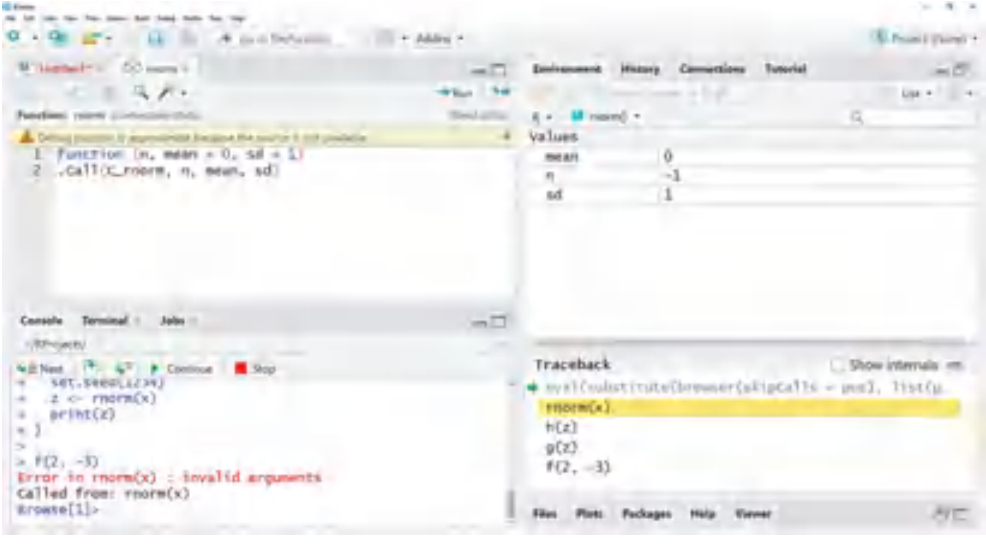


Рис. 20.3. Просмотр стека вызовов в RStudio. Функция f(2,–3) вызывает g(z), которая вызывает h(z), которая вызывает rnorm(x), и именно здесь возникает ошибка. Вы можете просмотреть каждую функцию и ее значения, выбрав соответствующую строку в панели Traceback (Трассировка)

20.7. Дополнительная информация

Существует несколько отличных источников информации о программировании на R. Для начала можно воспользоваться описанием самого языка R (http://mng.bz/U4Cm). Отличная статья Джона Фокса (John Fox) «Frames, environments, and scope in R and S-PLUS» (http://mng.bz/gxw8) поможет вам лучше понять области видимости. Статья Сураджа Гупты (Suraj Gupta) «How R searches and finds stuff» (http://mng.bz/2o5B) рассказывает о том, как R разрешает имена функций, объектов, переменных и т. д. Узнать больше об эффективном программировании можно в руководстве «FasteR! HigheR! StrongeR! – A Guide to Speeding Up R Code for Busy People» Ноама Росса (Noam Ross; http://mng.bz/Iq3i). Наконец, могут оказаться полезными две книги. «Advanced R» Хэдли Уикхема (Hadley Wickham; https://adv-r.hadley.nz/) и «R Programming for Bioinformatics» (Chap-

man & Hall, 2009) Роберта Джентльмена (Robert Gentleman) – всеобъемлющие и вполне доступные книги для программистов, которые хотят заглянуть под капот. Я настоятельно рекомендую их всем, кто хочет стать эффективным программистом на R.

Итоги

- R поддерживает самые разные структуры данных, включая атомарные и обобщенные векторы. Научиться извлекать из них информацию – ключевой навык для эффективного программирования на R.
- Каждый объект в R имеет атрибут `class` и ряд дополнительных свойств.
- Управляющие структуры, такие как `for()` и `if()`/`else()`, позволяют выполнять код условно и непоследовательно.
- Окружения предлагают механизм поиска имен объектов и их содержимого. Они обеспечивают точное управление областями видимости имен и функций.
- R поддерживает несколько систем объектно-ориентированного программирования. Наиболее распространенной на сегодняшний день является система S3.
- Нестандартная оценка позволяет программисту настроить порядок и время оценки выражений.
- Два основных способа ускорить код – векторизация и параллельная обработка. Профилирование кода поможет вам найти узкие места в смысле скорости выполнения.
- R имеет богатый набор инструментов отладки, а RStudio предоставляет графические интерфейсы для этих инструментов. Вместе они упрощают поиск ошибок в коде.

21 Создание динамических отчетов

В этой главе:

- публикация результатов в интернете;
- включение результатов вычислений из R в отчеты в Microsoft Word или Open Document;
- создание динамических отчетов, изменяющихся с изменением данных;
- создание документов издательского качества с помощью R, Markdown и LaTeX;
- как избежать распространенных ошибок R Markdown.

В предыдущих главах мы узнали, как получить доступ к своим данным, очистить их, описать, смоделировать отношения и визуализировать результаты. Следующий наш шаг:

1) отдохнуть и, может быть, отправиться в Дисней-ленд;
2) отправить полученные результаты другим.

Если вы выбрали пункт 1, то, пожалуйста, возьмите меня с собой. Если вы выбрали пункт 2, то добро пожаловать в реальный мир.

Исследования не заканчиваются с завершением последнего статистического анализа или построением графика. Вам почти всегда придется передавать результаты исследований другим. Это означает создание некоторого отчета.

Существует три распространенных сценария отчета. В первом вы создаете отчет, включающий ваш код и результаты, чтобы можно было вспомнить о том, что делалось, спустя шесть месяцев. Вспомнить детали проделанной работы всегда проще, имея перед глазами единый документ, чем набор взаимосвязанных файлов.

Во втором сценарии отчет создается для преподавателя, руководителя, клиента, государственного учреждения, интернет-аудитории или редактора журнала. Ясность и привлекательность в этом случае имеют особое значение, и такой отчет может быть создан только один раз.

В третьем сценарии требуется регулярно генерировать отчет определенного типа. Это может быть ежемесячный отчет об использовании продуктов или ресурсов, еженедельный финансовый анализ или ежечасно обновляемый отчет о веб-трафике. В любом случае данные меняются, но процедуры их анализа и структура отчета остаются прежними.

Один из подходов к внедрению выходных данных в отчет предполагает выполнение анализа, вырезание и вставку каждого графика и текстовой таблицы в текстовый документ и переформатирование результатов. Такой подход обычно отнимает много времени, он неэффективен и вызывает разочарование. Хотя R способен генерировать самую современную графику, его текстовый вывод выглядит довольно уныло – таблицы моноширинного текста со столбцами, выстроенными с использованием пробелов. Переформатировать их – непростая задача. И если данные изменятся, вам придется пройти весь процесс заново!

Учитывая эти ограничения, вы могли бы подумать, что R вам не подойдет. Но не спешите с выводами! (Хорошо, вы можете испытать чувство легкого опасения – это важный механизм выживания.) R предлагает элегантное решение для включения программного кода на R и результатов в отчеты с использованием языка разметки под названием *R Markdown* (https://rmarkdown.rstudio.com). Кроме того, данные можно привязать к отчету, чтобы при изменении данных менялся и сам отчет. Такие динамические отчеты можно сохранять как:

- веб-страницы;
- документы Microsoft Word;
- документы Open Document;
- слайды Beamer, HTML5 и PowerPoint;
- готовые к публикации документы PDF или PostScript.

Например, предположим, что вы используете регрессионный анализ для изучения взаимосвязи между весом и ростом в выборке `women`. R Markdown позволяет преобразовать моноширинный вывод, сгенерированный функцией `lm()`:

```
> lm(weight ~ height, data=women)

Call:
lm(formula = weight ~ height, data = women)

Residuals:
    Min      1Q  Median      3Q     Max
-1.7333 -1.1333 -0.3833  0.7417  3.1167

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -87.51667    5.93694  -14.74 1.71e-09 ***
height        3.45000    0.09114   37.85 1.09e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 1.525 on 13 degrees of freedom
Multiple R-squared:  0.991,     Adjusted R-squared:  0.9903
F-statistic:  1433 on 1 and 13 DF,  p-value: 1.091e-14
```

в веб-страницу, как на рис. 21.1. В этой главе вы узнаете, как это сделать.

Regression report

RobK

7/8/2021

Heights and weights

Linear regression was used to model the relationship between weights and height in a sample of 15 women. The equation **weight = -87.517 + 3.45 * height** accounted for 0.99% of the variance in weights. The ANOVA table is given below.

term	estimate	std.error	statistic	p.value
(Intercept)	-87.52	5.937	-14.7	0
height	3.45	0.091	37.9	0

The regression is plotted in the following figure.

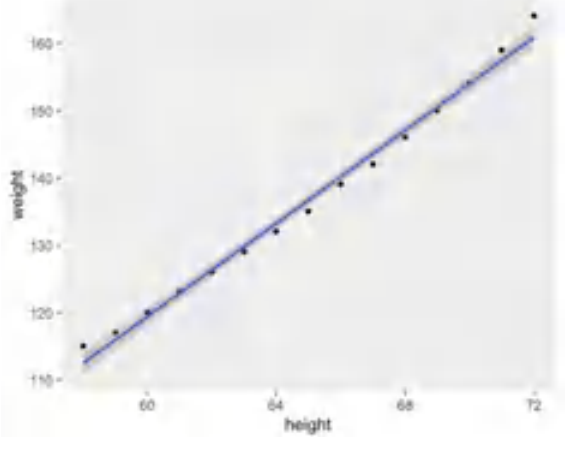


Рис. 21.1. Результаты регрессионного анализа, сохраненные в виде веб-страницы

Динамические документы и воспроизводимые исследования

В академическом сообществе растет мощное движение в поддержку воспроизводимых исследований. Цель воспроизводимых исследований – облегчить воспроизведение научных результатов путем включения необходимых данных и программного кода в публикации. Это позволяет читателям самостоятельно проверить полученные данные и дает возможность использовать полученные результаты непосредственно в своей работе. Методы, описанные в этой главе, включая встраивание в документы данных и исходного кода, напрямую поддерживают эту задачу.

21.1. Шаблонный подход к отчетам

В этой главе используется шаблонный подход к созданию отчетов, согласно которому сначала создается файл шаблона, содержащий текст отчета, команды форматирования и фрагменты кода на R.

Файл шаблона обрабатывается, код на R запускается, к результатам применяются команды форматирования, и в результате получается готовый отчет. Включением выходных данных в отчет управляют различные параметры. На рис. 21.2 показан простой пример шаблона R Markdown для создания веб-страницы.

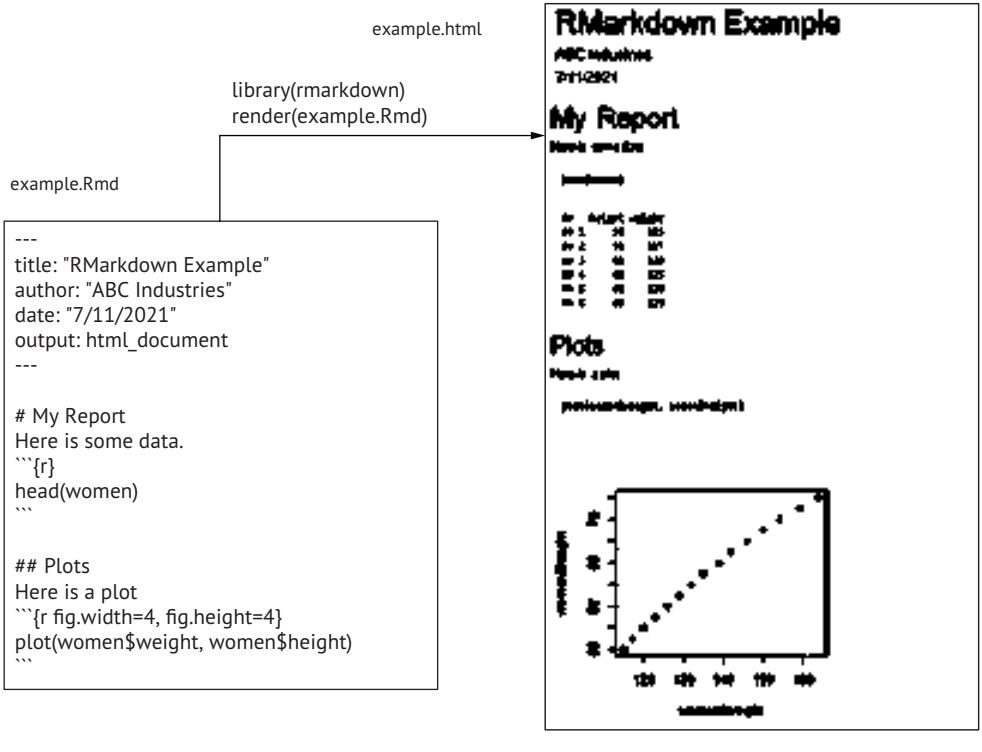


Рис. 21.2. Создание веб-страницы из текстового файла с разметкой Markdown, текстом отчета и фрагментами кода на R

Файл шаблона (example.Rmd) – это простой текстовый файл, содержащий четыре раздела:

- *метаданные* (заголовок YAML) заключаются в теги – пару из трех дефисов (`---`) и содержат информацию о документе и желаемом результате;
- *текст отчета* – любые поясняющие фразы и текст. Здесь текст отчета включает словосочетания: «My Report» (мой отчет), «Here is some data» (здесь располагаются некоторые данные), «Plots» (диаграммы) и «Here is a plot» (это диаграмма);
- *разметка* – теги, управляющие форматированием отчета. В этом файле для форматирования результатов используются теги Markdown. Markdown – простой язык разметки, который можно использовать для преобразования обычных текстовых файлов в структурно правильную разметку HTML или XHTML. Символ решетки # в первой строке не является комментарием. Он создает заголовок уровня 1. Два символа решетки ## создают заголовок уровня 2 и т. д.;
- *код на R* – операторы R, которые необходимо выполнить. В документах R Markdown фрагменты кода на R заключены в теги ```` ```{r} ```` и ```` ``` ````. В первом фрагменте кода перечислены первые шесть строк из набора данных, а во втором создается диаграмма рассеяния. В этом примере в отчет выводятся и код, и результаты, но параметры позволяют контролировать вывод по частям.

Файл шаблона передается в вызов функции `render()` из пакета `rmarkdown`, а та создает веб-страницу с именем example.html. Веб-страница содержит как текст отчета, так и результаты выполнения кода на R.

Примеры в этой главе основаны на задачах получения описательных статистик, регрессии и дисперсионного анализа. Ни один из примеров не является полным анализом данных. Цель этой главы – показать, как включать результаты выполнения кода на R в различные типы отчетов.

В зависимости от исходного файла шаблона и функций, используемых для его обработки, создаются отчеты в различных форматах (веб-страницы HTML, документы Microsoft Word, документы OpenOffice Writer, отчеты в формате PDF, статьи, слайды и книги). Отчеты являются динамическими в том смысле, что при изменении данных и повторной обработке файла шаблона создается новый отчет.

21.2. Создание отчета с помощью R и R Markdown

В этом разделе мы используем пакет `rmarkdown` для создания документов на основе разметки Markdown и кода на R. В процессе обработки документа выполняется код на R, а его вывод форматируется и встраивается в готовый документ. Этот подход можно ис-

пользовать для создания отчетов в самых разных форматах. Вот типичная последовательность шагов:

1. установить пакет `rmarkdown` (`install.packages("rmarkdown")`). Вместе с `rmakdown` будет установлено еще несколько пакетов, в том числе `knitr`. Если вы используете последнюю версию RStudio, то можете пропустить этот шаг, потому что все необходимые пакеты уже установлены;
2. установить Pandoc (http://johnmacfarlane.net/pandoc/), бесплатное приложение, доступное для Windows, macOS и Linux. Оно преобразует файлы из одного формата в другой. Опять же, пользователи RStudio могут пропустить этот шаг;
3. чтобы получить возможность создавать документы PDF, следует установить компилятор LaTeX, преобразующий документы LaTeX в документы PDF полиграфического качества. Дополнительно будут установлены MiKTeX (http://www.miktex.org/) для Windows, MacTeX для Mac (http://www.tug.org/texlive) и TeX Live для Linux (http://www.tug.org/texlive). Я рекомендую также установить кросс-платформенный пакет TinyTex (http://yihui.org/tinytex):

```
install.packages("tinytex")
tinytex::install_tinytex()
```

4. рекомендуется также (хотя это и не обязательно) установить пакет `broom` (`install.packages("broom")`). Функция `tidy()` из этого пакета может экспортировать результаты более чем 135 статистических функций R в таблицы данных для включения в отчеты. Полный список поддерживаемых объектов можно получить вызовом `method(tidy)`;
5. и наконец, `kableExtra` (`install.packages("kableExtra")`) установить. Функция `kable` из пакета `knitr` преобразует матрицу или таблицу данных в таблицу LaTeX или HTML для включения в отчет. Пакет `kableExtra` включает функции для дополнительного оформления таких таблиц.

После подготовки программного окружения можно приступать к работе.

Чтобы включить вывод кода на R (значения, таблицы, графики) в документ с использованием синтаксиса Markdown, сначала создайте текстовый документ, содержащий:

- заголовок YAML;
- текст отчета;
- разметку Markdown.
- фрагменты кода на R (окруженные разделителями).

Текстовый файл должен иметь расширение `.Rmd`.

В листинге 21.1 показан пример файла (с именем women.Rmd). Чтобы сгенерировать HTML-документ, выполните следующий код:

```
library(rmarkdown)
render("women.Rmd")
```

Или в RStudio щелкните на кнопке Knit. Результат показан на рис. 21.1.

Листинг 21.1. women.Rmd: шаблон с разметкой R Markdown и встроенным кодом на R

````
---                                                                    1
title: "Regression Report"                                             1
author: "RobK"                                                         1
date: "7/8/2021"                                                       1
output: html_document                                                  1
---                                                                    1

# Heights and weights                                                  2

```{r echo = FALSE}                                                    3
options(digits=3)                                                      3
n    <- nrow(women)                                                    3
fit  <- lm(weight ~ height, data=women)                                3
sfit <- summary(fit)                                                   3
b    <- coefficients(fit)                                              3
```                                                                    3

Linear regression was used to model the relationship between
weights and height in a sample of `r n` women. The equation            4
**weight = `r b[1]` +  `r b[2]` * height**                             4
accounted for `r round(sfit$r.squared,2)`% of the variance             4
in weights. The ANOVA table is given below.

```{r echo=FALSE}                                                      5
library(broom)                                                         5
library(knitr)                                                         5
library(kableExtra)                                                    5

results <- tidy(fit)                                                   5
tbl <- kable(results)                                                  5
kable_styling(tbl, "striped", full_width=FALSE, position="left")       5
```                                                                    5

The regression is plotted in the following figure.

```{r fig.width=5, fig.height=4}
library(ggplot2)
ggplot(data=women, aes(x=height, y=weight)) +
      geom_point() + geom_smooth(method="lm", formula=y~x)
```
````

1. **Заголовок YAML.**
2. **Заголовок уровня 2.**
3. **Фрагмент кода на R.**
4. **Код на R внутри текста.**
5. **Форматирование таблицы с результатами.**

Отчет начинается с заголовка YAML (1), в котором указываются название, автор, дата и выходной формат. Дата жестко запрограммирована. Чтобы динамически вставить текущую дату, нужно заменить `"7/8/2021"` на `"`r Sys.Date()`"` (включая двойные кавычки). В заголовке YAML обязательным является только поле `output`, определяющее формат выходного документа. В табл. 21.1 перечислены наиболее распространенные значения для этого поля. Полный список доступен на сайте RStudio (https://rmarkdown.rstudio.com/lesson-9.html).

Таблица 21.1. Поддерживаемые форматы выходных документов для преобразования разметки R Markdown

Значение	Описание
`html_document`	Документ HTML
`pdf_document`	Документ PDF
`word_document`	Документ Microsoft Word
`odt_document`	Документ Open Document Text
`rtf_document`	Документ Rich Text

Далее следует заголовок первого уровня (2). Согласно разметке, текст «Heights and Weights» (Рост и вес) должен быть напечатан крупным жирным шрифтом. В табл. 21.2 показано несколько примеров оформления разметки Markdown.

Таблица 21.2. Код Markdown и результат преобразования

Синтаксис Markdown	Вывод в формате HTML
`# Заголовок 1` `## Заголовок 2` `...` `###### Заголовок 6`	Заголовок 1 Заголовок 2 ... Заголовок 6
Пустые строки в тексте	Разделяет текст на абзацы
Два или более пробелов в конце строки	Добавляет разрыв строки
`*Имеется в виду*`	Имеется в виду
`**Только так и никак иначе**`	Только так и никак иначе
`* пункт 1` `* пункт 2`	 - пункт 1 - пункт 2

Окончание табл. 21.2

Синтаксис Markdown	Вывод в формате HTML
`1. пункт 1` `2. пункт 2`	 - пункт 1 - пункт 2
`[Google](http://google.com)`	Google
`![Текст подписи](путь к файлу изображения)`	<img src="путь к файлу изображения", alt="Текст подписи">
`\newpage`	Разрыв страницы – начинает новую страницу. (Это команда LaTeX, распознаваемая пакетом `rmarkdown`)

Далее идет фрагмент кода на R (3). Код на R в документах Markdown отделяется символами ```` ```{r options} ```` и ```` ``` ````. Когда файл обрабатывается, код на R выполняется, и на его место вставляются результаты. Параметр `echo=FALSE` удаляет код из вывода. В табл. 21.3 перечислены поддерживаемые параметры.

Таблица 21.3. Параметры фрагмента кода

Параметр	Описание
`echo`	Определяет необходимость включения (TRUE) или исключения (FALSE) кода на R из вывода
`results`	Вывод (asis) или сокрытие (hide) результатов
`warning`	Определяет необходимость включения (TRUE) или исключения (FALSE) предупреждений из вывода
`message`	Определяет необходимость включения (TRUE) или исключения (FALSE) информационных сообщений из вывода
`error`	Определяет необходимость включения (TRUE) или исключения (FALSE) сообщений об ошибках из вывода
`cache`	Сохранять результаты и перезапускать код, только если код изменился
`fig.width`	Ширина диаграммы (в дюймах)
`fig.height`	Высота диаграммы (в дюймах)

Простой код на R (возвращающий число или строку) также можно разместить непосредственно в тексте отчета. Такой встроенный код на R позволяет добавлять текст в отдельных предложениях. Встроенный код должен заключаться в теги `` `r `` и `` ` ``. В примере регрессии в первый абзац встроены размер выборки, формула прогнозирования и значение множественного коэффициента детерминации (R-квадрат) (4).

Следующий фрагмент кода на R создает хорошо отформатированную таблицу ANOVA (5). Функция `tidy()` из пакета `broom` экспортирует результаты регрессии в виде усовершенствованной таблицы

данных (tibble). Функция `kable()` из пакета `knitr` преобразует эту таблицу данных в HTML-код, а функция `kable_styling()` из пакета `kableExtra` устанавливает ширину и выравнивание таблицы, а также добавляет чередование цветов. Дополнительные параметры форматирования см. в `help(kable_styles)`.

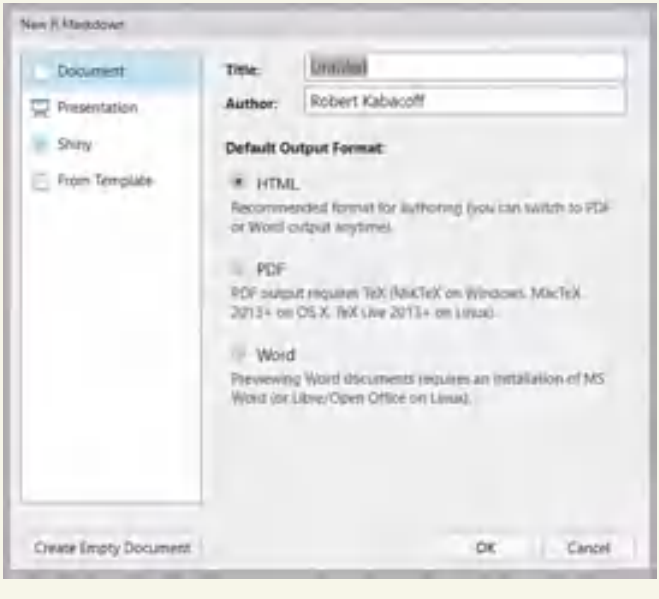
Функции `kable()` и `kable_styling()` просты и удобны в использовании. В R имеются пакеты, способные создавать более сложные таблицы и предлагающие множество вариантов оформления. К ним относятся `xtable`, `expss`, `gt`, `huxtable`, `flextable`, `pixiedust` и `stargazer`. У каждого есть свои достоинства и недостатки (http://mng.bz/aKy9 и http://mng.bz/gx08). Используйте пакеты, которые лучше всего соответствуют вашим потребностям.

Последняя часть файла R Markdown выводит диаграмму `ggplot2` с результатами. Размер диаграммы задается как 5 дюймов в ширину и 4 дюйма в высоту. По умолчанию выбирается размер 7 на 5 дюймов.

Создание и обработка документов R Markdown в RStudio

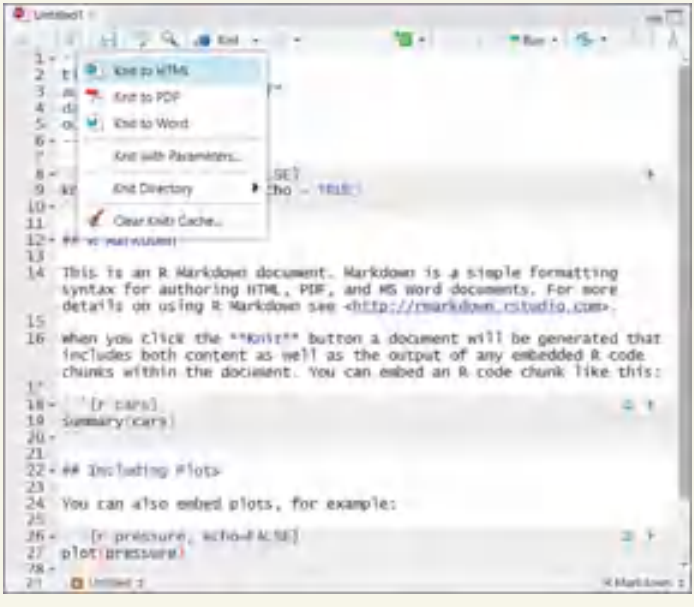
RStudio позволяет особенно легко отображать отчеты из документов Markdown.

Если выбрать пункт меню File > New File > R Markdown (Файл > Создать > R Markdown) в графическом интерфейсе, то появится диалог, показанный ниже.



Диалог создания нового документа K Markdown в RStudio

Выберите тип отчета, который вы хотите создать, и RStudio создаст заготовку файла шаблона. Отредактируйте его, а затем выберите параметр отображения в раскрывающемся списке Knit, и все!



Раскрывающееся меню, где можно выбрать тип отчета, HTML, PDF или Word, который должен создаваться на основе документа R Markdown

Синтаксис Markdown удобно использовать для быстрого создания простых документов. Чтобы узнать больше о Markdown, выберите в меню пункт HELP > Markdown Quick Reference (ПОМОЩЬ > Краткий справочник по Markdown) или посетите справочную страницу Pandoc Markdown (http://mng.bz/0r8E).

Если вы хотите создавать сложные документы, такие как статьи и книги полиграфического качества, то подумайте о возможности использования LaTeX в качестве языка разметки. В следующем разделе мы посмотрим, как с помощью LaTeX и пакета `knitr` создавать высококачественные документы.

21.3. Создание отчетов на R и LaTeX

LaTeX – это система подготовки документов, доступная бесплатно для Windows, macOS и Linux. LaTeX позволяет создавать красивые, сложные, составные документы и позволяет преобразовывать документы одного типа (например, статью) в документ другого типа (например, отчет) простой заменой нескольких строк кода. Это чрезвычайно мощное программное обеспечение и поэтому требует серьезного обучения.

Если вы незнакомы с LaTeX, то я рекомендую прочитать руководство «The Not So Short Introduction to LaTeX 2e»[1] Тобиаса Отикера

[1] Перевод на русский язык доступен по адресу: http://www.ccas.ru/voron/download/books/tex/oetiker99latex.pdf. – *Прим. перев.*

(Tobias Oetiker), Хьюберта Партла (Hubert Partl), Ирен Хайны (Irene Hyna) и Элизабет Шлегль (Elisabeth Schlegl), доступное по адресу https://tobi.oetiker.ch/lshort/lshort.pdf, или «LaTeX Tutorials: A Primer» на сайте Indian TEX Users Group (http://mng.bz/2c00). Этот язык определенно достоин изучения, но для его освоения потребуется некоторое время и терпение. Зато после знакомства с LaTeX создание динамических отчетов превратится для вас в простой процесс.

Пакет `knitr` позволяет встраивать код R в документ LaTeX, используя методы, аналогичные тем, что применяются для создания веб-страниц. Если вы установили `rmarkdown` или используете RStudio, то у вас уже есть пакет `knitr`. Если нет, то установите его сейчас (`install.packages("knitr")`). Кроме того, вам понадобится компилятор LaTeX; подробности см. в разделе 21.2.

В этом разделе мы создадим отчет, описывающий реакцию пациентов на различные лекарства, используя данные из пакета `multcomp`. Если вы не установили его в главе 9, то обязательно запустите `install.packages("multcomp")`, прежде чем продолжить.

Чтобы сгенерировать отчет с использованием R и LaTeX, сначала нужно создать текстовый файл с расширением `.Rnw`, содержащий текст отчета, код разметки LaTeX и фрагменты кода на R (пример показан в листинге 21.2). Каждый фрагмент кода на R начинается с разделителя `<<options>>=` и заканчивается разделителем `@`. В табл. 21.3 перечислены поддерживаемые параметры для фрагментов кода. Встроенный код R включается в текст с помощью тегов `\Sexpr{код на R}`. После выполнения такой код на R замещается возвращаемым им числом или строкой.

Затем этот файл обрабатывается функцией `knit2pdf()`:

```
library(knitr)
knit2pdf("drugs.Rnw")
```

или щелчком на кнопке Compile PDF (Скомпилировать PDF) в RStudio. На этом этапе обрабатываются фрагменты кода на R и, в зависимости от параметров, заменяются кодом на R в формате LaTeX и выводятся. По умолчанию `knit("drugs.Rnw")` вводит файл drugs.Rnw и выводит файл drugs.tex. Затем файл `.tex` передается компилятору LaTeX, который создает файл PDF. На рис. 21.3 показан полученный в результате документ PDF.

Листинг 21.2. drugs.Rnw: шаблон LaTeX со встроенным кодом на R

```
\documentclass[11pt]{article}
\title{Sample Report}
\author{Robert I. Kabacoff, Ph.D.}
\usepackage{float}
\usepackage[top=.5in, bottom=.5in, left=1in, right=1in]{geometry}
\begin{document}
\maketitle
```

```
<<echo=FALSE, results='hide', message=FALSE>>=
library(multcomp)
library(dplyr)
library(xtable)
library(ggplot2)
df <- cholesterol
@

\section{Results}

Cholesterol reduction was assessed in a study
that randomized \Sexpr{nrow(df)} patients
to one of \Sexpr{length(unique(df$trt))} treatments.
Summary statistics are provided in
Table \ref{table:descriptives}.

<<echo=FALSE, results='asis'>>=
descTable <- df %>%
  group_by(trt) %>%
  summarize(N = n(),
            Mean = mean(response, na.rm=TRUE),
            SD = sd(response, na.rm=TRUE)) %>%
  rename(Treatment = trt)
print(xtable(descTable, caption = "Descriptive statistics
for each treatment group", label = "table:descriptives"),
caption.placement = "top", include.rownames = FALSE)
@

The analysis of variance is provided in Table \ref{table:anova}.

<<echo=FALSE, results='asis'>>=
fit <- aov(response ~ trt, data=df)
print(xtable(fit, caption = "Analysis of variance",
             label = "table:anova"), caption.placement = "top")
@

\noindent and group distributions are plotted in Figure \ref{figure:tukey}.

\begin{figure}[H]\label{figure:tukey}
\begin{center}

<<echo=FALSE, fig.width=4, fig.height=3>>=
  ggplot(df, aes(x=trt, y=response )) +
    geom_boxplot() +
    labs(y = "Response", x="Treatment") +
    theme_bw()
@

\caption{Distribution of response times by treatment.}
\end{center}
\end{figure}
\end{document}
```

Документацию к пакету `knitr` можно найти по адресу http://yihui.name/knitr и в книге Ихуи Се (Yihui Xie) «Dynamic Documents with R and knitr» (Chapman & Hall, 2013). За более полной информацией о LaTeX обращайтесь к перечисленным выше руководствам и посетите сайт https://www.latex-project.org/.

21.3.1. Создание параметризованного отчета

Во время создания в отчет можно передавать дополнительные параметры, что позволяет настраивать вывод без изменения шаблона R Markdown. Параметры определяются в заголовке YAML, в разделе, обозначенном ключевым словом `params`. Значения параметров доступны в коде с применением нотации `$`.

Sample report

Robert I. Kabacoff, Ph.D.

July11,2021

1 Results

Cholesterol reduction was assessed in a study that randomized 50 patients to one of 5 treatments. Summary statistics are provided in Table 1.

Table 1: Descriptive statistics for each treatment group

Treatment	N	Mean	SD
1tim	10	5.78	2.88
2time	10	9.22	3.48
4times	10	12.37	2.92
drugD	10	15.36	3.45
drugE	10	20.95	3.35

The analysis of variance is provided in Table 2.

Table 2: Analysis of variance

	Df	Sum Sq	Mean Sq	F value	Pr(> F)
trt	4	1351.37	337.84	32.43	0.0000
Residuals	45	468.75	10.42		

and group distributions are plotted in Figure 1.

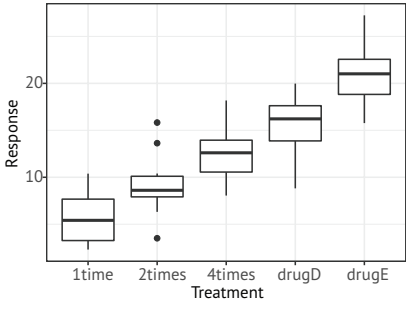


Figure 1: Distribution of response times by treatment.

1

Рис. 21.3. Текстовый файл drugs.Rnw обрабатывается с помощью функции knit2pdf(), в результате чего создается документ PDF (drugs.pdf)

Рассмотрим параметризованный отчет, представленный в листинге 21.3. Этот документ R Markdown загружает цены акций четырех крупных технологических компаний (Apple, Amazon, Google и Microsoft) из Yahoo Finance (https://finance.yahoo.com/) и отображает их с помощью `ggplot2` (рис. 21.4).

Рис. 21.4. Динамический отчет, созданный на основе параметризованного файла R Markdown, показанного в листинге 21.3

По умолчанию код анализирует изменение цен на акции за последние 30 дней с момента создания файла. Для получения исходных данных в отчете используется пакет `tidyquant`, поэтому установите его перед созданием документа (`install.packages("tidyquant")`).

Листинг 21.3. Отчет R Markdown с параметрами (techstocks.Rmd)

```
---
title: "Tech Stock Monthly Report"
author: "IT Department"
```

```
output: html_document
params:                                          (1)
  enddate: !r Sys.Date()                         (1)
  startdate: !r Sys.Date() - 30                  (1)

---
## Largest tech stocks

One month performance for

- Apple (AAPL)
- Amazon (AMZN)
- Alphabet/Google (GOOG)
- Microsoft (MSFT)

```{r, echo=FALSE, message=FALSE}

library(tidyquant)
library(ggplot2)

tickers = c("AAPL", "MSFT", "AMZN", "GOOG")

prices <- tq_get(tickers,                        (2)
                 from = params$startdate,        (2)
                 to = params$enddate,            (2)
                 get = "stock.prices")           (2)

ggplot(prices, aes(x=date, y=adjusted, color=symbol)) +
  geom_line(size=1) + facet_wrap(~symbol, scale="free") +
  theme_bw() +
  theme(legend.position="none") +
  scale_x_date(date_labels="%m-%d") +
  scale_color_brewer(palette="Set1") +
  labs(title="Stock Price Chart",
       subtitle = paste(params$startdate, "to", params$enddate),
       x = "Date",
       y="Adjusted Price")
```
Source: [Yahoo Finance](https://finance.yahoo.com/)
```

(1) Определение значений параметров.

(2) Получение котировок акций с учетом значений параметров.

В заголовке YAML определяются параметры `enddate` и `startdate` (1). Эти значения можно задать жестко или использовать `!r` для включения кода на R, который вычислит нужные значения. Здесь конечная дата устанавливается равной текущей дате, а начальная дата – на 30 дней раньше.

При запуске кода функция `tq_get()` из пакета `tidyquant` загрузит суточные цены акций (название, дата, открытие, максимум, минимум, закрытие, объем, скорректированная цена) для диапазона дат,

определяемого параметрами `params$startdate` и `params$enddate` (2). Затем значения отображаются в виде диаграмм с помощью `ggplot2`.

Параметры можно переопределить во время выполнения, передав список в функцию `render`. Например, следующий вызов:

```
render("techstocks.Rmd", params=list(startdate="2021-01-01",
                                     enddate="2019-01-31"))
```

отобразит посуточную динамику изменения стоимости акций за январь 2019 года. В качестве альтернативы, если вызвать `render("techstocks.Rmd", params="ask")` (или выбрать пункт меню Knit > Knit with parameters... (Сформировать > Сформировать с параметрами...) в RStudio), то вам будет предложено ввести значения `startdate` и `enddate`. Введенные вами значения будут использоваться вместо значений по умолчанию. Добавление поддержки параметров сделает ваши отчеты более интерактивными. За дополнительными подробностями обращайтесь к разделу 15.3 книги Ихуи Се (Yihui Xie), Дж. Дж. Аллера (J. J. Allaire) и Гаррета Гролемунда (Garret Grolemund) «R Markdown: The Definitive Guide» (Chapman & Hall, 2019). Онлайн-версия книги доступна по адресу https://bookdown.org/yihui/rmarkdown/.

21.4. Преодоление типичных проблем с R Markdown

R Markdown – мощный инструмент создания динамичных и привлекательных отчетов в R. Но многие допускают типичные ошибки, которых следует избегать (табл. 21.4).

Таблица 21.4. Преодоление типичных проблем с R Markdown

Правило	Верно	Не верно
Отступы в заголовке YAML имеют значение. С отступами должны оформляться только поля	`---` `title: "Tech Stock Monthly Report"` `author: "IT Department"` `output: html_document` `params:` `  enddate: 2019-01-31` `  startdate: 2019-01-01` `---`	`---` `title: "Tech Stock Monthly Report"` `author: "IT Department"` `output: html_document` `params:` `enddate: 2019-01-31` `startdate: 2019-01-01` `---`
За тегом оформления заголовка должен следовать пробел	`# Это заголовок первого уровня`	`#Это заголовок первого уровня`

Окончание табл. 21.4

Правило	Верно	Не верно
Списки должны предваряться и завершаться пустой строкой, а за тегом звездочки должен следовать пробел	Это список * пункт 1 * пункт 2	Это список *пункт 1 *пункт 2
При необходимости фрагменты кода на R можно помечать метками, но метки должны быть уникальными	```{r anova1} Код на R ``` ```{r anova2} Код на R ```	```{r anova} Код на R ``` ```{r anova} Код на R ```
В фрагментах кода на R нельзя устанавливать пакеты	Установка пакета ggplot2 за пределами документа R Markdown, затем: ```r{} library(ggplot2) ```	```r{} install.packages(ggplot2) library(ggplot2) ```

Отлаживать ошибки, возникающие при отображении документов R Markdown, часто намного сложнее, чем отлаживать ошибки в простом коде на R. Но если предварительно убедиться, что каждый фрагмент кода на R работает сам по себе, и проявить осторожность, чтобы предотвратить ошибки, описанные выше, то отладка может и не понадобиться.

Прежде чем двигаться дальше, следует упомянуть еще одну проблему, обусловленную низкой эффективностью. Допустим, вы создали документ R Markdown, содержащий вводный текст и фрагмент кода на R, который выполняет сложный анализ. После правки вводного текста при пересоздании документа код на R будет выполнен повторно, даже притом что результаты не изменились.

Чтобы избежать этого, можно настроить кеширование результатов вычислений. Добавьте в фрагмент кода параметр `cache = TRUE` – и R сохранит результаты в папке, благодаря чему код будет повторно запускаться, только если он сам изменится. Если код не изменился, то будут использоваться сохраненные результаты. Но будьте осторожны: кеширование не учитывает изменения в базовых данных, только в самом коде. Если данные изменятся, а имя файла останется прежним, то код не будет запущен повторно. В подобных случаях

можно добавить параметр `cache.extra = file.mtime(mydatafile)`, где `mydatafile` – это путь к набору данных. Если временная метка файла данных изменится, то фрагмент кода будет запущен повторно.

21.5. Дополнительная информация

В этой главе мы увидели несколько способов включения в отчеты результатов выполнения кода на R. Отчеты являются динамическими в том смысле, что изменение данных и повторное выполнение кода приводят к обновлению отчета. Кроме того, отчет можно изменить, передав параметры. Мы изучили методы создания веб-страниц, верстки документов, отчетов в формате Open Document Format и документов Microsoft Word.

Шаблонный подход, описанный в этой главе, имеет несколько преимуществ. Внедрив код, выполняющий статистический анализ, непосредственно в шаблон, можно точно увидеть, как были получены результаты. Спустя полгода это поможет вам вспомнить, что было сделано. Вы также сможете изменить статистический анализ или добавить новые данные и сразу же заново сгенерировать отчет. Кроме того, вам не понадобится вырезать, вставлять и переформатировать результаты. Уже одно это стоит времени, затраченного на изучение новых возможностей.

Шаблоны в этой главе являются статическими в том смысле, что их структура фиксирована. Однако те же методы можно использовать для создания различных систем экспертных отчетов. Например, вывод, созданный фрагментом кода на R, может зависеть от отправленных данных. Если представлены числовые переменные, то можно создать матрицу диаграмм рассеяния. Если представлены категориальные переменные, то можно построить мозаичную диаграмму. Аналогично пояснительный текст может генерироваться в зависимости от результатов анализа. Использование конструкции `if/then` и функций конструирования текста, таких как `grep` и `substr`, делает возможности настройки почти безграничными. По сути, шаблон сохраняется во временном хранилище и редактируется с помощью кода перед формированием отчета. Применив этот подход, можно создать довольно сложную экспертную систему.

Узнать больше о R Markdown можно в книгах Ихуи Се (Yihui Xie), Кристоф Дервье (Christophe Dervieux) и Эмили Ридерер (Emily Riederer) «R Markdown Cookbook» (https://bookdown.org/yihui/rmarkdown-cook book/) и Ихуи Се (Yihui Xie), Дж. Дж. Аллера (J.J. Allaire) и Гаррета Гролемунда (Garret Grolemund) «R Markdown: The Definitive Guide» (https://bookdown.org/yihui/rmarkdown/), а также на веб-сайте RStudio R Markdown (https://rmarkdown.rstudio.com/).

Итоги

- R Markdown можно использовать для создания привлекательных отчетов, включающих пояснительный текст, код и результаты вычислений.
- Отчеты могут выводиться в форматах HTML, Word, Open Document, RTF и PDF.
- Документы R Markdown можно параметризовать – передавать дополнительные параметры в код во время формирования отчета и тем самым создавать более динамичные отчеты.
- Для создания сложных отчетов с широкими возможностями настройки можно использовать язык разметки LaTeX. Но кривая обучения может быть очень крутой.
- Для повышения воспроизводимости исследований, поддержки документации и улучшения передачи результатов можно использовать шаблонный подход с R Markdown (и LaTeX).

22 Создание пакетов

В этой главе:

- создание функций для пакета;
- добавление документации с описанием пакета;
- создание пакета и его распространение.

В предыдущих главах мы решали свои задачи, в основном используя функции, написанные другими. Функции брались из пакетов, входящих в состав дистрибутива R, или из репозитория CRAN.

Установка нового пакета расширяет возможности R. Например, установка пакета `mice` открывает новые возможности обработки отсутствующих данных, а установка пакета `ggplot2` – визуализации данных. Многие из самых мощных возможностей R реализованы в дополнительных пакетах.

Технически пакет – это просто набор функций, документации и данных, хранимых вместе в стандартизированном формате. Пакет позволяет организовать функции определенным и документированным образом и облегчает обмен вашими программами с другими.

Есть несколько причин, почему может понадобиться создать пакет:

- сделать легкодоступным набор часто используемых функций вместе с документацией по их использованию;
- создать набор примеров и данных, которые можно раздать учащимся в классе;

- создать программу (набор взаимосвязанных функций), которую можно использовать для решения серьезной аналитической задачи (например, подстановки пропущенных значений);
- способствовать воспроизводимости исследований путем организации данных, кода и документации в переносимом и стандартизированном формате.

Создание полезного пакета – также отличный способ заявить о себе и отблагодарить сообщество R. Пакеты можно распространять напрямую или через онлайн-репозитории, такие как CRAN и GitHub.

В этой главе мы попробуем разработать свой пакет от начала до конца. К концу главы вы научитесь создавать свои пакеты для R (и сможете похвастаться перед другими своими умениями).

Наш пакет мы назовем `edatools`. Он будет содержать функции для описания содержимого набора данных. Функции будут преднамеренно простыми, чтобы мы могли сосредоточиться на процессе создания пакета, не увязая в деталях реализации. В разделе 22.1 мы проведем тест-драйв существующего пакета `edatools`, а в разделе 22.2 создадим свою копию пакета с нуля.

22.1. Пакет edatools

Разведочный анализ данных (Exploratory Data Analysis, EDA) – это один из подходов к изучению данных путем описания их характеристик путем получения статистических оценок и диаграмм. Пакет `edatools` является небольшим подмножеством более обширного пакета разведочного анализа под названием `qacBase` (http://rkabacoff.github.io/qacBase).

Пакет `edatools` содержит функции, описывающие и визуализирующие содержимое набора данных, а также набор данных с именем `happiness`, включающий результаты опроса 460 человек об удовлетворенности жизнью. В результатах опроса содержатся оценки согласия людей с утверждением «Я счастлив большую часть времени», а также демографические переменные, такие как доход, образование, пол, раса и количество детей. Оценки выставлялись по 6-балльной шкале от «полностью не согласен» (1) до «полностью согласен» (6). Данные являются фиктивными и включены в состав пакета, исключительно чтобы дать возможность поэкспериментировать с набором данных, содержащим переменные нескольких типов, которые имеют пропущенные данные.

Установить пакет `edatools` можно инструкцией

```
if(!require(remotes)) install.packages("remotes")
remotes::install_github("rkabacoff/edatools")
```

Она загрузит и установит пакет из репозитория GitHub.

В пакете `edatools` есть одна главная функция с именем `contents()`, которая собирает информацию о наборе данных, а также функции

print() и plot(), отображающие результаты. Эти функции показаны в листинге 22.1, а на рис. 22.1 представлена полученная диаграмма.

Листинг 22.1. Описание набора данных с помощью пакета edatools

```
> library(edatools)
> help(contents)
> df_info <- contents(happiness)
> df_info

Data frame: happiness
460 observations and 11 variables
size: 0.1 Mb
 pos   varname      type n_unique n_miss pct_miss
   1       ID* character      460      0    0.000
   2      Date      Date       12      0    0.000
   3       Sex    factor        2      0    0.000
   4      Race    factor        8     92    0.200
   5       Age   integer       73     46    0.100
   6 Education    factor       13     23    0.050
   7    Income   numeric      415     46    0.100
   8        IQ   numeric       45    322    0.700
   9       Zip character       10      0    0.000
  10  Children   integer       11      0    0.000
  11     Happy   ordered        6     18    0.039

> plot(df_info)
```

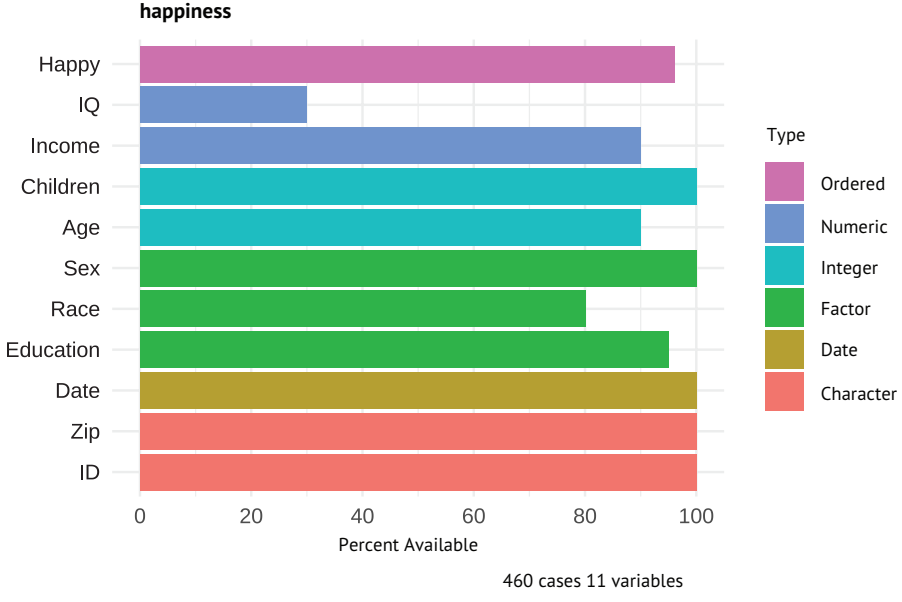


Рис. 22.1. Диаграмма, иллюстрирующая содержимое набора данных happiness. Всего в наборе данных имеется 11 переменных. В переменной IQ пропущено 70 % данных, в переменной Race – 20 %. Набор содержит шесть разных типов данных (упорядоченные факторы, числовые, целочисленные и т. д.)

Текстовый вывод отражает количество переменных и наблюдений, а также размер набора данных в мегабайтах. Для каждой переменной в наборе данных сообщаются: ее позиция, имя, тип, количество уникальных значений, а также количество и процент пропущенных значений. Переменные, которые могут служить уникальными идентификаторами (когда строка имеет уникальное значение переменной), отмечены звездочками. На диаграмме переменные упорядочены и выделены цветом по типу, а длина столбиков соответствует проценту наблюдений, доступных для анализа.

Глядя на таблицу и диаграмму, можно сказать, что набор данных содержит 11 переменных и 460 наблюдений. Он занимает 0,1 мегабайта в оперативной памяти. `ID` – это уникальный идентификатор, в данных есть 10 почтовых индексов, и 70 % наблюдений не имеют значения в переменной IQ. Как и ожидалось, рейтинг довольства жизнью (`happy`) – это упорядоченный фактор с шестью уровнями. Что еще вы заметили?

В следующих разделах я опишу общие шаги по созданию пакета для R, а затем мы вместе выполним их, чтобы создать пакет `edatools` с нуля.

22.2. Создание пакета

Когда-то создание пакета для R было сложной задачей, решить которую было под силу только высококвалифицированным профессионалам. С появлением удобных инструментов разработки этот процесс стал намного проще. И все же это не самый простой процесс. Вот основные шаги, которые необходимо выполнить:

1) установить средства разработки;
2) создать проект пакета;
3) добавить функции;
4) добавить документацию с описанием функций;
5) добавить общий файл справки;
6) добавить образцы данных и документацию с их описанием;
7) добавить виньетку;
8) отредактировать файл `DESCRIPTION`;
9) собрать и установить пакет.

Шаги 5–7 необязательны, но их выполнение считается хорошим тоном в разработке пакетов. В следующих разделах мы выполним все эти шаги по очереди. Вы также можете загрузить готовый пакет, доступный по адресу http://rkabacoff.com/RiA/edatools.zip, и сэкономить время на вводе текста.

22.2.1. Установка средств разработки

В этой главе я буду предполагать, что вы используете RStudio для сборки пакета. Кроме того, есть дополнительные пакеты поддержки, которые желательно установить. Сначала установите пакеты `devtools`, `usethis` и `roxygen2` с помощью инструкции `install.packages(c("devtools", "usethis", "roxygen2"), depend=TRUE)`. Пакеты `devtools` и `usethis` содержат функции, упрощающие и автоматизирующие разработку пакетов. Пакет `roxygen2` упрощает создание документации пакета.

Необходимость остального программного обеспечения зависит от ситуации:

- документация для пакета обычно составляется в формате HTML. Если вы решите создать документацию в формате PDF, то вам потребуется LaTeX – высококачественная система набора текста. Доступно несколько дистрибутивов с этой системой. Я рекомендую TinyTex (https://yihui.org/tinytex/), легкий кросс-платформенный дистрибутив. Его можно установить с помощью следующих инструкций:

```
install.packages("tinytex")
tinytex::install_tinytex()
```

- пакет `pkgdown` поможет создать веб-сайт для вашего пакета. Этот необязательный шаг описан в разделе 22.3.4;
- наконец, если вы работаете на платформе Windows и собираетесь создавать пакеты R, включающие код на C, C++ или Fortran, то вам понадобится установить Rtools.exe (http://cran.r-project.org/bin/windows/Rtools). Инструкции по установке вы найдете на странице загрузки. У пользователей macOS и Linux уже есть все необходимые инструменты. Я кратко опишу использование внешнего скомпилированного кода в разделе 22.4, но в этой книге мы не будем использовать его.

22.2.2. Создание проекта пакета

Следующий шаг после установки необходимых инструментов – создание проекта пакета. Если выполнить следующий код:

```
library(usethis)
create_package("edatools")
```

будет создан проект RStudio со структурой папок, показанной на рис. 22.2. После чего вы автоматически попадете в проект, а текущим рабочим каталогом станет папка `edatools`.

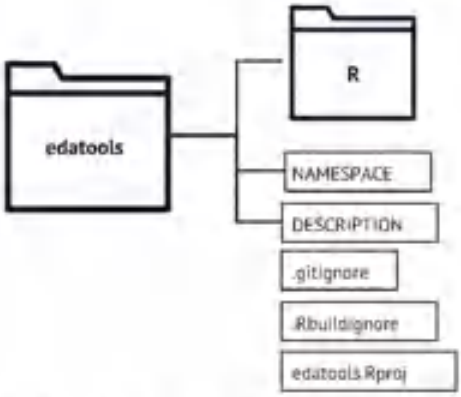


Рис. 22.2. Содержимое папки проекта edatools, созданного функцией `create_project()`

Код функции будет находиться в папке `R`. Другими важными файлами являются `DESCRIPTION` и `NAMESPACE`. Мы обсудим их в следующих разделах. `.gitignore`, `.Rbuildigore` и `edatools.Rproj` – это вспомогательные файлы, предназначенные для настройки процесса создания пакета. Их можно оставить как есть.

22.2.3. Написание функций для пакета

Пакет `edatools` состоит из трех функций: `contents()`, `print.contents()` и `plot.contents()`. Первая – это основная функция, которая собирает информацию о наборе данных, а остальные – объектно-ориентированные обобщенные функции S3 (раздел 20.4.1), используемые для отображения результатов в текстовом и графическом виде.

Файлы с кодом функций помещаются в папку `R`, созданную в разделе 22.2.2. Рекомендуется каждую функцию помещать в отдельный текстовый файл с расширением `.R`. Это не обязательно, но упрощает организацию работы. Кроме того, необязательно, чтобы имена функций и имена файлов совпадали, но опять же, это считается хорошей практикой. Каждый файл имеет заголовок, состоящий из набора комментариев, начинающихся с символов `#'`. Интерпретатор R игнорирует эти строки, но их использует пакет `roxygen2`, превращая комментарии в документацию. Эти заголовочные комментарии будут обсуждаться в разделе 22.2.4.

Содержимое файлов представлено в листингах с 22.2 по 22.4.

Сбор информации о наборе данных

Функция `contents()` в текстовом файле `contents.R` собирает сведения о наборе данных и сохраняет результаты в списке. Она вызывается как `contents(data)`, где `data` – это имя набора данных в рабочем окружении.

Листинг 22.2. Содержимое файла contents.R

```
#' @title Описание набора данных
#' @description
#' \code{contents} генерирует описание для набора
#' данных.
#'
#' @param data набор данных.
#'
#' @importFrom utils object.size
#'
#' @return список с 4 компонентами:
#' \describe{
#' \item{dfname}{имя набора данных}
#' \item{nrow}{количество строк}
#' \item{ncol}{количество столбцов}
#' \item{size}{объем данных в байтах}
#' \item{varinfo}{общие характеристики набора данных}
#' }
#'
#' @details
#' Для каждой переменной в наборе данных  \code{contents} сообщает
#' позицию, тип, число уникальных значений, число пропущенных
#' значений и процент пропущенных значений.
#'
#' @seealso \link{print.contents}, \link{plot.contents}
#'
#' @export
#'
#' @examples
#' df_info <- contents(happiness)
#' df_info
#' plot(df_info)

contents <- function(data){

  if(!(is.data.frame(data))){                                    1
    stop("You need to input a data frame")                       1
  }

  dataname <- deparse(substitute(data))                          2

  size <- object.size(data)                                      3

  varnames <- colnames(data)                                     4
  colnames <- c("pos", "variable", "type", "n_unique",           4
                "n_miss", "pct_miss")                            4
  pos = seq_along(data)                                          4
  varname <- colnames(data)                                      4
  type = sapply(data, function(x)class(x)[1])                    4
  n_unique = sapply(data, function(x)length(unique(x)))          4
  n_miss = sapply(data, function(x)sum(is.na(x)))                4
```

```
  pct_miss = n_miss/nrow(data)                                    4
  varinfo <- data.frame(                                          4
    pos, varname, type, n_unique, n_miss, pct_miss                4
  )                                                               4

  results <- list(dfname=dataname, size=size,                     5
                  nrow=nrow(data), ncol=ncol(data),               5
                  varinfo=varinfo)                                5
  class(results) <- c("contents", "list")                         5
  return(results)                                                 5
}                                                                 5
```

1. **Проверка ввода.**
2. **Сохранение названия набора данных.**
3. **Получение размера набора данных.**
4. **Сбор информации о переменных.**
5. **Возврат результатов.**

Когда вызывается эта функция, она проверяет, является ли входной аргумент набором данных. Если нет, то генерируется сообщение об ошибке 1. Затем сохраняются имя набора данных 2 и его размер в байтах 3. Для каждой переменной определяются позиция, имя, тип, количество уникальных значений, а также количество и процент пропущенных значений и сохраняются в таблице данных с именем `varinfo` 4. Наконец, результаты объединяются и возвращаются в виде списка 5. Список содержит пять компонентов (табл. 22.1). Кроме того, для списка устанавливается класс `c("contents", "list")`. Это важный шаг в создании обобщенных функций обработки результатов.

Таблица 22.1. Объект списка, возвращаемый функцией contents()

Компонент	Описание
`dfname`	Имя набора данных
`size`	Объем набора данных
`nrow`	Количество строк (наблюдений)
`ncol`	Количество столбцов (переменных)
`varinfo`	Информация о переменных в наборе данных (позиция, имя, тип, количество уникальных значений, а также количество и процент пропущенных значений)

Список содержит всю необходимую информацию, но обращаться напрямую к компонентам списка неудобно. Поэтому создадим обобщенные функции `print()` и `plot()`, представляющие эту информацию более лаконичным и осмысленным образом. Далее мы рассмотрим эти функции.

Функции вывода информации в текстовом и графическом виде

Большинство аналитических пакетов включают обобщенные функции `print()` и `summary()`. Первая из них выводит основную информацию об объекте, а вторая – более подробную и обобщенную информацию. Функция `plot()` часто включается в пакеты, когда представление данных в виде диаграммы имеет смысл. Для этого пакета мы напишем только функции `print` и `plot`.

Следуя рекомендациям модели объектно-ориентированного программирования S3, описанным в разделе 20.4.1: если объект имеет атрибут класса `foo`, то `print(x)` вызывает функцию `print.foo(x)`, если та существует, и `print.default(x)` в противном случае. То же касается `plot()`. Поскольку функция `contents()` возвращает объект класса `"contents"`, нам необходимо определить функции `print.contents()` и `plot.contents()`. Функция `print.contents()` представлена в листинге 22.3, а функция `plot.contents()` – в листинге 22.4.

Листинг 22.3. Содержимое файла print.R

```
#' @title Описание набора данных
#'
#' @description
#' \code{print.contents} выводит краткую информацию о наборе данных.
#'
#' @param x объект класса \code{contents}.
#' @param digits число значимых разрядов для вывода процентов.
#' @param ... дополнительные аргументы для функции print.
#'
#' @return NULL
#' @method вывод содержимого в текстовом виде
#' @export
#'
#' @examples
#' df_info <- contents(happiness)
#' print(df_info, digits=4)

print.contents <- function(x, digits=2, ...){

  if (!inherits(x, "contents"))                                    (1)
    stop("Object must be of class 'contents'")                     (1)

  cat("Data frame:", x$dfname, "\n")                               (2)
  cat(x$nrow, "observations and", x$ncol, "variables\n")
  x$varinfo$varname <- ifelse(x$varinfo$n_unique == x$nrow,        (3)
                              paste0(x$varinfo$varname, "*"),      (3)
                              x$varinfo$varname)
  cat("size:",  format(x$size, units="Mb"), "\n")                  (4)
  print(x$varinfo, digits=digits, row.names=FALSE, ...)            (5)
}
```

(1) **Проверка ввода.**

2 Вывод заголовка.

3 Определение уникальных идентификаторов.

4 Вывод размера набора данных.

5 Вывод информации о переменных.

Сначала функция проверяет соответствие входных данных классу `"contents"` 1. Затем выводит заголовок с именем набора данных и количеством переменных и наблюдений 2. Если количество уникальных значений в столбце равно количеству наблюдений, то эта переменная однозначно идентифицирует каждый случай. Такие переменные отмечаются звездочкой (`*`) 3. Далее выводится объем набора данных в мегабайтах 4. Наконец, выводится таблица с информацией о переменной информации 5.

Последняя функция, `plot()`, отображает данные, возвращаемые функцией `contents()`, в графическом виде – в форме столбиковой диаграммы (листинг 22.4).

Листинг 22.4. Содержимое файла plot.R

```
#' @title Визуальное представление набора данных
#'
#' @description
#' \code{plot.contents} визуализация переменных в наборе данных.
#'
#' @details
#' Для каждой переменной plot отображает
#' \itemize{
#'   \item{тип (\code{числовой},
#'               \code{целочисленный},
#'               \code{фактор},
#'               \code{упорядоченный фактор},
#'               \code{логический} или \code{дата})}
#'   \item{процент доступных (и отсутствующих) наблюдений}
#' }
#' Переменные сортируются по типу и в заголовке выводятся:
#' общее число переменных и наблюдений.
#'
#' @param x объект класса \code{contents}.
#' @param ... дополнительные аргументы (в настоящее время не используются).
#'
#' @import ggplot2
#' @importFrom stats reorder
#' @method plot contents
#' @export
#' @return a \code{ggplot2} graph
#' @seealso See \link{contents}.
#' @examples
#' df_info <- contents(happiness)
#' plot(df_info)
```

```
plot.contents <- function(x, ...){
  if (!inherits(x, "contents"))                                          1
    stop("Object must be of class 'contents'")                           1

  classes <- x$varinfo$type                                              2

  pct_n <-  100 *(1 - x$varinfo$pct_miss)                                2

  df <- data.frame(var = x$varinfo$varname,                              2
                   classes = classes,                                    2
                   pct_n = pct_n,                                        2
                   classes_n = as.numeric(as.factor(classes)))           2

  ggplot(df,                                                             3
         aes(x=reorder(var, classes_n),
             y=pct_n,
             fill=classes)) +
    geom_bar(stat="identity") +
    labs(x="", y="Percent Available",
         title=x$dfname,
         caption=paste(x$nrow, "cases",
                       x$ncol, "variables"),
         fill="Type") +
    guides(fill = guide_legend(reverse=TRUE)) +                          4
    scale_y_continuous(breaks=seq(0, 100, 20)) +
    coord_flip() +                                                       5
    theme_minimal()
}
```

1. **Проверка ввода.**
2. **Подготовка данных для диаграммы.**
3. **Вывод столбиковой диаграммы.**
4. **Переупорядочивание элементов легенды.**
5. **Смена местами осей *x* и *y*.**

И снова первым делом функция проверяет класс полученного объекта 1. Затем имена переменных, классы и процент непропущенных данных помещаются в таблицу данных. Класс каждой переменной добавляется в виде числа, которое используется для сортировки переменных по типу при построении графика 2. Данные отображаются в виде столбиковой диаграммы 3, а порядок условных обозначений в легенде меняется на противоположный, поэтому они оказываются выровненными по вертикали со столбиками 4. Наконец, оси *x* и *y* меняются местами, чтобы получить горизонтальную столбиковую диаграмму 5. Когда набор данных содержит много переменных или переменные имеют длинные имена, такой прием поможет избежать перекрытия меток.

22.2.4. Добавление документации с описанием функций

Каждый пакет R следует одному и тому же набору обязательных правил документирования. Все функции в пакете должны быть задокументированы единообразно с использованием LaTeX. Каждая функция помещается в отдельный файл с расширением `.R`, а документация для этой функции (написанная в LaTeX) помещается в файл с расширением `.Rd`. Оба файла, `.R` и `.Rd`, являются текстовыми.

У этого подхода есть два ограничения. Во-первых, документация хранится отдельно от описываемых функций. Если меняется код функции, то вы должны найти соответствующую документацию и изменить ее. Во-вторых, вы должны уметь пользоваться LaTeX. Если вы думали, что у R крутая кривая обучения, то не спешите с выводами, пока не начали работать с LaTeX!

Пакет `roxygen2` может существенно упростить создание документации. В начало каждого файла `.R` мы поместили заголовочный комментарий. Он будет служить документацией для функций. Эти комментарии начинаются с символов `#'` и используют простой набор тегов разметки. В табл. 22.2 перечислены наиболее часто используемые теги. Интерпретатор R воспринимает эти строки как комментарии и игнорирует их. Но когда файл обрабатывается пакетом `roxygen2`, то строки, начинающиеся с `#'`, используются для автоматического создания файлов документации в формате LaTeX (`.Rd`).

Таблица 22.2. Теги, поддерживаемые пакетом roxygen2

Тег	Описание
`@title`	Название функции
`@description`	Краткое описание функции
`@details`	Подробное описание функции (с отступами после первой строки)
`@parm`	Параметры функции
`@export`	Делает функцию доступной для пользователей пакета
`@import`	Пакеты, импортируемые целиком для использования в функциях данного пакета
`@importFrom`	Отдельные импортируемые функции, которые используются в функциях данного пакета
`@return`	Значение, возвращаемое функцией
`@author`	Автор(ы) и контактные данные
`@examples`	Примеры использования функции
`@note`	Примечания, касающиеся работы функции
`@references`	Ссылки на методологию, используемую функцией
`@seealso`	Ссылки на связанные функции

В дополнение к тегам в табл. 22.2 при создании документации полезно знать еще несколько элементов разметки:

- `\code{`*`text`*`}` выводит *`text`* моноширинным шрифтом;
- `\link{`*`function`*`}` генерирует гипертекстовую ссылку на справку функции *`function`*;
- `\href{`*`URL`*`}{`*`text`*`}` добавляет гиперссылку;
- `\item{`*`text`*`}` можно использовать для создания маркированного списка.

Комментарии для `roxygen2` из функции `contents()` в листинге 22.2 повторно приводятся в листинге 22.5. Здесь сначала указываются имя и описание функции (1). Далее описываются параметры (2). Теги могут следовать в любом порядке.

Листинг 22.5. Комментарии для roxygen2 в исходном коде функции `contents()`

```
#' @title Описание набора данных                                    1
#'                                                                  1
#' @description                                                     1
#' \code{contents} генерирует описание для набора                   1
#' данных.                                                          1
#'
#' @param data набор данных.                                        2
#'
#' @importFrom utils object.size                                    3
#'
#' @return список с 4 компонентами:                                 4
#' \describe{                                                       4
#' \item{dfname}{имя набора данных}                                 4
#' \item{nrow}{количество строк}                                    4
#' \item{ncol}{количество столбцов}                                 4
#' \item{size}{обьем данных в байтах}                               4
#' \item{varinfo}{общие характеристики набора данных}               4
#' }
#'
#' @details
#' Для каждой переменной в наборе данных  \code{contents} сообщает
#' позицию, тип, число уникальных значений, число пропущенных
#' значений и процент пропущенных значений.
#'
#' @seealso \link{print.contents}, \link{plot.contents}
#'
#' @export                                                          5
#'
#' @examples
#' df_info <- contents(happiness)
#' df_info
#' plot(df_info)
```

1. **Имя и описание.**
2. **Параметры.**
3. **Импортируемые функции.**
4. **Возвращаемое значение.**
5. **Экспорт функции.**

Тег `@importFrom` сообщает, что эта функция использует функцию `object.size()` из пакета `utils` 3. Используйте `@importFrom`, если хотите сделать доступным данному пакету лишь ограниченный набор внешних функций. Используйте `@import`, чтобы сделать доступными *все* функции в указанном пакете. Например, функция `plot.contents()` использует тег `@import ggplot2`, чтобы сделать все функции `ggplot2` доступными для этой функции.

Тег `@return` указывает, какие данные возвращает функция (в нашем случае список) 4. Это описание появится в справке под заголовком `Value`. Используйте `@details`, чтобы предоставить дополнительную информацию о функции. В теге `@seealso` можно определить гиперссылки на другие функции. Тег `@export` делает функцию доступной для пользователя 5. Если его не указать, то функция будет доступна другим функциям в вашем пакете, но пользователи не смогут вызвать ее из командной строки. Для функций, которые не должны вызываться непосредственно пользователем, тег `@export` можно опустить.

Наконец, тег `@examples` позволяет включить примеры, демонстрирующие применение функции. Обратите внимание, что это должны быть работающие примеры! Если примеры служат исключительно описательным целям, заключите их в тег `\dontrun{}`. Например, опишите пример так:

```
@examples
\dontrun{
contents(prettyflowers)
}
```

если набора данных с именем `prettyflowers` не существует. Поскольку код заключен в тег `\dontrun{}`, он не выдаст ошибку. Разметка `\dontrun{}` часто используется также, когда выполнение примера требует много времени.

22.2.5. Добавление общего файла справки (необязательно)

Если в код каждой функции добавить комментарии `roxygen2`, то в процессе сборки (раздел 22.3) автоматически будут сгенерированы файлы справки. В этом случае после установки и загрузки пакета можно ввести `? content` или `help(contents)` и вызвать справку для этой функции. Но для самого пакета справка не создается автоматически. Другими словами, вызов `help(edatools)` потерпит неудачу.

Как пользователю узнать, какие функции доступны в пакете? Один из способов – ввести `help(package="edatools")`, но вы можете

упростить им задачу, добавив в документацию еще один файл. Например, давайте добавим файл `edatools.R` (листинг 22.6) в папку `R`.

Листинг 22.6. Содержимое файла edatools.R

```
#' @title Функции для исследования содержимого набора данных.
#'
#' @details
#' edatools предоставляет инструменты для исследования переменных в
#' наборе данных.
#'
#' @docType package
#' @name edatools-package
#' @aliases edatools
NULL
... этот файл должен завершаться дополнительной пустой строкой после NULL...
```

Обратите внимание, что этот файл должен завершаться пустой строкой. После сборки пакета вызов `help(edatools)` теперь создаст описание пакета с гиперссылкой на список функций.

22.2.6. Добавление демонстрационных данных в пакет (необязательно)

При создании пакета рекомендуется включать один или несколько наборов данных, которые можно использовать для демонстрации работы его функций. Для пакета `edatools` мы добавим набор данных `happiness`. Он содержит шесть типов переменных и различное количество пропущенных данных в каждой переменной.

Чтобы добавить набор данных в пакет, его сначала нужно поместить в память. Следующий код загружает набор данных `happiness` в глобальное окружение:

```
load(url("http://rkabacoff.com/RiA/happiness.rdata"))
```

Следующие инструкции:

```
library(usethis)
use_data(happiness)
```

создадут папку с именем `data` (если ее нет) и поместят в нее набор данных `happiness` в виде сжатого файла `.rda` с именем `happy.rda`.

Также необходимо создать файл `.R` с описанием набора данных, как показано в листинге 22.7.

Листинг 22.7. Содержимое файла happiness.R

```
#' @title Набор данных happiness
#'
#' @description
#' Набор данных с результатами опроса об удовлетворенности жизнью
#' и демографическими данными.
```

```
#' Это фиктивный набор данных.
#'
#' @source
#' Данные сгенерированы случайным образом с использованием функций из пакета
#' \href{https://cran.r-project.org/web/packages/wakefield/index.html}{wakefield}.
#'
#' @format Набор данных с 460 наблюдениями и 11 переменными:
#' \describe{
#'   \item{\code{ID}}{текст. Уникальный идентификатор.}
#'   \item{\code{Date}}{дата. Дата опроса.}
#'  \item{\code{Sex}}{фактор. Пол, 2-уровневый фактор \code{Male}
    or \code{Female}.}
#'   \item{\code{Race}}{фактор. Раса, 8-уровневый фактор.}
#'   \item{\code{Age}}{целое число. Возраст в годах.}
#'   \item{\code{Education}}{фактор. Образование, 13-уровневый фактор.}
#'   \item{\code{Income}}{действительное число. Годовой доход в долларах США.}
#'   \item{\code{IQ}}{double. Уровень интеллекта. Эта переменная имеет
#'   большой процент пропущенных данных.}
#'   \item{\code{Zip}}{текст. Почтовый индекс.}
#'   \item{\code{Children}}{целое число. Количество детей.}
#'   \item{\code{Happy}}{фактор. Оценка степени согласия с утверждением
#'   "Я счастлив большую часть времени", 6-уровневый фактор
#'   \code{Strongly Disagree}, \code{Disagree}, \code{Neutral},
#'   \code{Agree} и \code{Strongly Agree}.}
#' }
"happiness"
```

Обратите внимание, что код в листинге 22.7 полностью состоит из комментариев. Поместите его в папку `R` с файлами функций.

22.2.7. Добавление виньетки (необязательно)

Люди с бóльшей охотой будут использовать ваш пакет, если включить в него краткую статью с описанием функций и способов использования. Чтобы создать такую виньетку, выполните следующие инструкции:

```
library(usethis)
use_vignette("edatools", "Introduction to edatools")
```

Они создадут папку с именем `vignettes` (если она не существует), поместят в нее файл шаблона RMarkdown с именем `edatools.Rmd` и откроют его для редактирования в RStudio. Порядок редактирования файлов RMarkdown описан в разделе 21.2. В листинге 22.8 показана завершенная виньетка.

Листинг 22.8. Виньетка для пакета edatools

```
---
title: "Введение в edatools"
output: rmarkdown::html_vignette
vignette: >
```

```
  %\VignetteIndexEntry{Введение в edatools}
  %\VignetteEngine{knitr::rmarkdown}
  %\VignetteEncoding{UTF-8}
---

```{r, include = FALSE}
knitr::opts_chunk$set(
  collapse = TRUE,
  comment = "#>"
)
```

Пакет `edatools` -- это демонстрационный проект для изучения принципов создания пакетов для R. Обсуждается в главе 22 книги [R в действии (3-е изд.)](https://www.manning.com/books/r-in-action-third-edition).

Пакет имеет одну основную функцию, генерирующую описание набора данных, и две обобщенные функции.

```{r example}
library(edatools)
df_info <- contents(happiness)
print(df_info, digits=3)
plot(df_info)
```
```

После установки пакета пользователи смогут просмотреть виньетку вызовом `vignette("edatools")`.

22.2.8. Редактирование файла DESCRIPTION

В каждом пакете есть файл `DESCRIPTION` с метаданными, такими как название пакета, номер версии, автор(ы), лицензия и зависимости. Файл создается автоматически во время выполнения `build_package("edatools")` (раздел 22.2.2), а для его заполнения можно использовать функции из пакета `usethis`.

Для начала нужно указать, какие дополнительные пакеты необходимы вашему пакету для работы. Наш пакет `edatools` зависит от пакета `ggplot2`. Инструкции

```
library(usethis)
use_package("ggplot2")
```

добавят это требование в файл `DESCRIPTION`. Функцию `use_package()` можно вызвать несколько раз, если пакет имеет несколько зависимостей. При этом нет необходимости указывать какие-либо пакеты, распространяемые в составе дистрибутива R. После установки `edtools` автоматически будут установлены все необходимые дополнительные пакеты, если они отсутствуют.

Далее указываем лицензию, под которой выпускается пакет. Программное обеспечение с открытым исходным кодом чаще всего рас-

пространяется на условиях лицензий MIT, GPL-2 и GPL-3. Их описание можно найти по адресу: http://www.r-project.org/Licenses. Мы будем использовать лицензию MIT, которая фактически говорит, что вы можете делать все, что хотите, с этим программным обеспечением, при условии сохранения уведомления об авторских правах. Выполните инструкцию

```
use_mit_license("ваше имя")
```

чтобы добавить лицензию в свой пакет.

Наконец, отредактируйте файл DESCRIPTION вручную. Это обычный текстовый файл, поэтому его можно редактировать в RStudio или в любом текстовом редакторе. В листинге 22.9 представлена окончательная версия файла DESCRIPTION для нашего примера.

Листинг 22.9. Содержимое файла DESCRIPTION

```
Package: edatools
Title: Функции для разведочного анализа данных
Version: 0.0.0.9000
Authors@R:
    person(given = "Robert",
           family = "Kabacoff",
           role = c("aut", "cre"),
           email = "rkabacoff@wesleyan.edu")
Description: Этот пакет содержит функции для
    разведочного анализа данных. Это демонстрационный
    пакет для книги "R в действии" (3-е изд.).
License: MIT + file LICENSE
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.1.1
Depends:
    R (>= 2.10)
Imports:
    ggplot2
Suggests:
    rmarkdown,
    knitr
VignetteBuilder: knitr
```

Единственные поля, которые нужно отредактировать: Title, Version, Authors@R и Description. Поле Title должно состоять из одной строки. Формат номера версии в поле Version выбирается вами. Обычно номера версий следуют соглашению: *старший_номер.младший_номер.номер_исправлений.номер_сборки*. Более подробно о выборе номеров версий рассказывается в статье https://r-pkgs.org/release.html.

В разделе Authors@R следует указать всех участников и их роли. Здесь я указал свое имя и сообщил, что являюсь полноправным ав-

тором ("auth") и ответственным за сопровождение ("cre"). Конечно, при создании своих пакетов вы не должны использовать мое имя (разве что ваш пакет окажется особенно хорош!). Более полную информацию о ролях участников можно найти на странице http://mng.bz/eMIP.

Раздел Description может занимать несколько строк, но все строки, начиная со второй, должны иметь отступы. Параметр LazyData: yes указывает, что наборы данных в пакете (в данном случае happiness) должны быть доступны сразу после загрузки пакета. Если бы в этом параметре было установлено значение "no", то пользователю пришлось бы использовать data(happiness) для доступа к набору данных.

22.2.9. Сборка и установка пакета

Наконец, пришло время собрать пакет. К этому моменту структура папок с пакетом должна выглядеть, как показано на рис. 22.3. Папки и файлы, выделенные курсивом, необязательны. (Обратите внимание, что файлы plot.R и print.R необходимы только потому, что пакет edatools включает специализированные функции вывода и построения диаграмм.)

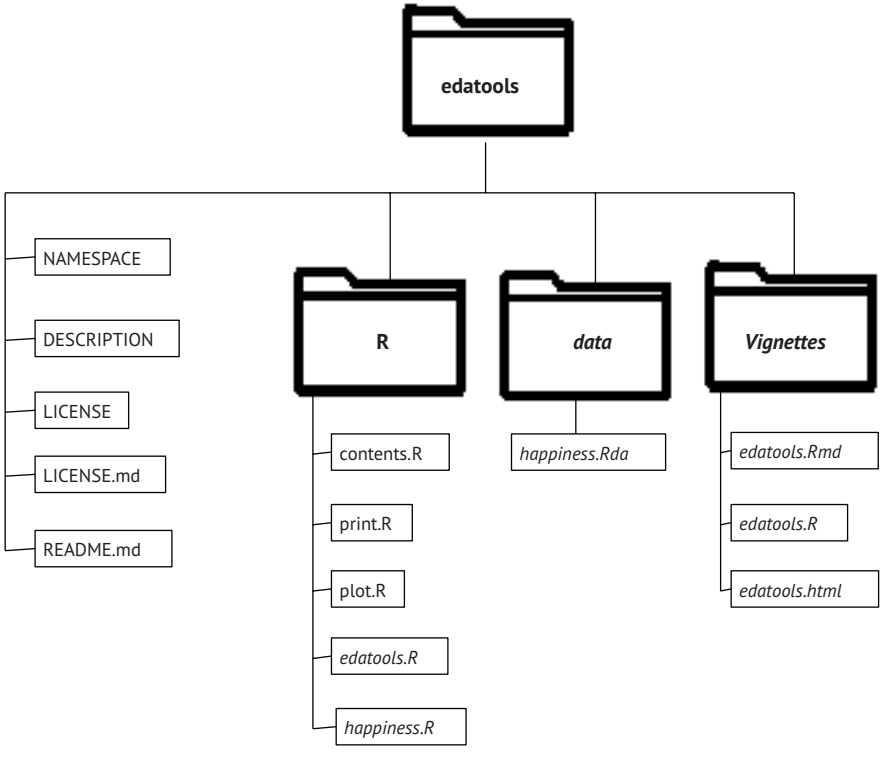


Рис. 22.3. Структура папок с пакетом edatools перед этапом сборки. Папки и файлы, выделенные курсивом, необязательны

Для сборки пакетов в RStudio предназначена вкладка Build (Сборка), как показано на рис. 22.4. Щелкните на раскрывающемся меню More (Дополнительно) и выберите пункт Configure Build Tools (Настроить инструменты сборки). После этого появится диалоговое окно, изображенное на рис. 22.5. Установите все флажки. Затем щелкните на кнопке Configure (Настроить), чтобы открыть диалог Roxygen Options (Параметры Roxygen). Установите первые шесть флажков (рис. 22.6).

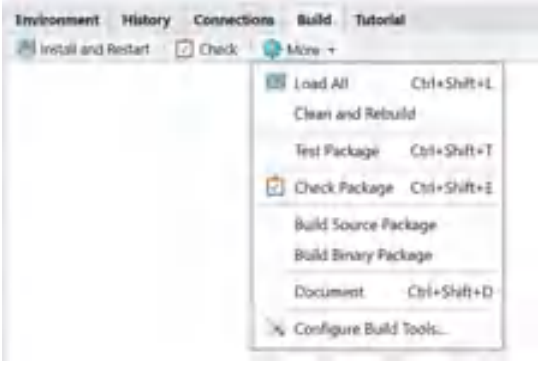


Рис. 22.4. Вкладка Build (Сборка) в RStudio. Здесь можно настроить параметры создания документации, сборки и установки готового пакета

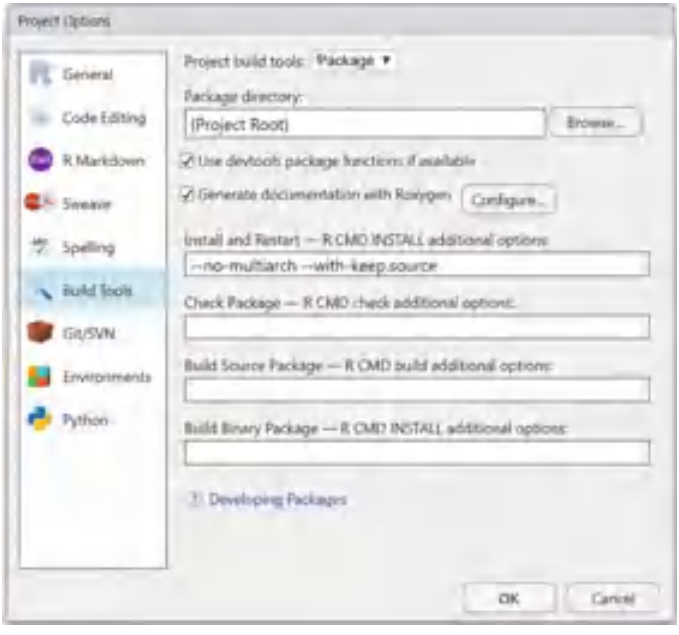


Рис. 22.5. Диалог Project Options (Параметры проекта). Установите все флажки

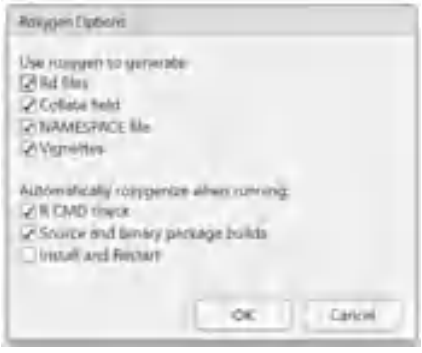


Рис. 22.6. Диалог Roxygen Options (Параметры Roxygen). Установите первые шесть флажков

Теперь осталось выполнить два простых шага. Сначала создайте документацию, выбрав в раскрывающемся меню More > Document (Дополнительно > Документация) или нажав комбинацию Ctrl+Shift+D. Это действие преобразует комментарии `roxygen2` в исходном коде на R в файлы справки LaTeX (с расширением `.Rd`). Файлы `.Rd` будут помещены в папку `man` (сокращенно от manual – руководство), а также добавится информация в файлы `DESCRIPTION` и `NAMESPACE`.

В листинге 22.10 показан получившийся в данном случае файл `NAMESPACE`, который управляет видимостью функций в пакете, т. е. доступностью функций пользователям пакета для использования напрямую. В данном случае пользователям доступны все функции. Также на этом шаге организуется доступ к функциям из сторонних пакетов, импортируемых вашим пакетом. В данном случае открывается доступ к функциям `object.size()`, `reorder()` и ко всем функциям в `ggplot2`. Дополнительные сведения о пространствах имен можно найти на странице http://adv-r.had.co.nz/Namespaces.html.

Листинг 22.10. Содержимое файла NAMESPACE

```
# Generated by roxygen2: do not edit by hand

S3method(plot,contents)
S3method(print,contents)
export(contents)
import(ggplot2)
importFrom(stats,reorder)
importFrom(utils,object.size)
```

Далее щелкните на кнопке Install and Restart (Установить и перезапустить). После этого пакет будет собран, установлен в локальную библиотеку R и загружен в текущий сеанс. На этом этапе структура папок пакета будет иметь вид, как показано на рис. 22.7. Поздравляем, ваш пакет готов к использованию!

Где моя виньетка?

Если вы добавили виньетку в пакет и следовали инструкциям в этом разделе, то не увидите ее. Но почему?! Кнопка Install and Restart (Установить и перезапустить) по умолчанию не создает виньетку. Это может занять много времени, поэтому предполагается, что разработчик не захочет делать это каждый раз при повторной сборке пакета. Виньетка будет собрана и добавлена при сборке *исходного пакета* (раздел 22.2.9). Чтобы увидеть ее, установите пакет из этого исходного пакета: вкладка Packages > Install > Install from Package Archive File (Пакеты > Установить > Установить из файла архива пакета). Также можно выполнить следующий код:

```
library(devtools)
build_vignettes()
install(build_vignettes=TRUE)
```

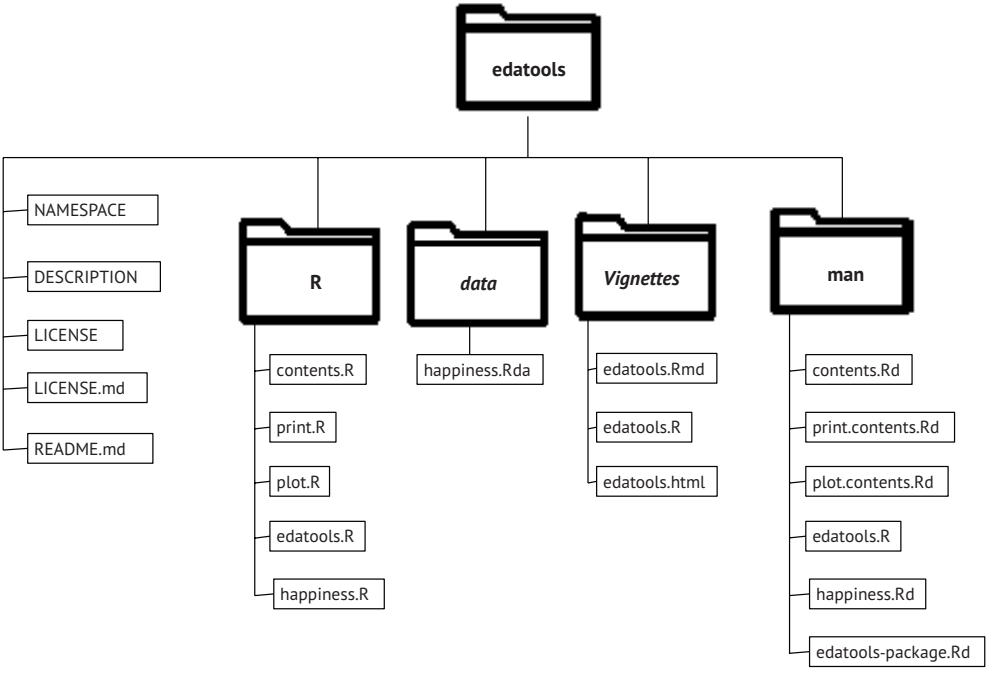


Рис. 22.7. Структура папок пакета после выбора пункта меню More > Document (Дополнительно > Документация) и щелчка на кнопке Install and Restart (Установить и перезапустить). И снова курсивом выделены необязательные файлы и папки

Прежде чем двигаться дальше, нужно проверить пакет на наличие потенциальных проблем. Если щелкнуть на кнопке Check (Проверить) на вкладке Build (Сборка), то выполнится обширная проверка согласованности пакета. Желательно всегда запускать эту проверку, прежде чем распространять свой пакет. Если обнаружатся какие-то ошибки, исправьте их, задокументируйте (если документация изменилась) и пересоберите пакет.

22.3. Распространение пакета

Создав полезный пакет, вы наверняка захотите поделиться им с другими. Существует несколько методов распространения пакетов R, в том числе:

- распространение исходного или бинарного файла пакета;
- отправка пакета в CRAN;
- размещение пакета на GitHub;
- создание веб-сайта пакета.

В этом разделе мы рассмотрим все эти варианты по очереди.

22.3.1. Распространение исходного файла пакета

Пакет можно упаковать в один сжатый файл и отправить другим по электронной почте или через облачные сервисы. На вкладке Build (Сборка) выберите пункт меню More > Build Source Package (Дополнительно > Создать исходный пакет), чтобы создать исходный файл пакета в родительском каталоге проекта. В этом случае в папке, вмещающей папку `edatools`, появится файл `edatools_0.0.0.90000.tar.gz`. Номер версии в названии будет взят из файла `DESCRIPTION`. Полученный файл можно переслать другим или отправить в CRAN.

Получатели могут установить пакет, выбрав в меню пункт Tools > Install Packages (Инструменты > Установить пакеты) и затем в списке Install from (Установить из) выбрав пункт Package Archive File (Архивный файл пакета). Также пакет можно установить с помощью функции `install.packages()`:

```
install.packages(choose.files(), repos = NULL, type="source")
```

Эта инструкция откроет диалоговое окно для выбора исходного пакета.

Если у вас установлен дистрибутив LaTeX (раздел 22.2.1), то точно так же из консоли можно создать руководство в формате PDF для пакета:

```
library(devtools)
build_manual()
```

22.3.2. Отправка в CRAN

Сеть архивов R (Comprehensive R Archive Network, CRAN) является основной службой распространения пакетов. Сегодня утром я насчитал 17 788 пакетов в ней, но это число уже безнадежно устарело. Чтобы добавить свой пакет в CRAN, выполните четыре шага:

1) прочитайте правила сети CRAN «CRAN Repository Policy» (http://cran.r-project.org/web/packages/policies.html);

2) убедитесь, что ваш пакет успешно проходит все проверки (кнопка Check (Проверить) на вкладке Build (Сборка)). Если имеются какие-либо ошибки или предупреждения, то пакет не будет принят;
3) создайте исходный файл проекта (раздел 22.3.1);
4) отправьте пакет через веб-форму http://cran.r-project.org/submit.html. В ответ по электронной почте вам придет письмо с подтверждением, которое необходимо принять.

Только, пожалуйста, не выгружайте в CRAN только что созданный пакет `edatools`. Теперь у вас есть все необходимое, чтобы создать собственный пакет.

22.3.3. Размещение на GitHub

Многие разработчики размещают свои пакеты на GitHub (http://www.github.com), даже после размещения в CRAN. GitHub – популярный сервис репозиториев Git со множеством дополнительных функций. Вот несколько веских причин для размещения пакета на GitHub:

- пакет может быть не готов к выпуску в публичное пространство (т. е. он еще не полностью разработан);
- есть желание работать над пакетом вместе с коллегами или получать отзывы и предложения от пользователей;
- сеть CRAN можно использовать для размещения рабочей версии, а GitHub – текущей разрабатываемой версии;
- есть желание использовать механизмы управления версиями в Git в процессе разработки (хотя это и не обязательно).

GitHub позволяет размещать пакеты бесплатно как частным лицам, так и организациям.

Чтобы разместить пакет на GitHub, добавьте в него файл `REAME.md` и в консоли введите команды:

```
library(usethis)
use_readme_md()
```

Они откроют файл `README.md` в редакторе. Для оформления содержимого файла используется простой язык разметки, такой как RMarkdown. Более подробную информацию об этом языке разметки можно получить, выбрав в RStudio пункт меню Help > Markdown Quick Reference (Справка > Краткий справочник по Markdown). В листинге 22.11 показано содержимое файла `README.md` для пакета `edatools`.

Листинг 22.11. Содержимое файла README.md

```
# edatools

Это демонстрационный пакет для книги [R в действии (3-е изд.)]
(https://www.manning.com/books/r-in-action-third-edition ).
Содержит функции для разведочного анализа данных.
```

```
## Установка

Установить пакет можно с помощью следующего кода:

``` r
if(!require(remotes)){
   install.packages("remotes")
}
remotes::install_github("rkabacoff/edatools")
```

Пример

Этот простой пример демонстрирует, как можно получить описание набора данных:

``` r
library(edatools)
df_info<- contents(happiness)
df_info
plot(df_info)
```
```

Сохраните файл, и теперь вы готовы разместить пакет на GitHub. Чтобы разместить пакет:

1) зарегистрируйте учетную запись и выполните вход;
2) щелкните на кнопке Create (Создать), чтобы создать новый репозиторий. На следующей странице введите имя, совпадающее с именем пакета (рис. 22.8). Оставьте настройки по умолчанию и щелкните на кнопке Create Repository (Создать репозиторий);

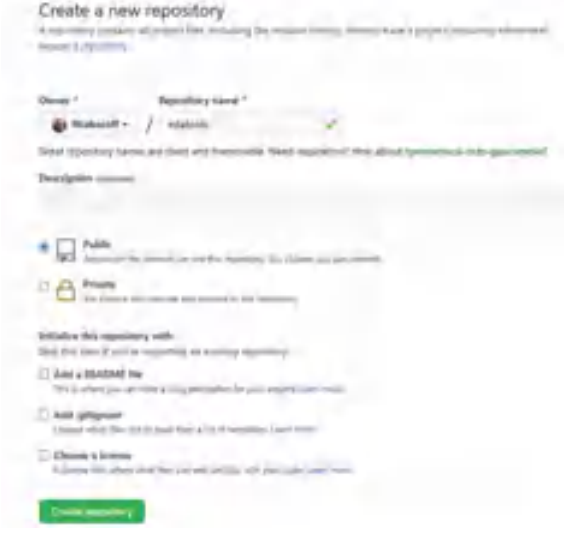


Рис. 22.8. Страница создания нового репозитория на GitHub. Введите имя пакета и щелкните на кнопке Create Repository (Создать репозиторий)

3) на следующей странице щелкните на ссылке Uploading an Existing File (Выгрузить существующий файл). Эту ссылку трудно заметить (рис. 22.9). Обратите внимание, что она предлагает выгрузить файлы пакета напрямую. Если вы используете систему управления версиями Git, то процесс выгрузки будет выглядеть иначе; он подробно описывается в статье «Happy Git and GitHub for the useR» (https://happygitwithr.com/);
4) на следующей странице выгрузите содержимое папки с пакетом (файлы и папки в пакете, а не саму папку пакета). Щелкните на кнопке Commit Changes (Принять изменения) внизу страницы. Не забудьте этот последний шаг, иначе файлы не появятся в репозитории;
5) передайте другим URL своего репозитория (https://github.com/имя_вашей_учетной_записи/имя_пакета).

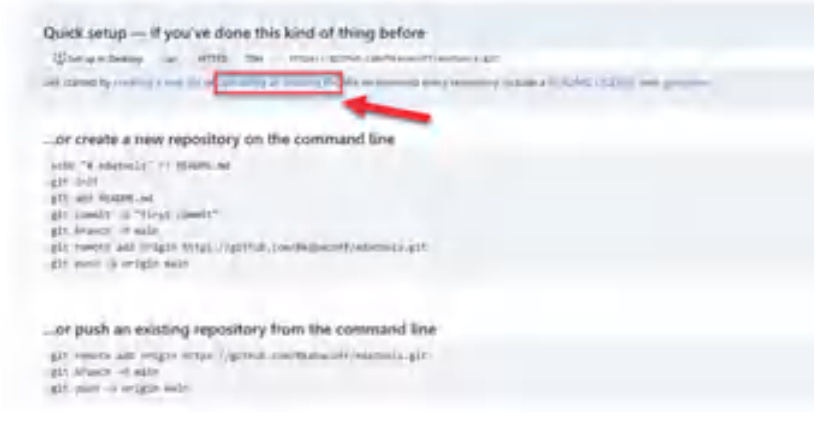


Рис. 22.9. Страница быстрой настройки репозитория на GitHub. Щелкните на ссылке Uploading an Existing File (Выгрузить существующий файл)

22.3.4. Создание веб-сайта пакета

Специальный веб-сайт может стать отличным способом продвижения вашего пакета. Создать такой сайт можно с помощью пакета `pkgdown` (https://pkgdown.r-lib.org/). В консоли перейдите в каталог проекта пакета и введите:

```
library(pkgdown)
build_site()
```

Эти команды добавят в проект папку с именем `docs`, содержащую страницу HTML, каскадные таблицы стилей и файлы JavaScript, необходимые для веб-сайта. Просто поместите эти файлы на веб-сервер, чтобы сделать сайт вашего пакета доступным.

GitHub Pages предоставляет бесплатную возможность создания веб-сайтов для проектов. Просто выгрузите папку `docs` в репозито-

рий пакета на GitHub (раздел 22.3.3). Перейдите в настройки репозитория и щелкните на ссылке GitHub Pages. В качестве источника выберите `main`, `/docs` и щелкните на кнопке Save (Сохранить), как показано на рис. 22.10. После этого ваш веб-сайт будет доступен по адресу https://имя_вашей_учетной_записи.github.io/имя_пакета.

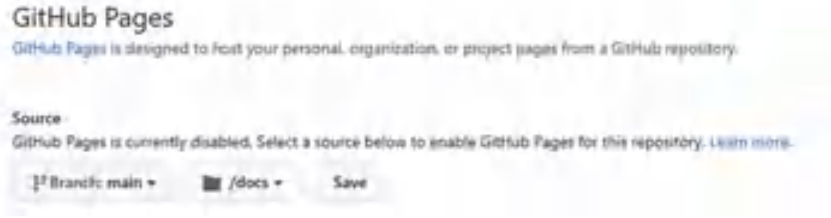


Рис. 22.10. Настройка веб-страницы для GitHub Pages

Веб-сайт пакета `edatools` доступен по адресу: https://rkabacoff.github.io/edatools (рис. 22.11).

Рис. 22.11. Веб-сайт пакета edatools на GitHub Pages

Обратите внимание, что файл `REAME.md` содержит ссылку на домашнюю страницу. Виньетка `edatools` ссылается на вкладку Get Started (Начало работы). Все функции описаны на вкладке Reference (Справка).

Сайт легко настроить перед его созданием, отредактировав файл `_pkgdown.yml` в папке `docs`. Подробности ищите в статье «Customise Your Site» (http://mng.bz/pJw2).

22.4. Дополнительная информация

Весь код, что использовался в этой главе для создания пакета `edtools`, написан на R. На самом деле большинство пакетов содержат код, написанный исключительно на R. Но R позволяет также вызывать скомпилированный код на C, C++, Fortran и Java. Внешний код обычно используется для повышения скорости выполнения или когда автор хочет использовать существующие библиотеки из своего кода на R.

Включить в пакет скомпилированный внешний код можно несколькими способами. Для этого широко используются базовые функции R: `.C()`, `.Fortran()`, `.External()` и `.Call()`. Чтобы упростить эту задачу, было также написано несколько пакетов, в том числе `inline` (C, C++, Fortran), `Rcpp` (C++) и `rJava` (Java). Обсуждение вопросов добавления внешнего скомпилированного кода в пакет на R выходит за рамки этой книги, поэтому за дополнительной информацией я советую обращаться по адресу https://r-pkgs.org/src.html.

Есть множество отличных источников информации о разработке пакетов на R. Ярким примером может служить руководство «Writing R Extensions», написанное разработчиками R (https://cran.r-project.org/doc/manuals/r-release/R-exts.html). Книга «R Packages» Хэдли Уикхема (Hadley Wickham) и Дженни Брайан (Jenny Bryan) содержит огромное количество информации и свободно доступна в интернете (https://r-pkgs.org/). Козима Мейер (Cosima Meyer) и Деннис Хаммершмидт (Dennis Hammershmidt) предлагают исчерпывающую статью в своем блоге «How to Write Your Own R Package and Publish it on CRAN» (http://mng.bz/O14o).

Пакеты R – отличный способ упорядочить часто используемые функции, разработать приложение и поделиться своими результатами с другими. В нашу эпоху воспроизводимых исследований пакеты также могут стать отличным способом объединения данных и кода для обширного проекта. В этой главе мы создали полноценный пакет на R от начала до конца. Поначалу процесс создания пакета кажется сложным, но, пройдя его один раз, вы убедитесь, что он довольно прост. А теперь приступайте к работе! И получите удовольствие!

Итоги

- Пакет R – это набор функций, документации и данных, сохраненных в стандартном формате.
- Пакеты R упрощают доступ к часто используемым функциям, помогают решать сложные аналитические задачи, обмениваться кодом и данными с другими и приносить пользу сообществу R.
- Разработка пакета – это многоэтапный процесс. Такие пакеты, как `devtools`, `usethis` и `roxygen2`, могут упростить его.

- Шаги включают: создание проекта пакета, написание функций и документации, сборку и установку пакета, а также распространение пакета.
- Пакетами можно делиться с другими, распространяя исходные или двоичные файлы, размещая пакеты в CRAN и/или на GitHub.
- Вы можете использовать пакет `pkgdown` и GitHub Pages, чтобы создать веб-сайт пакета. Он также может стать частью вашего портфолио как специалиста по анализу данных.

23 Продвинутая графика с использованием пакета lattice

В этой главе:

- введение в пакет `lattice`;
- группирующие и условные переменные;
- добавление информации с функциями настройки ячеек;
- настройка внешнего вида диаграмм, созданных с помощью `lattice`.

В этой книге мы создали множество самых разных диаграмм, используя функции из пакета `graphics`, входящего в стандартную библиотеку R, и специализированные функции из пакетов, написанных автором. В главе 19 мы познакомились с новым синтаксисом создания диаграмм с помощью функций из пакета `ggplot2`. Пакет `ggplot2` предлагает альтернативу стандартным графическим средствам R и особенно удобен для создания сложных диаграмм.

В этой дополнительной главе мы рассмотрим пакет `lattice`, написанный Дипаяном Саркаром (Deepayan Sarkar; 2008). Он реализует категоризованные, или решетчатые, диаграммы, описанные

Кливлендом (Cleveland; 1985, 1993). Пакет lattice вышел за рамки первоначального подхода к визуализации данных, предложенного Кливлендом, и теперь являет собой комплексную систему создания статистических диаграмм. Как и ggplot2, пакет lattice графика имеет свой синтаксис, предлагает альтернативу стандартным графическим средствам и с успехом может использоваться для отображения сложных данных. Аналитики склонны использовать либо lattice, либо ggplot2, исходя из личных предпочтений. Попробуйте поработать с обоими пакетами и выберите тот, который вам больше понравится.

23.1. Пакет lattice

Пакет lattice – это комплексная система визуализации одномерных и многомерных данных. В частности, многие пользователи обращаются к пакету lattice, т. к. он позволяет легко генерировать решетчатые диаграммы.

Категоризованные диаграммы (называемые еще «графиками на решетке») отображают распределение переменной или взаимосвязи между переменными отдельно для каждого уровня одной или нескольких других переменных. Рассмотрим следующий вопрос: *как зависит рост певцов Нью-Йоркского хорового общества от их вокальных партий?*

Данные о росте и голосовых партиях участников хора представлены в наборе данных singer, содержащемся в пакете lattice. В следующем коде:

```
library(lattice)
histogram(~height | voice.part, data = singer,
    main="Distribution of Heights by Voice Pitch",
    xlab="Height (inches)")
```

height – это зависимая переменная, voice.part называется *условной переменной* (conditioning variable) и гистограммы строятся отдельно для каждой из восьми вокальных партий. Полученная диаграмма представлена на рис. 23.1. Похоже, теноры и басы в среднем имеют более высокий рост, чем альты и сопрано.

На категоризованных диаграммах для каждого значения условной переменной создается своя *ячейка* (*панель*). Если используется более одной условной переменной, то отдельная ячейка создается для каждой комбинации значений факторов. Чтобы упростить сравнение, ячейки объединяются в матрицу. Подпись в каждой ячейке находится в области, называемой *планкой* (strip). Как будет показано далее, пользователь имеет полный контроль над отображением диаграмм в каждой ячейке, форматом и расположением планок, порядком ячеек, размещением и содержимым легенды и многими другими графическими параметрами.

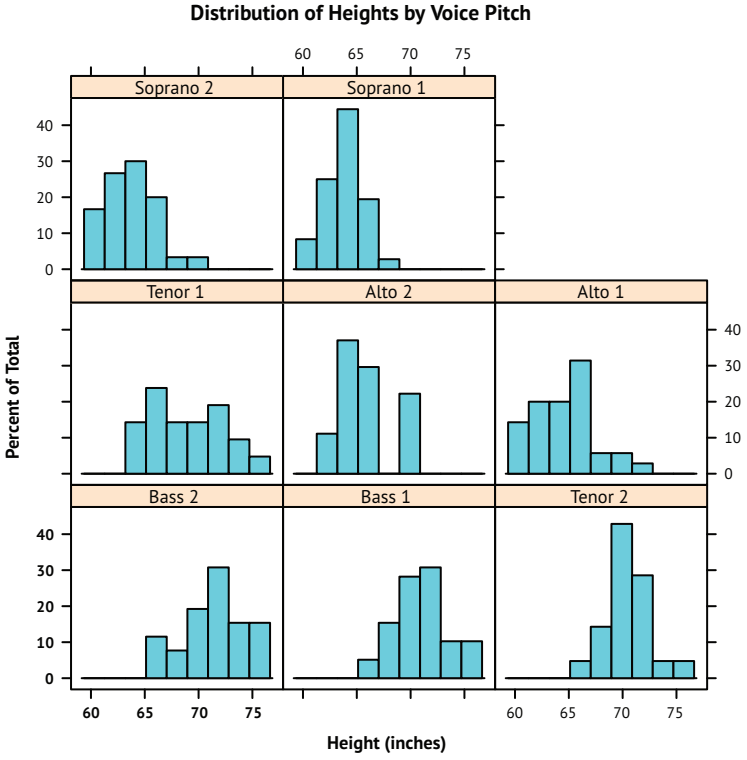


Рис. 23.1. Категоризованная диаграмма зависимости роста исполнителей от вокальных партий

Пакет `lattice` содержит много функций для создания одномерных (точечные диаграммы, диаграммы ядерной оценки функции плотности, гистограммы, столбиковые диаграммы, коробчатые диаграммы), двухмерных (диаграммы рассеяния, параллельные диаграммы рассеяния) и многомерных (трехмерные диаграммы, матрицы диаграмм рассеяния) диаграмм.

Все графические функции высокого уровня имеют следующий синтаксис:

```
graph_function(formula, data=, options)
```

где

- *`graph_function`* – одна из функций, перечисленных во втором столбце в табл. 23.1;
- *`formula`* определяет переменные, которые нужно изобразить, а также условные переменные;
- `data=` задает таблицу данных;
- *`options`* – параметры, перечисленные через запятую, которые определяют настройки содержимого, расположения и анно-

таций диаграммы. Наиболее часто используемые параметры приведены в табл. 23.2.

Пусть строчные буквы соответствуют числовым переменным, а прописные – категориальным (факторам). Тогда типичная формула в вызове высокоуровневой функции имеет вид:

```
y ~ x | A * B
```

где переменные слева от вертикальной черты называются *первичными*, а справа – *условными*. Первичные переменные задают оси в каждой ячейке. В данном случае выражение *y ~ x* описывает переменные, отложенные по вертикальной и горизонтальной осям соответственно. Для одномерных диаграмм это выражение заменяется на *~x*. Для трехмерных диаграмм оно будет выглядеть как *z~x*y*. Наконец, для многомерных диаграмм (матрица диаграмм рассеяния или диаграмма параллельных координат) выражение *y~x* заменяется на название таблицы данных. Обратите внимание, что условные переменные можно не указывать.

Следуя этой логике, *~x|A* позволяет изобразить значения числовой переменной *x* при каждом значении фактора *A*. Выражение *y~x|A*B* отображает связь между числовыми переменными *y* и *x* отдельно для каждой комбинации значений факторов *A* и *B*. Выражение *A~x* отображает значения категориальной переменной *A* по вертикальной оси и числовой переменной *x* по горизонтальной. Выражение *~x* отображает только значения числовой переменной *x*. Остальные примеры перечислены в табл. 23.1.

Чтобы быстро получить общее представление о категоризованных диаграммах, попробуйте выполнить пример в листинге 23.1. Диаграммы основаны на данных об автомобилях (пробег на одном галлоне топлива, вес, число передач, число цилиндров и т. д.) из таблицы `mtcars`. Вы можете поэкспериментировать с формулами и посмотреть, что получается в результате. (Вывод здесь не приводится для экономии места.)

Таблица 23.1. Типы диаграмм и соответствующие функции в пакете lattice

Тип диаграммы	Функция	Пример формулы
Трехмерная контурная диаграмма	`contourplot()`	`z~x*y`
Трехмерная уровневая диаграмма	`levelplot()`	*z~y*x*
Трехмерная диаграмма рассеяния	`cloud()`	*z~x*y|A*
Трехмерная каркасная диаграмма	`wireframe()`	*z~y*x*
Столбиковая диаграмма	`barchart()`	*x~A* или *A~x*
Диаграммы рассеяния	`bwplot()`	*x~A* или *A~x*

Окончание табл. 23.1

Тип диаграммы	Функция	Пример формулы
Точечная диаграмма	`dotplot()`	*~x|A*
Гистограмма	`histogram()`	*~x*
Диаграмма ядерной оценки функции плотности	`densityplot()`	*~x|A*B*
Диаграмма параллельных координат	`parallel()`	`таблица данных`
Диаграмма рассеяния	`xyplot()`	*y~x|A*
Матрица диаграмм рассеяния	`splom()`	`таблица данных`
Одномерная диаграмма рассеяния	`stripplot()`	*x~A* или *A~x*

Примечание: в приведенных формулах строчные буквы обозначают числовые переменные, а прописные – категориальные.

Листинг 23.1. Примеры категоризованных диаграмм

```
library(lattice)
attach(mtcars)

gear <- factor(gear, levels=c(3, 4, 5),
               labels=c("3 gears", "4 gears", "5 gears"))
cyl <- factor(cyl, levels=c(4, 6, 8),
              labels=c("4 cylinders", "6 cylinders", "8 cylinders"))
densityplot(~mpg,
            main="Density Plot",
           xlab="Miles per Gallon")
densityplot(~mpg | cyl,
            main="Density Plot by Number of Cylinders",
            xlab="Miles per Gallon")
bwplot(cyl ~ mpg | gear,
       main="Box Plots by Cylinders and Gears",
       xlab="Miles per Gallon", ylab="Cylinders")

xyplot(mpg ~ wt | cyl * gear,
       main="Scatter Plots by Cylinders and Gears",
       xlab="Car Weight", ylab="Miles per Gallon")
cloud(mpg ~ wt * qsec | cyl,
      main="3D Scatter Plots by Cylinders")
dotplot(cyl ~ mpg | gear,
        main="Dot Plots by Number of Gears and Cylinders",
        xlab="Miles Per Gallon")
splom(mtcars[c(1, 3, 4, 5, 6)],
      main="Scatter Plot Matrix for mtcars Data")
detach(mtcars)
```

Высокоуровневые графические функции в пакете `lattice` создают графические объекты, которые можно сохранять и видоизменять. Например, инструкции

```
library(lattice)
mygraph <- densityplot(~height|voice.part, data=singer)
```

создадут категоризованную диаграмму плотности рассеяния и сохранят ее в виде объекта `mygraph`. При этом на экран она не выводится. Ввод инструкции `plot(mygraph)` или просто `mygraph` позволит увидеть диаграмму.

Категоризованные диаграммы легко видоизменять, задавая их параметры. Наиболее часто используемые из них перечислены в табл. 23.2. Далее в этой главе вы увидите множество примеров их применения.

Таблица 23.2. Наиболее часто используемые параметры графических функций в пакете lattice

Параметр	Описание
`aspect`	Число, определяющее соотношение размеров (высота/ширина) диаграммы в каждой ячейке
`col, pch, lty, lwd`	Векторы, определяющие цвета, символы, типы и ширину линий соответственно
`groups`	Группирующая переменная (фактор)
`index.cond`	Список, определяющий порядок расположения ячеек
`key (или auto.key)`	Функция, используемая для описания значений группирующей переменной (переменных) в легенде
`layout`	Числовой вектор из двух элементов, который определяет расположение ячеек (число колонок и строк). При необходимости может быть введен третий элемент, задающий число страниц
`main, sub`	Текстовые векторы, которые определяют заголовок и подзаголовок
`panel`	Функция, используемая для создания диаграммы в каждой ячейке
`scales`	Список с информацией о масштабах осей
`strip`	Функция, используемая для определения формата планок
`split, position`	Числовые векторы, управляющие размещением нескольких диаграмм на странице
`type`	Текстовый вектор, задающий один или более параметров диаграмм рассеяния (`p` = точки, `l` = линии, `r` = регрессионная линия, `smooth` = сглаживание, `g` = сетка и т. д.)
`xlab, ylab`	Текстовые векторы, определяющие подписи горизонтальных и вертикальных осей
`xlim, ylim`	Числовые векторы из двух элементов, которые определяют наименьшее и наибольшее значения по горизонтальной и вертикальной осям соответственно

Эти параметры можно передавать в высокоуровневые функции или использовать в функциях, работающих с ячейками, которые описаны в разделе 23.3.

Для модификации объектов диаграмм, созданных при помощи пакета `lattice`, можно также использовать функцию `update()`. Продолжая пример с певцами, следующая инструкция:

```
newgraph <- update(mygraph, col="red", pch=16,
                   cex=.8, jitter=.05, lwd=2)
```

позволит нарисовать диаграмму заново, используя красные линии и символы (`color="red"`), заполненные цветом точки (`pch=16`), более мелкие (`cex=.8`) и сильнее смещенные друг относительно друга символы (`jitter=.05`), а также линии двойной толщины (`lwd=2`). Теперь, когда мы в общих чертах рассмотрели устройство высокоуровневых функций пакета, давайте подробнее поговорим об условных переменных.

23.2. Условные переменные

Как вы уже поняли, одна из наиболее впечатляющих особенностей категоризованных диаграмм – это возможность добавления условных переменных. Если есть одна условная переменная, то для каждого ее значения создается отдельная ячейка. Если есть две условные переменные, то отдельные ячейки создаются для каждого сочетания их значений. Использование более двух условных переменных редко бывает полезным.

Обычно условные переменные – это факторы. Но что, если вы решите использовать непрерывную переменную в роли условной? Одна из возможностей – преобразовать непрерывную переменную в дискретную при помощи функции `cut()`. В качестве альтернативы можно использовать функции из пакета `lattice`, которые преобразуют непрерывную переменную в объект типа *shingle* («галька»). В данном случае непрерывная переменная разделяется на ряд (возможно) перекрывающихся диапазонов значений. Например, функция

```
myshingle <- equal.count(x, number=n, overlap=proportion)
```

разделит непрерывную переменную *x* на *n* интервалов с перекрытием *proportion* и одинаковым числом наблюдений в каждом интервале и сохранит результат в объекте `myshingle` (класса `shingle`). Вывод этого объекта на экран (например, `plot(myshingle)`) позволит увидеть полученные интервалы.

После преобразования непрерывной переменной в объект класса `shingle` ее можно использовать в качестве условной переменной. Например, давайте исследуем связь между пробегом на одном галлоне топлива и весом машины у автомобилей с разным объемом двигателя, охарактеризованных в наборе данных `mtcars`. Поскольку объем двигателя – это непрерывная переменная, сначала преобразуем ее в переменную с тремя уровнями:

```
displacement <- equal.count(mtcars$disp, number=3, overlap=0)
```

Затем передадим полученную переменную в функцию xyplot():

```
xyplot(mpg~wt|displacement, data=mtcars,
  main = "Miles per Gallon vs. Weight by Engine Displacement",
  xlab = "Weight", ylab = "Miles per Gallon",
  layout=c(3, 1), aspect=1.5)
```

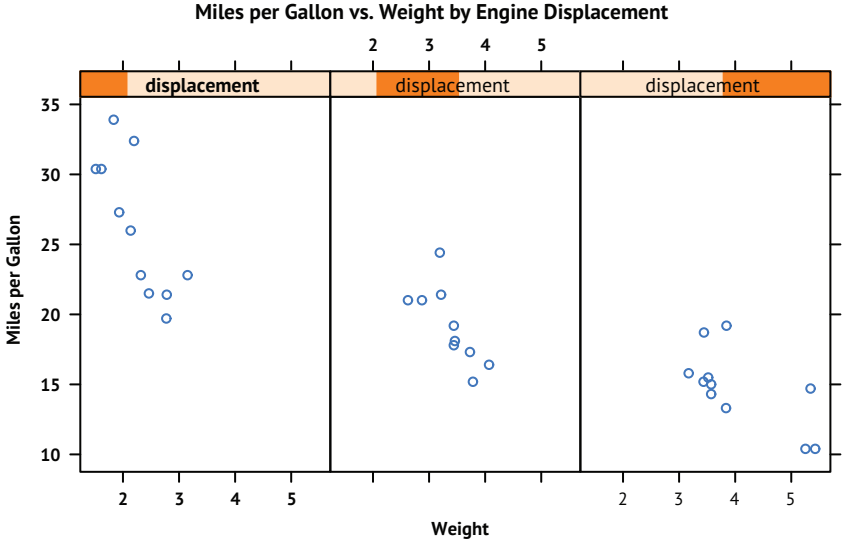


Рис. 23.2. Категоризованная диаграмма, отражающая зависимость пробега на одном галлоне топлива от веса машины для разных объемов двигателей (displacement). Поскольку объем двигателя – это непрерывная переменная, она была разделена на три неперекрывающихся интервала с одинаковым числом значений в каждом

Результаты показаны на рис. 23.2. Обратите внимание, что здесь также использованы параметры для изменения расположения ячеек (три столбца и одна строка) и соотношения их размеров (отношение высоты к ширине), чтобы упростить сравнение полученных трех групп данных.

Как видите, подписи на планках ячеек, приведенных на рис. 23.1 и рис. 23.2, различаются. Вид планок на рис. 23.2 указывает на непрерывность исходной условной переменной, диапазон ее значений в данной ячейке отмечен более темным цветом. В следующем разделе мы будем использовать специальные функции для изменения формата ячеек.

23.3. Функции для изменения формата ячеек

Каждая высокоуровневая функция в табл. 23.1 по умолчанию использует для создания ячеек определенную функцию. Эти функции имеют имена вида panel.*graph_function*, где *graph_function* – это высокоуровневая функция. Например, функцию

```
xyplot(mpg~wt|displacement, data=mtcars)
```

можно также вызвать таким образом:

```
xyplot(mpg~wt|displacement, data=mtcars, panel=panel.xyplot)
```

Это весьма заманчивая возможность, поскольку позволяет заменить функцию, по умолчанию контролирующую формат ячеек, вашей собственной. При создании этой функции можно использовать одну или несколько из более чем 50 функций в пакете `lattice`, работающих с ячейками. Настройка функций для работы с ячейками предоставляет большие возможности для создания итоговой диаграммы, которая отвечает вашим потребностям. Рассмотрим некоторые примеры.

В предыдущем разделе мы рассмотрели зависимость пробега на одном галлоне топлива от веса машины при разных объемах двигателя. А что, если вам понадобится добавить в эту диаграмму регрессионные линии, графики-щетки и координатную сетку? Это можно сделать, создав свою функцию для управления форматом ячеек (листинг 23.2). Полученная диаграмма представлена на рис. 23.3.

Листинг 23.2. Изменение формата ячеек для функции xyplot

```
library(lattice)
displacement <- equal.count(mtcars$disp, number=3, overlap=0)

mypanel <- function(x, y) {
             panel.xyplot(x, y, pch=19)
             panel.rug(x, y)
             panel.grid(h=-1, v=-1)
             panel.lmline(x, y, col="red", lwd=1, lty=2)
           }
xyplot(mpg~wt|displacement, data=mtcars,
       layout=c(3, 1),
       aspect=1.5,
       main = "Miles per Gallon vs. Weight by Engine Displacement",
       xlab = "Weight",
       ylab = "Miles per Gallon",
       panel = mypanel)      (1)
```

В данном случае мы объединили четыре функции в одной новой функции `mypanel()` и вызвали ее внутри функции `xyplot()` при помощи параметра `panel=` (1). Функция `panel.xyplot()` создает диаграмму рассеяния, на которой изображены залитые кружки (`pch=19`). Функция `panel.rug()` добавляет графики-щетки к осям `x` и `y` каждой ячейки. Функции `panel.rug(x, FALSE)` или `panel.rug(FALSE, y)` добавили бы графики-щетки только к горизонтальной или вертикальной оси соответственно. Функция `panel.grid()` добавляет горизонтальные и вертикальные линии координатной сетки (использование отрицательных чисел позволяет расположить эти линии напротив делений). Наконец, функция `panel.lmline()` добавляет регрессионную красную (`col="red"`) пунктирную (`lty=2`) линию

стандартной толщины (lwd=1). Каждая функция, которая применяется по умолчанию для определения вида ячеек, имеет свою структуру и параметры. За дополнительной информацией обращайтесь к справке (например, help(panel.abline)).

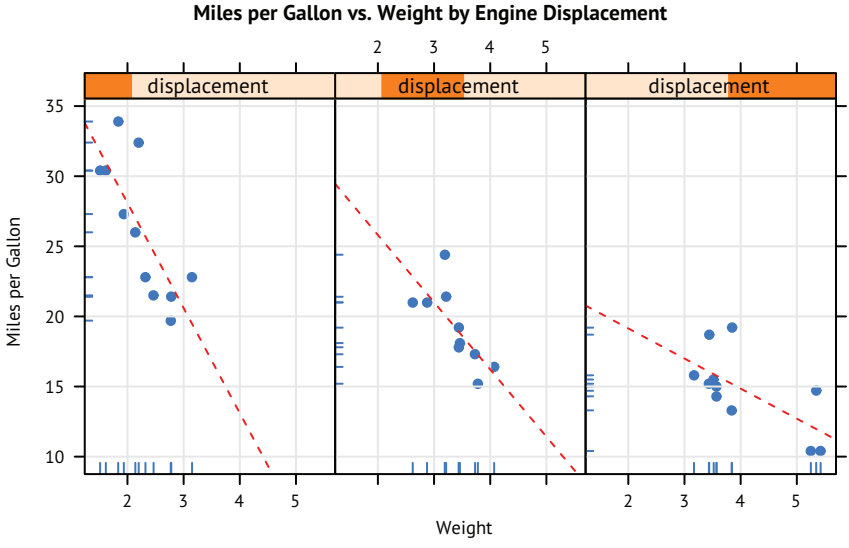


Рис. 23.3. Категоризованная диаграмма зависимости пробега на одном галлоне топлива от веса машины для разного объема двигателя. Для добавления регрессионных линий, графиков-щеток и координатной сетки была использована специально созданная функция изменения параметров ячеек

В качестве второго примера изобразим связь между пробегом на одном галлоне топлива и объемом двигателя (в формате непрерывной переменной) для разных типов коробок передач. В дополнение к отдельному изображению этой связи для автоматической и ручной коробки передач мы добавим аппроксимирующие линии и горизонтальные линии, соответствующие средним значениям (листинг 23.3).

Листинг 23.3. Изменение формата ячеек и дополнительные параметры функции xyplot

```
library(lattice)
mtcars$transmission <- factor(mtcars$am, levels=c(0,1),
                              labels=c("Automatic", "Manual"))

panel.smoother <- function(x, y) {
                    panel.grid(h=-1, v=-1)
                    panel.xyplot(x, y)
                    panel.loess(x, y)
                   panel.abline(h=mean(y), lwd=2, lty=2, col="darkgreen")
                  }
```

```
xyplot(mpg~disp|transmission,data=mtcars,
       scales=list(cex=.8, col="red"),
       panel=panel.smoother,
       xlab="Displacement", ylab="Miles per Gallon",
       main="MPG vs Displacement by Transmission Type",
       sub = "Dotted lines are Group Means", aspect=1)
```

Получившаяся диаграмма показана на рис. 23.4.

В этом новом коде стоит обратить внимание на несколько вещей. Функция `panel.xyplot()` изображает отдельные точки, а функция `panel.loess()` накладывает на точки аппроксимирующие линии. Функция `panel.abline()` добавляет на диаграмму горизонтальные линии, соответствующие средним пробегам на одном галлоне топлива в каждой группе. (Если заменить выражение `h=mean(y)` на `h=mean(mtcars$mpg)`, то будет нарисована одна линия, соответствующая среднему значению в целой выборке.) Параметр `scales=` позволяет сделать деления осей красными и уменьшить их до 80 % от размера по умолчанию.

В предыдущем примере мы могли бы использовать выражение `scales=list(x=list(), y=list())`, чтобы отдельно указать параметры горизонтальной и вертикальной осей. Дополнительную информацию о доступных для изменения параметрах осей ищите в справке `help(xyplot)`. В следующем разделе вы узнаете, как представлять данные для разных групп объектов на одной диаграмме вместо изображения их в отдельных ячейках.

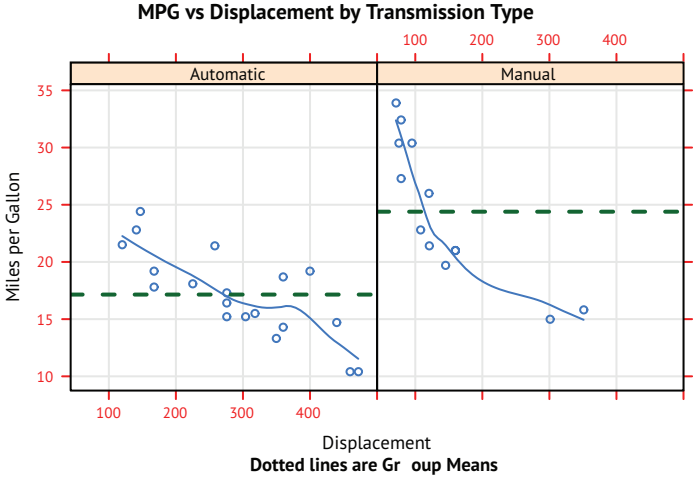


Рис. 23.4. Категоризованная диаграмма для зависимости пробега на одном галлоне топлива от объема двигателя при разных типах коробки передач. Добавлены аппроксимирующие линии, координатные сетки и линии, соответствующие среднему значению в группе

23.4. Группировка переменных

При включении условной переменной в формулу графической функции из пакета `lattice` для каждого значения этой переменной создается отдельная ячейка. Чтобы вместо этого наложить результаты для разных групп данных друг на друга, можно обозначить эту переменную как группирующую.

Допустим, мы решили отобразить распределение значений пробега на одном галлоне топлива для автомобилей с ручной и автоматической коробками передач на диаграмме ядерной оценки функции плотности. Эти диаграммы можно наложить при помощи следующих инструкций:

```
library(lattice)
mtcars$transmission <- factor(mtcars$am, levels=c(0, 1),
                              labels=c("Automatic", "Manual"))
densityplot(~mpg, data=mtcars,
            group=transmission,
            main="MPG Distribution by Transmission Type",
            xlab="Miles per Gallon",
            auto.key=TRUE)
```

Полученная диаграмма показана на рис. 23.5. По умолчанию параметр `group=` позволяет наложить друг на друга диаграммы для всех уровней группирующей переменной. Наблюдения обозначаются незалитыми кружками, линии рисуются сплошными, а разным значениям группирующей переменной соответствуют разные цвета. В черно-белом исполнении цвета трудно различать. Позже вы узнаете, как изменить эти настройки по умолчанию.

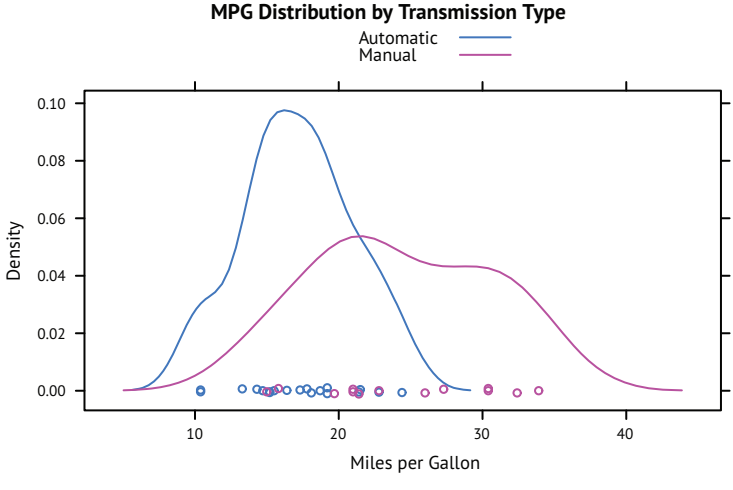


Рис. 23.5. Диаграммы ядерной оценки функции плотности значений пробега на одном галлоне топлива (Miles per Gallon) для разных типов коробок передач. Смещенные относительно друг друга значения переменной изображены на горизонтальной оси

Обратите внимание, что легенда не создается автоматически. Параметр `auto.key=TRUE` позволяет поместить примитивную легенду над диаграммой. Эту легенду можно немного изменить, определив список параметров. Например, инструкция

```
auto.key=list(space="right", columns=1, title="Transmission")
```

поместит легенду справа от диаграммы, перечислит значения группирующих переменных в одном столбце и добавит заголовок.

Если вам необходим больший контроль над изображением легенды, используйте параметр `key=`. Пример показан в листинге 23.4, а получившаяся диаграмма – на рис. 23.6.

Листинг 23.4. Диаграмма ядерной оценки функции плотности с группирующей переменной и улучшенной легендой

```
library(lattice)
mtcars$transmission <- factor(mtcars$am, levels=c(0, 1),
                              labels=c("Automatic", "Manual"))

colors <- c("red", "blue")                                    1
lines <- c(1,2)                                               1
points <- c(16,17)                                            1

key.trans <- list(title="Transmission",                       2
                  space="bottom", columns=2,                  2
                  text=list(levels(mtcars$transmission)),     2
                  points=list(pch=points, col=colors),        2
                  lines=list(col=colors, lty=lines),          2
                  cex.title=1, cex=.9)                        2

densityplot(~mpg, data=mtcars,                                3
            group=transmission,                               3
            main="MPG Distribution by Transmission Type",     3
            xlab="Miles per Gallon",                          3
            pch=points, lty=lines, col=colors,                3
            lwd=2, jitter=.005,                               3
            key=key.trans)                                    3
```

В данном случае типы значков и линий, а также цвета заданы в виде векторов 1. Первый элемент каждого вектора соответствует первому значению группирующей переменной, второй элемент – второму значению и т. д. Параметры легенды указаны в виде списка 2. Эти параметры позволяют расположить легенду внизу диаграммы в виде двух столбцов, включающих расшифровку значений переменной, типы значков и линий, а также цвета. Заголовок легенды напечатан немного крупнее, чем расшифровка значений переменной.

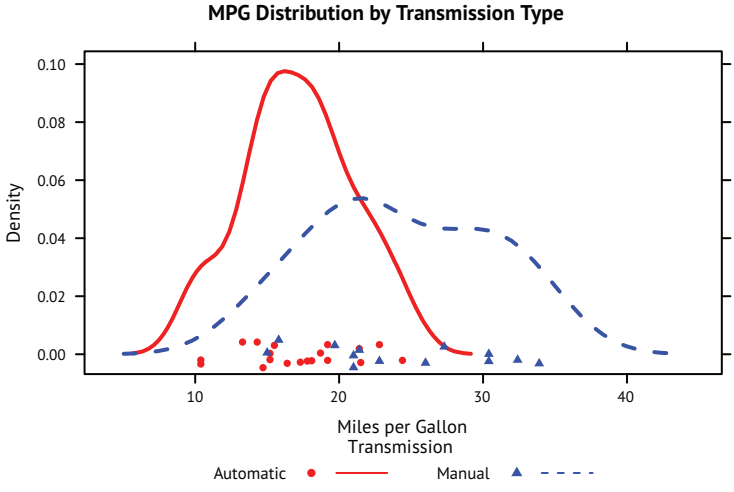


Рис. 23.6. Диаграммы ядерной оценки функции плотности значений пробега на одном галлоне топлива для разных типов коробок передач. Графические параметры были видоизменены, добавлена улучшенная легенда, в которой указаны цвет и форма символов, тип линий, размер текста и заголовок

Те же самые типы символов и линий, а также цвета заданы как параметры функции `densityplot()` 3. Вдобавок для улучшения вида диаграммы увеличены ширина линий и степень смещения символов друг относительно друга. В завершение указано, что легенда должна быть основана на ранее созданном списке. Такой подход к созданию условных обозначений для группирующей переменной открывает большие возможности. В действительности можно создать несколько легенд и разместить их в разных областях диаграммы (здесь не показано).

Прежде чем завершить этот раздел, рассмотрим пример одновременного применения группирующей и условной переменных. Набор данных `CO2`, входящий в состав дистрибутива R, описывает исследование холодоустойчивости злака ежовника обыкновенного (*Echinocholoa crus-galli*).

В наборе данных содержится информация об интенсивности поглощения углекислого газа (переменная `uptake`) у 12 растений (`Plant`) при семи уровнях концентрации двуокиси углерода в окружающей среде (`conc`). Шесть растений были из Квебека, а шесть – из Миссисипи. По три растения из каждого географического района содержались в условиях пониженной температуры. В данном примере `Plant` – это группирующая переменная, а `Type` (Квебек или Миссисипи) и `Treatment` (с пониженной/нормальной температурой) – условные. Код в листинге 23.5 создает диаграмму, изображенную на рис. 23.7.

Листинг 23.5. Применение функции xyplot с группирующей и условными переменными и улучшенной легендой

```
library(lattice)
colors <- "darkgreen"
symbols <- c(1:12)
linetype <- c(1:3)

key.species <- list(title="Plant",
                    space="right",
                    text=list(levels(CO2$Plant)),
                    points=list(pch=symbols, col=colors))

xyplot(uptake~conc|Type*Treatment, data=CO2,
       group=Plant,
       type="o",
       pch=symbols, col=colors, lty=linetype,
       main="Carbon Dioxide Uptake\nin Grass Plants",
       ylab=expression(paste("Uptake ",
              bgroup("(", italic(frac("umol","m"^2)), ")"))),
       xlab=expression(paste("Concentration ",
              bgroup("(", italic(frac(mL,L)), ")"))),
       sub = "Grass Species: Echinochloa crus-galli",
       key=key.species)
```

Обратите внимание на использование символа `\n` для создания двухстрочного заголовка и функции `expression()` для добавления математических обозначений в подписи осей. В данном случае обозначение разных значений группирующей переменной разными цветами не применяется – в параметре `col=` указан один цвет. Здесь наличие 12 разных цветов было бы избыточным и не позволило бы достичь наглядности изображения связей между переменными в каждой ячейке. Хорошо видно, что охлажденные злаки из Миссисипи отличаются от остальных.

До этого момента мы изменяли графические элементы при помощи параметров или высокоуровневых функций, например `xyplot(pch=17)`, или используемых ими функций для изменения характеристик ячеек, например `panel.xyplot(pch=17)`. Однако такие изменения действуют только во время выполнения данной функции. В следующем разделе мы познакомимся со способами изменения графических параметров в течение всего интерактивного сеанса или при запуске программы в пакетном режиме.

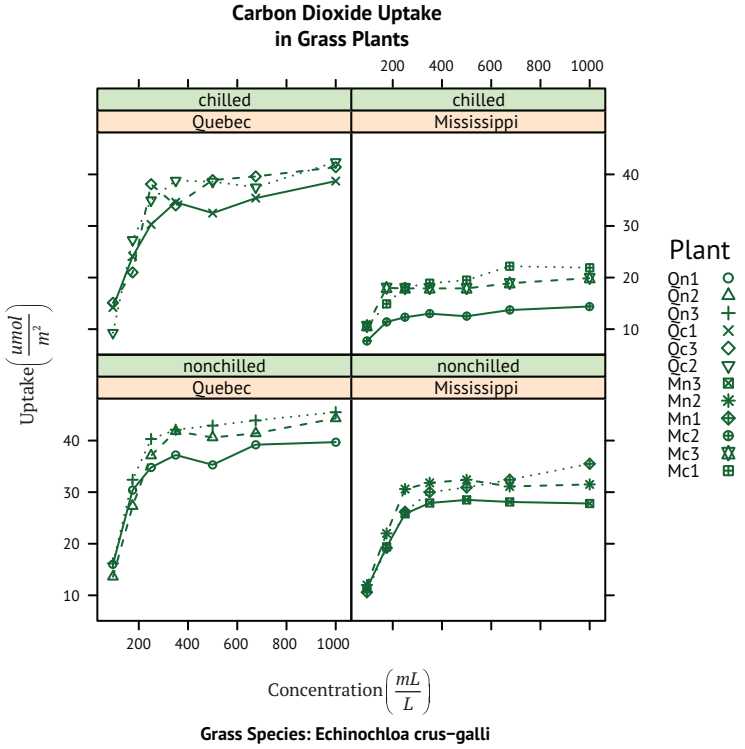


Рис. 23.7. Графики, демонстрирующие влияние концентрации углекислого газа в окружающей среде на его потребление 12 растениями при разных экспериментальных условиях и в разных регионах. Plant – это группирующая переменная, а Type и Treatment – условные

23.5. Графические параметры

В главе 3 вы узнали, как просматривать и устанавливать графические параметры при помощи функции `par()`. Этот прием прекрасно подходит для диаграмм, созданных в базовой графической системе R, но на пакет `lattice` такой подход не действует. Графические параметры, используемые в этом пакете по умолчанию, можно получить при помощи функции `trellis.par.get()` и изменить вызовом функции `trellis.par.set()`. Для визуального представления текущих графических параметров можно воспользоваться функцией `show.settings()`.

Давайте для примера изменим тип символов, которые применяются при наложении нескольких групп данных (то есть точек на диаграмме с группирующей переменной). По умолчанию кружки изображаются незалитыми. Вместо этого мы назначим каждой группе свой символ.

Сначала выведем на экран текущие настройки по умолчанию

```
show.settings()
```

и сохраним их в списке с именем `mysettings`:

```
mysettings <- trellis.par.get()
```

Вывести компоненты этого списка можно с помощью функции `names()`:

```
> names(mysettings)
 [1] "grid.pars"          "fontsize"           "background"
 [4] "panel.background"   "clip"               "add.line"
 [7] "add.text"           "plot.polygon"       "box.dot"
[10] "box.rectangle"      "box.umbrella"       "dot.line"
[13] "dot.symbol"         "plot.line"          "plot.symbol"
[16] "reference.line"     "strip.background"   "strip.shingle"
[19] "strip.border"       "superpose.line"     "superpose.symbol"
[22] "superpose.polygon"  "regions"            "shade.colors"
[25] "axis.line"          "axis.text"          "axis.components"
[28] "layout.heights"     "layout.widths"      "box.3d"
[31] "par.xlab.text"      "par.ylab.text"      "par.zlab.text"
[34] "par.main.text"      "par.sub.text"
```

Значения по умолчанию, определяющие наложенные символы, находятся в компоненте `superpose.symbol`:

```
> mysettings$superpose.symbol

$alpha
Customizing plot strips 15
[1] 1 1 1 1 1 1 1
$cex
[1] 0.8 0.8 0.8 0.8 0.8 0.8 0.8
$col
[1] "#0080ff" "#ff00ff" "darkgreen" "#ff0000" "orange"
[6] "#00ff00" "brown"
$fill
[1] "#CCFFFF" "#FFCCFF" "#CCFFCC" "#FFE5CC" "#CCE6FF" "#FFFFCC"
[7] "#FFCCCC"
$font
[1] 1 1 1 1 1 1 1
$pch
[1] 1 1 1 1 1 1 1
```

Как видите, для каждого уровня группирующей переменной используется символ в виде незалитого кружка (`pch=1`). Заданы семь уровней, после чего типы символов используются повторно.

Чтобы изменить настройки по умолчанию, выполним следующие инструкции:

```
mysettings$superpose.symbol$pch <- c(1:10)
trellis.par.set(mysettings)
```

Увидеть внесенные изменения можно повторным вызовом функции `show.settings()`. Теперь в диаграммах символ 1 (неза-

литый кружок) будет использоваться для первого значения группирующей переменной, символ 2 (незалитый треугольник) – для второго и т. д. Кроме того, разные символы определены для 10 значений группирующей переменной, а не для семи. Произведенные изменения будут действовать, пока все графические устройства не будут закрыты. Так можно изменить любой графический параметр.

23.6. Настройка планок на диаграммах

Для фона планок по умолчанию используются: персиковый цвет для первой условной переменной, бледно-зеленый – для второй и бледно-голубой – для третьей. Цвет, шрифт и другие параметры отображения планок можно настроить. Сделать это можно с помощью метода, описанного в предыдущем разделе, или же можно написать свою функцию.

Для начала определим свою функцию. По аналогии с высокоуровневыми графическими функциями в пакете `lattice` планки тоже позволяют задать функцию, управляющую их содержимым.

Рассмотрим диаграмму, показанную выше на рис. 23.1. Диаграмма отражает зависимость роста певцов Нью-Йоркского хорового общества от партии голоса. Цвет фона персиковый (или лососевый?). А что, если вы захотите, чтобы фон планки был светло-серым, текст планки – черным, а шрифт – курсивным и уменьшенным на 20 %? Это можно сделать с помощью следующего кода:

```
library(lattice)
histogram(~height | voice.part, data = singer,
    strip = strip.custom(bg="lightgrey",
            par.strip.text=list(col="black", cex=.8, font=3)),
    main="Distribution of Heights by Voice Pitch",
    xlab="Height (inches)")
```

Получившаяся диаграмма показана на рис. 23.8.

Параметр `strip=` задает функцию, используемую для настройки внешнего вида планки. Такую функцию можно написать самому (см. `?strip.default`), но часто бывает проще изменить лишь несколько настроек и оставить остальные со значениями по умолчанию. Сделать это можно с помощью функции `strip.custom()`. Параметр `bg` управляет цветом фона, а параметр `par.strip.text` – внешним видом текста на планке.

Параметр `par.strip.text` принимает список, определяющий свойства текста. Параметры `col` и `cex` управляют цветом и размером текста. Параметр `font` может принимать значения 1, 2, 3 или 4, соответствующие обычному, полужирному, курсивному и полужирному курсивному начертаниям.

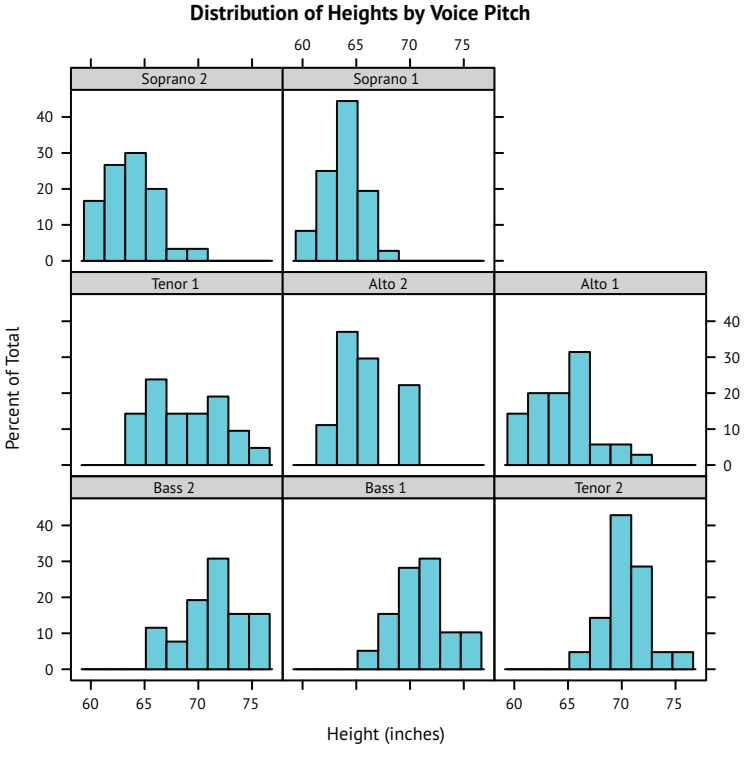


Рис. 23.8. Категоризованная диаграмма с настроенными параметрами отображения планок (светло-серый фон и курсивный шрифт меньшего размера)

Параметр `strip=` изменяет внешний вид планки на заданном графике. Чтобы изменить внешний вид всех планок, создаваемых в сеансе R, можно использовать графические параметры, описанные в предыдущем разделе. Код

```
mysettings <- trellis.par.get()
mysettings$strip.background$col <- c("lightgrey", "lightgreen")
trellis.par.set(mysettings)
```

устанавливает светло-серый фон планки для первой условной переменной и светло-зеленый для второй. Настройки будут действовать до конца сеанса или пока они не будут изменены снова. Использовать графические параметры удобнее, но применение функций дает больше возможностей.

23.7. Размещение диаграмм на странице

В главе 3 вы узнали, как разместить больше двух диаграмм на одной странице при помощи функции `par()`. Поскольку этот подход не действует на функции из пакета `lattice`, нужен другой способ. Самое простое – сохранить диаграммы как объекты, а затем исполь-

зовать функцию `plot()` с определением параметров `split=` или `position=`.

Параметр `split` разделяет страницу на заданное число строк и столбцов, размещая диаграммы в заданные ячейки получившейся матрицы:

```
split=c(x, y, nx, ny)
```

где *x* и *y* задают позицию размещения текущей диаграммы в массиве *nx*×*ny* диаграмм, где начало координат находится вверху слева. Например, следующий код:

```
library(lattice)
graph1 <- histogram(~height | voice.part, data = singer,
                    main = "Heights of Choral Singers by Voice Part" )
graph2 <- bwplot(height~voice.part, data = singer)
plot(graph1, split = c(1, 1, 1, 2))
plot(graph2, split = c(1, 2, 1, 2), newpage = FALSE)
```

поместит первую диаграмму над второй. В частности, первая инструкция `plot()` разделит страницу на один столбец и две строки, разместит диаграмму в первом (единственном) столбце и первой строке (считая сверху вниз и слева направо). Вторая инструкция `plot()` разделит страницу таким же образом, но поместит диаграмму во второй строке. По умолчанию вызов функции `plot()` создает новую страницу. Подавить такое ее поведение можно при помощи параметра `newpage=FALSE`. Получившаяся диаграмма показана на рис. 23.9.

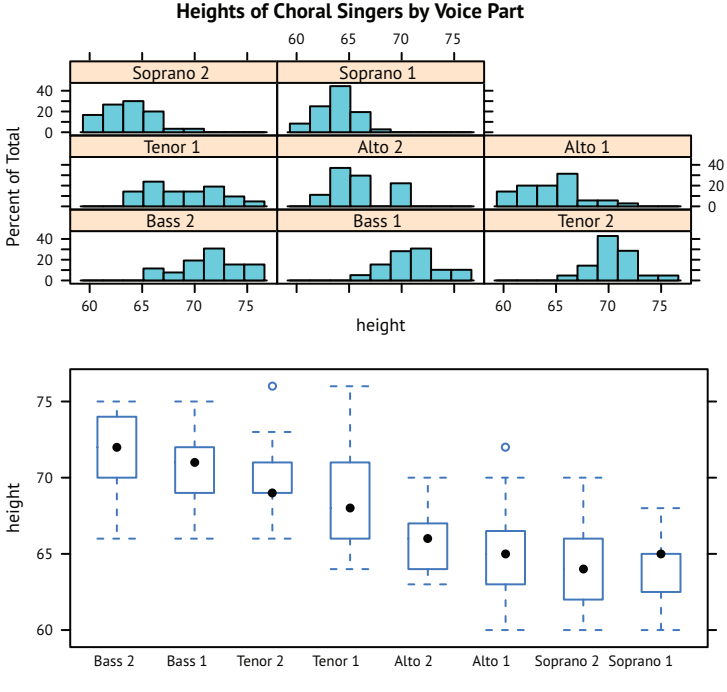


Рис. 23.9. Использование параметра split для объединения диаграмм

Больший контроль над размером диаграмм и их размещением можно получить при помощи параметра `position=`. Рассмотрим следующий программный код:

```
library(lattice)
graph1 <- histogram(~height | voice.part, data = singer,
                    main = "Heights of Choral Singers by Voice Part")
graph2 <- bwplot(height~voice.part, data = singer)
plot(graph1, position=c(0, .3, 1, 1))
plot(graph2, position=c(0, 0, 1, .3), newpage=FALSE)
```

В данном случае параметр `position=c( xmin, ymin, xmax, ymax)` определяет систему координат для страницы – прямоугольную сетку (от 0 до 1 по осям абсцисс и ординат), где левый нижний угол имеет координаты (0, 0). Получившаяся диаграмма показана на рис. 23.10. Дополнительную информацию о позиционировании диаграмм ищите в справке `help(plot.trellis)`.

В категоризованных диаграммах можно также изменять порядок ячеек. Параметр `index.cond=` определяет последовательность уровней условной переменной в высокоуровневых функциях категоризованных диаграмм. Уровни фактора `voice.part`:

```
> levels(singer$voice.part)
[1] "Bass 2" "Bass 1" "Tenor 2" "Tenor 1" "Alto 2"
[6] "Alto 1" "Soprano 2" "Soprano 1"
```

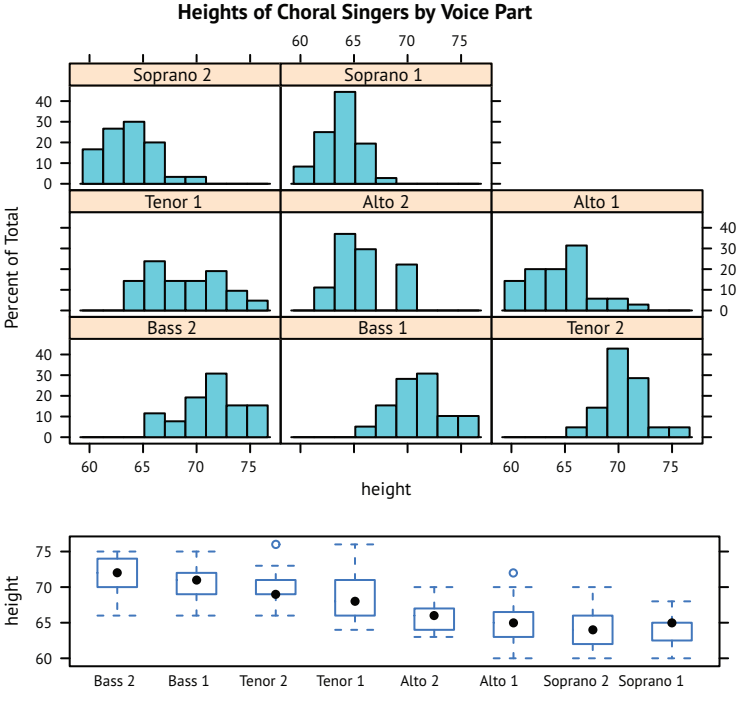


Рис. 23.10. Использование параметра position для размещения диаграмм с большей точностью

Используя эту информацию, можно написать такой код:

```
histogram(~height | voice.part, data = singer,
          index.cond=list(c(2, 4, 6, 8, 1, 3, 5, 7)))
```

который размещает голосовые партии 1 вместе (бас 1, тенор 1, ...), а затем голосовые партии 2 (бас 2, тенор 2, ...). При наличии двух условных переменных включите в список два вектора. В листинге 23.5 добавление `index.cond=list(c(1, 2), c(2, 1))` изменило бы порядок обработки на рис. 23.7. Информацию о параметре `index.cond` ищите в справке `help(xyplot)`.

23.8. Дополнительная информация

Пакет `lattice` предлагает мощное и легко настраиваемое решение для создания диаграмм. Дополнительную информацию о них можно получить из множества источников. Отличные обзоры можно найти в статьях «Lattice Graphics: An Introduction» Дипаяна Саркара (Deepayan Sarkar; 2008, http://mng.bz/jXUG) и «An Introduction to Lattice Graphics in R» Уильяма Дж. Джейкоби (William G. Jacoby; 2010, http://mng.bz/v4TO). А книга Дипаяна Саркара (Deepayan Sarkar; 2008) «Lattice: Multivariate Data Visualization with R» послужит исчерпывающим руководством по этому вопросу.

Послесловие. В погоне за кроликом

В этой книге мы рассмотрели самые разные темы, такие как устройство среды разработки на R, управление данными, традиционные статистические модели и статистические диаграммы. Мы также познакомились с тайными сокровищами, такими как методы повторных выборок, восстановление пропущенных данных и интерактивные диаграммы. Замечательное (или, может быть, приводящее в бешенство) свойство языка R – в нем всегда найдется что-то, чему стоит поучиться.

R – это обширная, мощная и постоянно развивающаяся платформа статистических вычислений и язык программирования. Как можно оставаться в курсе событий при таком большом числе новых пакетов, частых обновлениях и появлении новых направлений развития? К счастью, существует много сайтов с информацией об изменениях в этой статистической платформе и наборе пакетов, а также о новых методологиях. На них можно найти многочисленные руководства, описывающие, как пользоваться платформой и ее пакетами. Ниже я перечислил некоторые из моих любимых сайтов[1].

- Проект R (The R Project for Statistical Computing, http://www.r-project.org/).

  Официальный сайт платформы R и первая инстанция, куда следует обращаться, выясняя любые вопросы об использовании R. На сайте собрана обширная документация, включая «An Introduction to R», «The R Language Definition», «Writing

[1] Конечно, все они англоязычные. Русскоязычные интернет-сообщества можно найти, например, по адресам http://r-statistics.livejournal.com/ и http://vk.com/club8142131. – *Прим. перев.*

R Extensions», «R Data Import/Export», «R Installation and Administration» и «The R FAQ».

- Журнал «The R Journal» (http://journal.r-project.org).

  Рецензируемый бесплатный журнал, в котором публикуются статьи по R и дополнительным пакетам.

- Блоги, посвященные R (R Bloggers, http://www.r-bloggers.com/).

  Интернет-сообщество блогеров, рассказывающих о R. Новые статьи появляются там ежедневно. Я постоянно посещаю этот ресурс.

- БRусника (CRANberries, http://dirk.eddelbuettel.com/cranberries/).

  Сайт, на котором собирается информация о новых и обновленных пакетах со ссылками для их загрузки из CRAN.

- Журнал «Journal of Statistical Software» (http://www.jstatsoft.org/).

  Рецензируемый бесплатный журнал, содержащий статьи, отзывы о книгах и фрагменты программного кода для статистических вычислений. Там часто публикуются статьи по R.

- Тематический указатель CRAN Task Views (http://cran.r-project.org/web/views).

  Содержит рекомендации по использованию R в разных областях прикладных и теоретических исследований. Здесь размещены описания пакетов и методов, подходящих для определенных областей исследований. В настоящее время имеются рекомендации по 41 области[1] (см. табл. ниже).

Перечень задач, рассматриваемых в CRAN Task Views	
Байесовские методы	Развертывание моделей в R
Хемометрия и вычислительная физика	Многомерная статистика
Планирование, мониторинг и анализ клинических испытаний	Обработка лингвистической информации естественного языка
Кластерный анализ и конечные смешанные модели	Вычислительная математика
Базы данных в R	Официальная статистика и методология создания выборок
Дифференциальные уравнения	Оптимизация и математическое программирование
Вероятностные распределения	Анализ данных фармакокинетики

[1] Их число постоянно растет. Этот указатель можно установить командой `install.package()`. – *Прим. перев.*

Перечень задач, рассматриваемых в CRAN Task Views	
Эконометрика	Филогенетика, в особенности сравнительные методы
Анализ экологических данных и состояния окружающей среды	Психометрические модели и методы
Планирование экспериментов и анализ экспериментальных данных	Воспроизводимость результатов исследований
Анализ экстремальных значений	Устойчивые (робастные) статистические методы
Эмпирические финансовые исследования	Статистика в социологии
Анализ функциональных данных	Анализ пространственных данных
Статистическая генетика	Обработка и анализ пространственно-временных данных
Графика, графические модели и визуализация результатов	Анализ выживания
Высокопроизводительные и параллельные вычисления в R	Обучающая статистика
Машинное обучение и статистическое обучение	Анализ временных рядов
Графический анализ медицинских данных	Обработка и анализ данных трассировки
Метаанализ	Веб-технологии и сервисы
Восстановление пропущенных данных	Графические модели в R

- Электронная книга «Big Book of R» (https://www.bigbookofr.com/).
- Список рассылки R (https://www.r-project.org/mail.html).

  Список рассылки по электронной почте – лучшее место, чтобы задать вопрос по R. Поддерживается возможность поиска в архивах. Перед тем как задать вопрос, убедитесь в том, что он отсутствует в списке часто задаваемых вопросов.
- Сайт Cross Validated (http://stats.stackexchange.com), где можно задать вопрос и получить ответ.
- Quick-R (http://www.statmethods.net).
- Мой собственный сайт, посвященный R. На нем размещено более 80 руководств по разным аспектам R. Ложная скромность не позволяет мне сказать больше.
- Data Visualization with R (http://rkabacoff.github.io/datavis).

  Еще один мой сайт, посвященный графическим возможностям R.

Сообщество R всегда готово помочь, полно энергии и вдохновляет на многое. Добро пожаловать в Страну чудес.

Приложение А. Графические пользовательские интерфейсы

Вы заглянули сюда первым делом, не правда ли? По умолчанию в R реализован простой *интерфейс командной строки*. Пользователи вводят команды в ответ на приглашение (> по умолчанию), эти команды выполняются по одной. Для многих, занимающихся обработкой и анализом данных, интерфейс командной строки – это наиболее серьезное препятствие на пути овладения R.

Было предпринято множество попыток по созданию графических интерфейсов, начиная с редакторов кода, взаимодействующих с R (таких как RStudio), и графических интерфейсов для определенных функций и пакетов (таких как BiplotGUI) до развитых графических интерфейсов, позволяющих выполнять анализ при помощи меню и диалогов (таких как R Commander).

В табл. А.1 перечислены наиболее полезные редакторы кода, позволяющие писать и выполнять программный код на R и обладающие такими современными возможностями, как подсветка синтаксиса, автодополнение, исследование объектов, организация проектов и всплывающие подсказки. RStudio – самая популярная *интегрированная среда разработки* для R, но всегда приятно иметь выбор.

Таблица А.1. Интегрированные среды разработки и синтаксические редакторы

Название	URL
RStudio Desktop	https://www.rstudio.com/products/rstudio/
R Tools for Visual Studio	http://mng.bz/VGjr
Eclipse с плагином StatET	https://projects.eclipse.org/projects/science.statet
Architect	https://www.getarchitect.io/
ESS (Emacs Speaks Statistics)	http://ess.r-project.org
Atom Editor with Rbox	https://atom.io/ и https://atom.io/packages/rbox
Notepad++ с плагином NppToR (только для Windows)	http://notepad-plus-plus.org и http://sourceforge.net/projects/npptor

Несколько развитых графических пользовательских интерфейсов для разработки на R перечислены в табл. А.2. Эти интерфейсы не такие всеобъемлющие и удобные, как SAS или IBM SPSS, однако они быстро развиваются.

Таблица А.2. Развитые графические пользовательские интерфейсы для R

Название	URL
JGR/Deducer	http://rforge.net/JGR/ и http://www.deducer.org
R AnalyticFlow	http://r.analyticflow.com/en/
jamovi	https://www.jamovi.org/jmv/
JASP	https://jasp-stats.org/
Rattle (для исследования структуры данных)	http://rattle.togaware.com
R Commander	https://socialsciences.mcmaster.ca/jfox/Misc/Rcmdr/
RkWard	https://rkward.kde.org/
Radiant	https://radiant-rstats.github.io/docs/install.html

Мой любимый графический интерфейс для вводных курсов по статистике – R Commander (показан на рис. А.1).

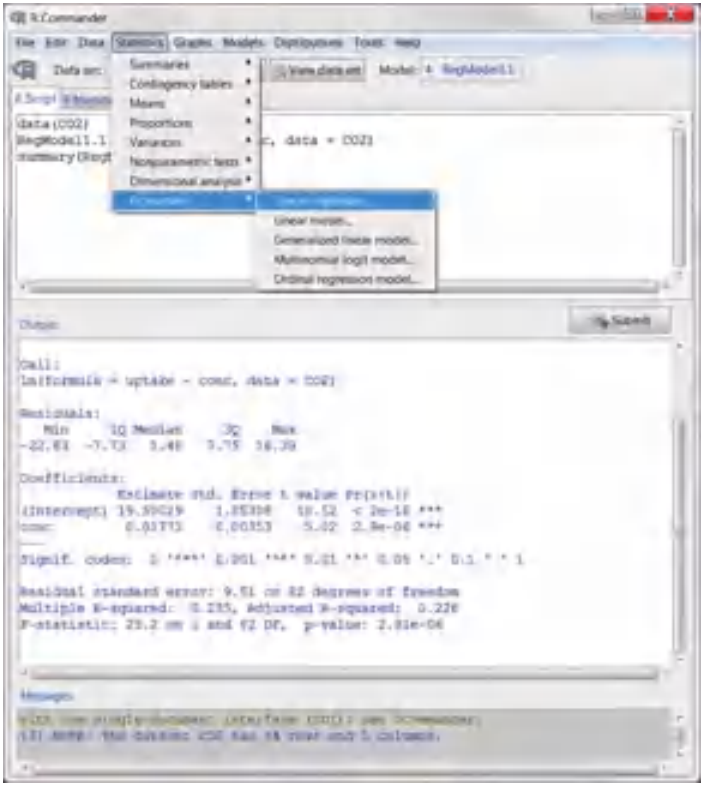


Рис. А.1. Графический интерфейс R Commander

Наконец, существует ряд приложений, которые позволяют создавать графические оболочки для функций R (включая созданные пользователями). К таким приложениям относится генератор графических пользовательских интерфейсов R (R GUI Generator, RGG, http://rgg.r-forge.r-project.org/), а также пакеты `fgui` и `twiddler`, доступные в CRAN. Наиболее комплексный подход в настоящее время предлагает Shiny (https://shiny.rstudio.com/) – инструмент, позволяющий создавать веб-приложения и информационные панели с интерактивным доступом к функциям R.

Приложение B. Начальная настройка окружения

Первое, с чего обычно начинает программист, – это начальная настройка окружения под свои предпочтения. Такая настройка позволяет задать параметры окружения и рабочий каталог, загрузить часто используемые пакеты и пользовательские функции, выбрать сайт CRAN по умолчанию для загрузки и выполнить ряд служебных задач.

Конфигурацию окружения R можно настроить либо при помощи файла инициализации сайта (Rprofile.site), либо файла инициализации каталога (.Rprofile). Это обычные текстовые файлы, содержащие программный код на R, который выполняется при запуске среды окружения.

При запуске R читает файл Rprofile.site из каталога R_HOME/etc, где `R_HOME` – это переменная окружения. Затем отыскивает файл .Rprofile в текущем рабочем каталоге. В случае неудачи поиск будет продолжен в домашнем каталоге пользователя. Вы можете использовать команды `Sys.getenv("R_HOME")`, `Sys.getenv("HOME")` и `getwd()`, чтобы выяснить местонахождение каталогов R_HOME, HOME и текущего рабочего каталога соответственно.

В упомянутых файлах можно разместить две специализированные функции. Функция `.First()` выполняется в начале каждого сеанса R, а функция `.Last()` – в конце. Пример содержимого файла Rprofile.site показан в листинге B.1.

Листинг В.1. Пример содержимого файла .Rprofile

```
options(digits=4)                                                    (1)
options(show.signif.stars=FALSE)                                     (1)
options(scipen=999)                                                  (1)
options
options(prompt="> ")                                                 (2)
options(continue=" ")                                                (2)

options(repos = c(CRAN = "https://cran.rstudio.com/"))               (3)

.libPaths("C:/my_R_library")                                         (4)

.env <- new.env()                                                    (5)
.env$h <- utils::head                                                (5)
.env$t <- utils::tail                                                (5)
.env$s <- base::summary                                              (5)
.env$ht <- function(x){                                              (5)
  base::rbind(utils::head(x), utils::tail(x))                        (5)
}                                                                    (5)
.env$phelp <- function(pckg){                                        (5)
  utils::help(package = deparse(substitute(pckg)))                   (5)
}                                                                    (5)
attach(.env)                                                         (5)

.First <- function(){                                                (6)
  v <- R.Version()                                                   (6)
  msg1 <- paste0(v$version.string, ' -- ', ' "', v$nickname, '"')    (6)
  msg2 <- paste0("Platform: ", v$platform)                           (6)
  cat("\f")                                                          (6)
  cat(msg1, "\n", msg2, "\n\n", sep="")                              (6)
                                                                     (6)
  if(interactive()){                                                 (6)
    suppressMessages(require(tidyverse))                             (6)
  }                                                                  (6)
                                                                     (6)
}                                                                    (6)

.Last <- function(){                                                 (7)
  cat("\nGoodbye at ", date(), "\n")                                 (7)
}                                                                    (7)
```

(1) **Настройка общих параметров.**

(2) **Настройка интерактивного приглашения к вводу команд.**

(3) **Настройка зеркала CRAN по умолчанию.**

(4) **Настройка пути к локальной библиотеке.**

(5) **Настройка сокращений.**

(6) **Функция, выполняемая в начале сеанса.**

(7) **Функция, выполняемая в конце сеанса.**

В этом файле .Rprofile определены следующие настройки:

- настраивается формат вывода данных. Количество значащих цифр устанавливается равным 4 (по умолчанию 7), подавляется вывод звездочек в сводных таблицах коэффициентов, подавляется вывод чисел в экспоненциальном представлении;
- настраивается приглашение к вводу как ">" для первой строки и пробел для строк продолжения (вместо "+");
- выбирается зеркало CRAN по умолчанию cran.rstudio.com, которое будет использовать функция `install.packages()`;
- определяется каталог для устанавливаемых пакетов. Задав значение переменной `.libPaths`, можно сформировать локальную библиотеку пакетов за пределами системы каталогов R. Это может пригодиться для создания резервных копий при обновлении пакетов;
- определяются однобуквенные сокращения для функций `tail()`, `head()` и `summary()`. Определяются новые функции `ht()` и `phelp()`. Функция `ht()` объединяет вывод функций `head` и `tail`. `phelp()` и перечисляет функции в пакете (с сопутствующей справкой);
- функция `.First`:
  - заменяет стандартное приветствие более коротким персонализированным приветствием. У меня новое приветствие выглядит так:

    ```
    R version 4.1.0 (2021-05-18)-"Camp Pontanezen"
    Platform: x86_64-w64-mingw32/x64 (64-bit)
    ```

  - загружает пакеты `tidyverse` в интерактивные сеансы и подавляет вывод начального сообщения `tidyverse`;
- функция `.Last` выводит прощальное сообщение. К тому же это отличное место для выполнения действий по наведению порядка после окончания работы, включая сохранение истории команд, полученных результатов и файлов данных.

ВНИМАНИЕ! Если определять функции и загружать пакеты в файле .Rprofile, а не в сценариях, то они будут менее переносимыми. Ваш код будет зависеть от вашего файла с настройками. Он не будет работать на других машинах, где нет этого файла или в нем определяются другие настройки.

Если вы захотите максимально «зарядить» свой файл .Rprofile, то взгляните на пакет `rprofile` из проекта Jumping Rivers (https://www.jumpingrivers.com/blog/customising-your-rprofile/). Существуют и другие способы настройки окружения на запуске, включая использование параметров командной строки и системных переменных окружения. За более подробной информацией обращайтесь к `help(Startup)` и приложению В в руководстве «Introduction to R» (http://cran.r-project.org/doc/manuals/R-intro.pdf).

Приложение C. Экспорт данных из R

В главе 2 мы рассмотрели широкий спектр методов импортирования данных в R. Однако иногда требуется обратное – экспортировать данные из R, чтобы их можно было заархивировать или импортировать в других программах. В этом приложении описано, как сохранить объект R в виде текстового файла CSV, таблицы Excel или в формате статистического приложения (такого как SPSS, SAS или Stata).

C.1. Текстовый файл CSV

Для сохранения объекта R в виде текстового файла в формате CSV можно использовать функцию `write.table()`. Она имеет следующий синтаксис:

```
write.table(x, outfile, sep=разделитель, quote=TRUE, na="NA"),
```

где *x* – это объект R, а *outfile* – имя файла для сохранения данных. Например, инструкция

```
write.table(mydata, "mydata.txt", sep=",")
```

сохранит набор данных `mydata` в файл mydata.txt с разделителями-запятыми в текущем рабочем каталоге. Если к имени файла добавить путь (например, `"c:/myprojects/mydata.txt"`), то файл будет сохранен в указанном каталоге. Заменив параметр `sep=","` на `sep="\t"`, можно сохранить данные в виде файла с разделителями-табуляторами. По умолчанию строки заключаются в кавычки (`""`), а пропущенные значения заменяются символами `NA`.

C.2. Электронная таблица Excel

Функция `write.xlsx()` из пакета `xlsx` сохраняет таблицы данных R в виде рабочей книги Excel 2007. Она имеет следующий синтаксис:

```
library(xlsx)
write.xlsx(x, outfile, col.Names=TRUE, row.names=TRUE,
           sheetName="Sheet 1", append=FALSE)
```

Например, инструкции

```
library(xlsx)
write.xlsx(mydata, "mydata.xlsx")
```

сохранят таблицу данных `mydata` в виде листа (по умолчанию первого) в рабочей книге Excel mydata.xlsx в текущем рабочем каталоге. По умолчанию имена переменных в наборе данных используются для создания названий столбцов в рабочем листе, а заголовки строк помещаются в первый столбец. Если файл mydata.xlsx уже существует, то он заменяется новым.

C.3. Другие статистические приложения

Для экспорта таблиц данных в другие статистические приложения можно использовать функцию `write.foreign()` из пакета `foreign`. Она создает два файла: текстовый файл в свободном формате, содержащий данные, и файл с инструкциями для чтения данных другими статистическими приложениями. Эта функция имеет следующий синтаксис:

```
write.foreign(dataframe, datafile, codefile, package=package)
```

Например, следующие инструкции:

```
library(foreign)
write.foreign(mydata, "mydata.txt", "mycode.sps", package="SPSS")
```

сохранят таблицу данных `mydata` в виде текстового файла mydata.txt свободного формата в текущем рабочем каталоге и программы для SPSS с названием mycode.sps, которую можно использовать для чтения текстового файла. Параметр `package` может также принимать значения `"SAS"` и `"Stata"`.

Дополнительные сведения об экспорте данных из R можно найти в документации по импорту/экспорту данных в R, доступной по адресу http://cran.r-project.org/doc/manuals/R-data.pdf.

Приложение D. Матричная алгебра в R

Многие из описанных в этой книге функций оперируют матрицами. Манипуляции с матрицами – неотъемлемая составляющая языка R. В табл. D.1 описаны операторы и функции, которые особенно важны для решения задач линейной алгебры. В этой таблице A и B обозначают матрицы, x и b – векторы, а k – скаляр.

Таблица D.1. Функции и операторы матричной алгебры в R

Оператор или функция	Описание
`+ - * / ^`	Поэлементное сложение, вычитание, умножение, деление и возведение в степень соответственно
`A %*% B`	Умножение матриц
`A %o% B`	Внешнее произведение, `AB'`
`cbind(A, B, ...)`	Объединение векторов или матриц в горизонтальной плоскости. Возвращает матрицу
`chol(A)`	Разложение Холецкого для матрицы `A`. Если `R <- chol(A)`, то `chol(A)` содержит фактор верхнетреугольной матрицы, такой что `R'R = A`
`colMeans(A)`	Возвращает вектор, содержащий средние значения для столбцов в `A`
`crossprod(A)`	`A'A`
`crossprod(A,B)`	`A'B`
`colSums(A)`	Возвращает вектор, содержащий суммы значений столбцов в `A`
`diag(A)`	Возвращает вектор, содержащий значения на главной диагонали
`diag(x)`	Создает диагональную матрицу, главная диагональ которой содержит элементы из `x`

Окончание табл. D.1

Оператор или функция	Описание
`diag(k)`	Создает единичную матрицу k×k
`eigen(A)`	Собственные значения и собственные векторы A. Если `y <- eigen(A)`, то: `y$val` – это собственные значения A; `y$vec` – собственные векторы A
`ginv(A)`	Квазиобращение Мура–Пенроуза матрицы A. (Требуется пакет `MASS`)
`qr(A)`	QR-разложение матрицы A. Если `y <- qr(A)`, то верхний треугольник матрицы `y$qr` содержит само разложение, а нижний – информацию о разложении; `y$rank` – ранг A; `y$qraux` – вектор с дополнительной информацией о Q; `y$pivot` содержит информацию об использованной стратегии разложения
`rbind(A, B, ...)`	Объединение векторов или матриц в вертикальной плоскости
`rowMeans(A)`	Возвращает вектор со средними значениями строк в A
`rowSums(A)`	Возвращает вектор с суммами значений по строкам в A
`solve(A)`	Обращение матрицы A, причем A должна быть квадратной матрицей
`solve(A, b)`	Находит вектор x из уравнения `b = Ax`
`svd(A)`	Сингулярное разложение матрицы A. Если `y <- svd(A)`, то: `y$d` – вектор с сингулярными значениями A; `y$u` – матрица со столбцами, содержащими левые сингулярные векторы A; `y$v` – матрица со столбцами, содержащими правые сингулярные векторы A
`t(A)`	Возвращает транспонированную матрицу A

Существует несколько дополнительных пакетов, реализующих матричные преобразования. Пакет `matlab` содержит функции-обертки и переменные, которые с максимально возможной точностью воспроизводят обращения к функциям в программе MATLAB. Они помогают переносить приложения из MATLAB в R. На сайте http://mathesaurus.sourceforge.net/octave-r.html вы найдете удобную памятку по преобразованию инструкций MATLAB в инструкции R.

Пакет `Matrix` содержит функции, расширяющие возможности R, позволяя обрабатывать плотные (dense) или разреженные (sparse) матрицы. Этот пакет включает эффективные реализации алгоритмов BLAS (Basic Linear Algebra Subroutines – базовые подпрограммы линейной алгебры), LAPACK (плотные матрицы), TAUCS (разреженные матрицы) и UMFPACK (разреженные матрицы).

Наконец, в пакете `matrixStats` реализованы методы работы со строками и столбцами матриц, включая функции, которые выполняют подсчет значений, вычисляют суммы, произведения, меры центральной тенденции и разброса данных и многое другое. Все эти функции оптимизированы по скорости выполнения и использованию памяти.

Приложение E. Пакеты, использованные в этой книге

Широта охвата и мощность R в значительной мере обусловлены наличием дополнительных пакетов, созданных самоотверженными пользователями. В табл. E.1 перечислены все дополнительные пакеты, упомянутые в этой книге, вместе с номерами глав, где они обсуждаются. Неекоторые пакеты написаны несколькими авторами. Кроме того, многие пакеты были улучшены третьими лицами. Подробности ищите в документации к пакетам.

Таблица E.1. Дополнительные пакеты, упомянутые в этой книге

Пакет	Авторы	Описание	Главы
AER	Кристиан Клайбер (Christian Kleiber) и Ахим Зейлейс (Achim Zeileis)	Функции, наборы данных, примеры и виньетки из книги «Applied Econometrics with R», написанной Кристианом Клайбером и Ахимом Зейлейсом	13
boot	Оригинальная версия на языке S – Анджело Канти (Angelo Canty). Версия для R – Брайан Рипли (Brian Ripley)	Функции для бутстреп-анализа	12
bootstrap	Оригинальная версия на языке S из библиотеки StatLib – Роб Тибширани (Rob Tibshirani). Версия для R – Фридрих Лейш (Friedrich Leisch)	Бутстреп-анализ, перекрестная проверка, метод последовательного исключения остатков и данные из книги «An Introduction to the Bootstrap» Б. Эфрона (B. Efron) и Р. Тибширани (R. Tibshirani) (Chapman and Hall, 1993 г.)	8

Продолжение табл. Е.1

Пакет	Авторы	Описание	Главы
broom	Дэвид Робинсон (David Robinson), Алекс Хейс (Alex Hayes) и Саймон Коуч (Simon Couch)	Обобщает ключевую информацию о статистических объектах в виде аккуратных таблиц	21
car	Джон Фокс (John Fox), Сэнфорд Вайсберг (Sanford Weisberg) и Брэд Прайс (Brad Price)	Функции для «Companion to Applied Regression»	8, 9, 11
carData	Джон Фокс (John Fox), Сэнфорд Вайсберг (Sanford Weisberg) и Брэд Прайс (Brad Price)	Наборы данных для «Companion to Applied Regression»	7
cluster	Мартин Махлер (Martin Maechler), Питер Руссо (Peter Rousseeuw) (оригинал на языке Fortran), Аня Стройф (Anja Struyf) (оригинал на языке S) и Миа Хьюберт (Mia Hubert) (оригинал на языке S)	Методы кластерного анализа	16
clusterability	Захария Невилл (Zachariah Neville), Наоми Браунштейн (Naomi Brownstein), Майя Акерман (Maya Ackerman) и Андреас Адольфссон (Andreas Adolfsson)	Критерии для проверки наличия кластерных тенденций в наборах данных	16
coin	Торстен Хотхорн (Torsten Hothorn), Курт Хорник (Kurt Hornik), Марк А. ван де Виль (Mark A. van de Wiel) и Ахим Зейлейс (Achim Zeileis)	Процедуры условных умозаключений в критериях с перестановками	12
corrgram	Кевин Райт (Kevin Wright)	Создание коррелограмм	11
DALEX	Пшемыслав Бичек (Przemyslaw Biecek), Шимон Максимюк (Szymon Maksymiuk) и Хьюберт Банеки (Hubert Baniecki)	Язык, независимый от модели, для исследования и объяснения	17
devtools	Хэдли Уикхэм (Hadley Wickham), Джим Хестер (Jim Hester) и Уинстон Чанг (Winston Chang)	Инструменты, упрощающие разработку пакетов	22
directlabels	Тоби Дилан Хокинг (Toby Dylan Hocking)	Размещение меток прямо на графиках	15
doParallel	Мишель Уоллиг (Michelle Wallig), Microsoft Corporation и Стив Уэстон (Steve Weston)	Параллельная реализация `foreach` для пакета `parallel`	20
dplyr	Хэдли Уикхэм (Hadley Wickham), Ромен Франсуа, (Romain François), Лайонел Анри (Lionel Henry) и Кирилл Мюллер (Kirill Müller)	Грамматика для манипуляции данными	3, 5, 6, 7, 9, 16, 19, 21
e1071	Дэвид Мейер (David Meyer), Евгения Димитриаду (Evgenia Dimitriadou), Курт Хорник (Kurt Hornik), Андреас Вайнгессель (Andreas Weingessel) и Фридрих Лейш (Friedrich Leisch)	Различные функции для использования на факультетах статистики и теории вероятностей в Венском университете	17

Продолжение табл. E.1

Пакет	Авторы	Описание	Главы
effects	Джон Фокс (John Fox) и Джангман Хонг (Jangman Hong)	Графическое отображение эффектов для линейных, обобщенных линейных, полиномиальных логит-моделей и логит-моделей отношений шансов	8, 9
factoextra	Альбукадель Кассамбара (Alboukadel Kassambara)	Извлекает и визуализирует результаты многомерного анализа данных	16
flexclust	Фридрих Лейш (Friedrich Leish) и Евгения Димнитриаду (Evgenia Dimnitriadou)	Гибкие алгоритмы кластерного анализа	16
foreach	Мишель Уоллиг (Michelle Wallig), Microsoft Corporation и Стив Уэстон (Steve Weston)	Реализует конструкцию `foreach` для R	20
forecast	Роб Дж. Хайндман (Rob J. Hyndman) и множество других авторов	Функции прогнозирования по временным рядам и линейным моделям	15
gapminder	Дженнифер Брайан (Jennifer Bryan)	Выдержка из данных, доступных на Gapminder.org	19
Ggally	Баррет Шлорке (Barret Schloerke) и множество других авторов	Расширяет возможности `ggplot2`	11
ggdendro	Андри де Врис (Andrie de Vries) и Брайан Д. Рипли (Brian D. Ripley)	Создает дендрограммы с использованием `ggplot2`	16
ggm	Джованни М. Маркетти (Giovanni M. Marchetti), Матиас Дртон (Mathias Drton) и Кайван Садеги (Kayvan Sadeghi)	Инструменты для вычисления корреляций и подгонки моделей методом максимального правдоподобия	7
ggplot2	Хэдли Уикхэм (Hadley Wickam) и множество других авторов	Реализация идеи графической грамматики	4, 6, 9, 10, 11, 12, 15, 16, 18, 19, 20, 21
ggrepel	Камил Словиковски (Kamil Slowikowski)	Автоматическое позиционирование меток на диаграммах `ggplot2`, чтобы избежать их наложения друг на друга	19
gmodels	Грегори Р. Уорнс (Gregory R. Warnes); включает исходный код на R и документацию, написанные Беном Болкером (Ben Bolker), Томасом Ламли (Thomas Lumley) и Рэндаллом С. Джонсоном (Randall C. Johnson). Вклад Рэндалла С. Джонсона защищен авторским правом (SAIC-Frederick, Inc., 2005)	Различные программные средства для подгонки моделей	7

Продолжение табл. Е.1

Пакет	Авторы	Описание	Главы
haven	Хэдли Уикхэм (Hadley Wickham) и Эван Миллер (Evan Miller)	Импорт и экспорт файлов в форматах SPSS, Stata и SAS	2
Hmisc	Фрэнк Э. Харрелл-младший (Frank E. Harrell, Jr.)	Разные функции для анализа данных, высокоуровневой графики, вспомогательных операций и т. д.	7
ISLR	Гарет Джеймс (Gareth James), Даниэла Виттен (Daniela Witten), Тревор Хасти (Trevor Hastie) и Роб Тибширани (Rob Tibshirani)	Данные для книги «Introduction to Statistical Learning with Applications in R»	19
kableExtra	Нао Чжу (Hao Zhu)	Конструирование сложных HTML-страниц и таблиц LaTeX	21
knitr	Ихуи Се (Yihui Xie)	Пакет для создания динамических отчетов на R	21
leaps	Томас Ламли (Thomas Lumley) с использованием кода на Fortran, написанного Аланом Миллером (Alan Miller)	Выбор подмножества регрессии посредством полного перебора вариантов	8
lmPerm	Боб Уилер (Bob Wheeler)	Критерии с перестановками для линейных моделей	12
MASS	Оригинал на языке S Венейблс (Venables) и Рипли (Ripley); адаптированная версия на R Брайан Рипли (Brian Ripley), после более ранней работы Курта Хорника (Kurt Hornik) и Альбрехта Гебхардта (Albrecht Gebhardt)	Функции и наборы данных для книги Venables & Ripley's «Modern Applied Statistics with S» (4-е изд., Springer, 2003)	7, 9, 12, 13, 14, приложение D
mice	Стеф ван Бюрен (Stef van Buuren) и Карин Гротуис-Оудсхорн (Karin Groothuis-Oudshoorn	Восстановление многомерных пропущенных данных с помощью цепных уравнений	18
mosaicData	Рэндалл Пруим (Randall Pruim), Дэниел Каплан (Daniel Kaplan) и Николас Хортон (Nicholas Horton)	Наборы данных для проекта MOSAIC	4
multcomp	Торстен Хотхорн (Torsten Hothorn), Фрэнк Бретц (Frank Bretz), Питер Вестфолл (Peter Westfall), Ричард М. Хейбергер (Richard M. Heiberger) и Андре Шутценмейстер (Andre Schuetzenmeister)	Одновременное вычисление критериев и доверительных интервалов для обобщенных линейных гипотез в параметрических моделях, включая линейные, обобщенные линейные, со смешанными линейными эффектами и модели выживания	9, 12, 21
MultiRNG	Хакан Демирташ (Hakan Demirtas), Раван Аллози (Rawan Allozi) и Ран Гао (Ran Gao)	Генераторы псевдослучайных чисел для 11 распределений	5
mvoutlier	Мориц Гшвандтнер (Moritz Gschwandtner) и Питер Фильцмозер (Peter Filzmoser)	Обнаружение выбросов в многомерных данных при помощи робастных методов	9

Продолжение табл. Е.1

Пакет	Авторы	Описание	Главы
naniar	Николас Тирни (Nicholas Tierney), Ди Кук (Di Cook), Майлз МакБейн (Miles McBain) и Колин Фэй (Colin Fay)	Структуры данных, обобщенные статистики и инструменты визуализации для анализа пропущенных данных	18
NbClust	Малика Чаррад (Malika Charrad), Надя Газзали (Nadia Ghazzali), Вероник Буато (Veronique Boiteau) и Азам Никнафс (Azam Niknafs)	Исследование показателей для определения количества кластеров	16
partykit	Торстен Хотхорн (Torsten Hothorn), Хайди Зайболд (Heidi Seibold) и Ахим Зейлейс (Achim Zeileis)	Инструменты для рекурсивного сегментирования	17
pastecs	Фредерик Ибанез (Frederic Ibanez), Филипп Грожан (Philippe Grosjean) и Мишель Этьен (Michele Etienne)	Пакет для анализа пространственно-временных рядов с экологической информацией	7
patchwork	Томас Лин Педерсен (Thomas Lin Pedersen)	Составляет композицию из нескольких диаграмм	19
pkgdown	Хэдли Уикхэм (Hadley Wickham) и Джей Хессельберт (Jay Hesselberth)	Создает хорошо оформленную документацию и веб-сайт для пакета	22
plotly	Карсон Зиверт (Carson Sievert) и многие другие авторы	Создает привлекательную веб-графику, используя plotly.js	19
psych	Уильям Ревелл (William) Revelle	Процедуры для психологических, психометрических и личностных исследований	7, 14
pwr	Стефан Шампели (Stephane Champely)	Базовые функции для анализа мощности	10
qcc	Лука Скрукка (Luca Scrucca)	Диаграммы контроля качества	13
randomForest	Оригинал на Fortran Лео Бреймана (Leo Breiman) и Адель Катлер (Adele Cutler); адаптация на R Энди Лио (Andy Liaw) и Мэтью Винера (Matthew Wiener)	Случайные леса Бреймана и Катлер для классификации и регрессии	17
rattle	Грэм Уильямс (Graham Williams), Марк Вир Калп (Mark Vere Culp), Эд Кокс (Ed Cox), Энтони Нолан (Anthony Nolan), Денис Уайт (Denis White), Даниэле Медри (Daniele Medri), Акбар Валджи (Akbar Waljee) (оценка ошибки вне выборки для случайных лесов) и Брайан Рипли (Brian Ripley) (оригинальный автор print.summary.nnet)	Графический интерфейс для исследования данных с использованием R	16, 17
readr	Хэдли Уикхэм (Hadley Wickham) и Джим Хестер (Jim Hester)	Гибкий импорт прямоугольных текстовых данных	2
readxl	Хэдли Уикхэм (Hadley Wickham) и Дженнифер Брайан (Jennifer Bryan)	Импорт файлов Excel	2

Продолжение табл. Е.1

Пакет	Авторы	Описание	Главы
rgl	Дэниел Адлер (Daniel Adler) и Дункан Мердок (Duncan Murdoch)	Система трехмерной визуализации (OpenGL)	11
rmarkdown	Джей-джей Аллер (JJ Allaire), Ихуи Се (Yihui Xie)	Преобразование документов R Markdown в различные форматы	21
robustbase	Мартин Махлер (Martin Maechler) и многие другие авторы	Инструменты для анализа данных с применением робастных методов	13
rpart	Терри Терно (Terry Therneau), Бет Аткинсон (Beth Atkinson) и Брайан Рипли (Brian Ripley) (автор первоначальной адаптации для R)	Рекурсивное сегментирование и деревья регрессии	17
rrcov	Валентин Тодоров (Valentin Todorov)	Робастная оценка местоположения и разброса, а также робастный многомерный анализ с высокой степенью надежности	9
scales	Хэдли Уикхэм (Hadley Wickham) и Дана Зайдель (Dana Seidel)	Функции масштабирования осей для визуализации данных	6, 19
scatterplot3d	Уве Лиггес (Uwe Ligges)	Изображение облака точек в трехмерном пространстве	11
showtext	Исюань Сью (Yixuan Qiu) и многие другие авторы	Поддержка использования разных шрифтов на диаграммах R	19
sqldf	Г. Гротендик (G. Grothendieck)	SQL-подобный язык для манипулирования таблицами данных	3
tidyquant	Мэтт Данчо (Matt Dancho) и Дэвис Воган (Davis Vaughan)	Инструменты для количественного финансового анализа	21
tidyr	Хэдли Уикхэм (Hadley Wickham)	Инструменты для преобразования таблиц данных из одного формата в другой, вычисления сводных статистик, а также добавления и удаления данных	5, 16, 20
treemapify	Дэвид Уилкинс (David Wilkins)	Функции визуализации плоских деревьев с помощью `ggplot2`	6
tseries	Адриан Траплетти (Adrian Trapletti) и Курт Хорник (Kurt Hornik)	Анализ временных рядов и финансовые вычисления	15
usethis	Хэдли Уикхэм (Hadley Wickham) и Дженнифер Брайан (Jennifer Bryan)	Автоматизация задач настройки проектов пакетов	22
vcd	Дэвид Мейер (David Meyer), Ахим Зейлейс (Achim Zeileis) и Курт Хорник (Kurt Hornik)	Функции для визуализации категориальных данных	1, 6, 7, 11, 12

Окончание табл. Е.1

Пакет	Авторы	Описание	Главы
VIM	Матиас Темпл (Matthias Templ), Андреас Альфонс (Andreas Alfons) и Александр Коварик (Alexander Kowarik)	Визуализация и восстановление пропущенных значений	18
xtable	Дэвид Б. Даль (David B. Dahl) и многие другие авторы	Экспорт таблиц в LaTeX и HTML	21
xts	Джеффри А. Райан (Jeffrey A. Ryan) и Джошуа М. Ульрих (Joshua M. Ulrich)	Единообразная обработка различных классов временны́х данных	15

Приложение F. Работа с большими наборами данных

Все объекты R хранятся в виртуальной памяти. Для многих из нас эта особенность обернулась запоминающимся опытом интерактивной работы, однако для аналитиков, имеющих дело с большими объемами данных, это может замедлить выполнение программ и привести к ошибкам нехватки памяти.

Ограничения памяти в основном зависят от разрядности сборки R (32 или 64 бита), а для 32-битной версии для Windows – еще и от версии операционной системы. Сообщения об ошибке, начинающиеся со слов «cannot allocate vector of size» (не могу разместить вектор данного размера), как правило, указывают на превышение лимита адресного пространства, а сообщения, начинающиеся со слов «cannot allocate vector of length» (не могу разместить вектор данной длины), свидетельствуют о превышении допустимой длины вектора. При работе с большими наборами данных старайтесь по возможности использовать 64-битную версию программы. Дополнительную информацию ищите в справке `help(Memory)`.

Есть три аспекта, которые нужно учитывать при работе с большими объемами данных: эффективное программирование для ускорения выполнения программ; хранение данных на внешних носителях для уменьшения объема используемой памяти; использование специальных статистических методов, разработанных для эффективного анализа больших объемов данных. Сначала рассмотрим простые решения каждого из этих аспектов, а затем познакомимся с более сложными (и комплексными) решениями, используемыми при обработке *больших* данных.

F.1. Эффективное программирование

Программисты могут использовать разные приемы, помогающие оптимизировать обработку больших наборов данных.

- Используйте векторизованные вычисления, если есть такая возможность. Применяйте встроенные функции R для работы с векторами, матрицами и списками (например, `ifelse`, `colMeans` и `rowSums`) и избегайте циклов (`for` и `while`).
- Используйте матрицы вместо таблиц данных (они обрабатываются намного быстрее).
- Для импорта текстовых файлов в формате CSV используйте оптимизированные функции, такие как `fread()` из пакета `data.table` или `vroom()` из пакета `vroom`. Они работают значительно быстрее, чем функция `read.table()` из стандартной библиотеки R.
- Сразу правильно устанавливайте размеры объектов, а не увеличивайте их постепенно, добавляя значения.
- Используйте распараллеливание для повторяющихся вычислительных задач.
- Проверяйте программы на небольшой части данных, чтобы оптимизировать программный код и удалить ошибки, перед тем как пытаться обработать большой набор данных целиком.
- Удаляйте временные объекты и объекты, ставшие ненужными. Вызов `rm(list=ls())` удалит все объекты из памяти. Отдельные объекты можно удалить вызовом `rm(`*`object`*`)`. После удаления больших объектов вызовите `gc()`, чтобы инициировать сборку мусора и гарантировать полное удаление объектов из памяти.
- Используйте функцию `.ls.objects()`, описанную Джероми Англимом (Jeromy Anglim) в статье «Memory Management in R: A Few Tips and Tricks» в его блоге (http://mng.bz/xGpq). Эта функция создает список всех объектов рабочего пространства, отсортированных по размеру, и помогает выявить «прожорливые» объекты и разобраться с ними.
- Профилируйте свои программы, чтобы узнать, сколько времени затрачивается на выполнение каждой функции. Эту задачу можно решить при помощи функций `Rprof()` и `summaryRprof()`. Функция `system.time()` тоже может пригодиться. Пакеты `profr` и `prooftools` содержат функции, которые можно использовать и для профилирования программы.
- Используйте внешние подпрограммы, написанные на компилируемых языках. Для передачи объектов из R в функции на C++ и обратно можно использовать пакет `Rcpp`.

Примеры векторизации, эффективного ввода данных, правильного определения размеров объектов и распараллеливания можно найти в разделе 20.5.

Увеличение эффективности программного кода – это только полдела при работе с большими объемами данных. Для преодоления ограничений памяти можно организовать хранение данных вне программы и использовать специализированные методы анализа.

F.2. Хранение данных вне оперативной памяти

Существует несколько пакетов, предназначенных для хранения данных вне оперативной памяти R. Идея заключается в том, чтобы хранить данные во внешних базах данных или в неструктурированных двоичных файлах на диске, а затем загружать их частями по мере необходимости. Несколько полезных пакетов описаны в табл. F.1.

Таблица F.1. Пакеты R для организации доступа к большим наборам данных

Пакет	Описание
`bigmemory`	Поддерживает создание, хранение и манипуляции с большими матрицами. Размещает матрицы в разделяемой памяти и в файлах, отображаемых в память
`disk.frame`	Фреймворк для обработки данных на диске, объем которых превышает объем оперативной памяти
`ff`	Предлагает структуры данных, которые хранятся на диске, но ведут себя так, будто находятся в оперативной памяти
`filehash`	Реализует идею простых баз данных типа «ключ-значение», где строковые ключи ассоциированы со значениями данных, хранящихся на диске
`ncdf, ncdf4`	Предоставляет интерфейс для файлов данных в формате Unidata netCDF
`RODBC, RMySQL, ROracle, RPostgreSQL, RSQLite`	Все эти пакеты дают возможность доступа к внешним реляционным базам данных

Перечисленные выше пакеты помогают преодолеть ограничения памяти. Однако при попытках анализа обширных наборов данных за разумное время требуются специализированные методы. Некоторые из них описаны ниже.

F.3. Аналитические пакеты для больших объемов данных

В R есть несколько пакетов, предназначенных для анализа больших объемов данных:

- пакеты `biglm` и `speedglm` подбирают линейные или обобщенные линейные модели на больших наборах данных с эффек-

тивным использованием памяти. В этих пакетах реализованы аналоги функций `lm()` и `glm()` для работы с большими объемами данных;

- некоторые пакеты предлагают аналитические функции для работы с большими матрицами, создаваемыми при помощи пакета `bigmemory`. Пакет `biganalytics` позволяет проводить кластерный анализ методом *k*-средних, вычислять статистики для переменных и создает программную оболочку для пакета `biglm`. Пакет `bigtabulate` содержит аналоги функций `table()`, `split()` и `tapply()`, а пакет `bigalgebra` позволяет применять продвинутые функции линейной алгебры;
- в пакете `biglars` в сочетании с пакетом `ff` представлены возможности регрессии методами наименьшего угла (least-angle), лассо и ступенчатой регрессии по наборам данных, не умещающимся в памяти;
- пакет `data.table` предлагает улучшенную версию `data.frame`, обеспечивающую более быстрые агрегирование, соединение упорядоченных и перекрывающихся диапазонов и добавление, изменение и удаление столбца по ссылке (без копирования). Структуру `data.table` можно использовать для работы с большими наборами данных (например, 100 Гбайт), и она совместима со всеми функциями R, принимающими таблицы данных.

Все эти пакеты адаптированы для работы с большими наборами данных и относительно просты в использовании. Далее описываются более комплексные решения для анализа наборов данных, объем которых исчисляется терабайтами.

F.4. Комплексные решения для работы с огромными наборами данных

Когда я писал первое издание этой книги, самое большее, что я мог сказать в этом разделе, было: «Мы стараемся». С тех пор появилось большое число проектов, объединяющих *высокопроизводительные вычисления* и язык R. В этом разделе я расскажу о некоторых из наиболее популярных подходов к использованию R с наборами данных терабайтного объема. Каждый из них требует некоторого знакомства с высокопроизводительными вычислениями и другими программными платформами, такими как Hadoop (бесплатный фреймворк на основе Java для обработки больших наборов данных в распределенном вычислительном окружении).

В табл. F.2 кратко описаны решения с открытым исходным кодом. Наиболее популярными являются RHadoop и sparklyr.

Таблица F.2. Платформы с открытым исходным кодом для работы с большими данными в R

Подход	Описание
RHadoop	Программное обеспечение для управления и анализа данных с помощью Hadoop в среде R. Состоит из пяти взаимосвязанных пакетов: `rhbase`, `ravro`, `rhdfs`, `plyrmr` и `rmr2`. Подробности ищите по адресу https://github.com/RevolutionAnalytics/RHadoop/wiki
RHIPE	*R и интегрированная среда программирования Hadoop* (R and Hadoop Integrated Programming Environment, RHIPE). Пакет для R, позволяющий запускать задания Hadoop MapReduce из R (https://github.com/delta-rho/RHIPE)
Hadoop Streaming	Hadoop Streaming (https://hadoop.apache.org/docs/r1.2.1/streaming.html) – это утилита для создания и запуска заданий Map/Reduce на любом языке. Пакет `HadoopStreaming` реализует поддержку создания таких заданий на R. Подробности ищите по адресу https://cran.rstudio.com/web/packages/HadoopStreaming/
RHIVE	Расширение для R, упрощающее распределенные вычисления с помощью запросов HIVE. RHIVE поддерживает простой язык запросов HIVE SQL для R, в инструкциях которого можно использовать объекты и функции R. Подробности ищите по адресу https://github.com/nexr/RHive.
pbdR	«Программирование обработки больших данных на R» (Programming with Big Data in R). Набор пакетов, поддерживающих параллельную обработку данных на R, предлагая простой интерфейс к масштабируемым высокопроизводительным библиотекам (таким как MPI, ZeroMQ, ScaLAPACK, netCDF4 и PAPI). Кроме того, pbdR поддерживает модель «одна программа – множество данных» (single program, multiple data; SPMD) в крупномасштабных вычислительных кластерах. Подробности ищите по адресу http://r-pbd.org
sparklyr	Пакет, предлагающий интерфейс к Apache Spark для R. Поддерживает подключение к локальным и удаленным кластерам Apache Spark, предоставляет серверную часть, совместимую с dplyr, и интерфейс для доступа к алгоритмам машинного обучения Spark. Подробности ищите по адресу https://cran.r-project.org/web/packages/sparklyr

Облачные сервисы предлагают готовую масштабируемую инфраструктуру с потенциально огромными ресурсами памяти и хранилищ. Самые популярные облачные сервисы для пользователей R предоставляются компаниями Amazon, Microsoft и Google. Платформа Oracle не является облачным решением, но она тоже предлагает поддержку больших данных для пользователей R (табл. F.3).

Таблица F.3. Коммерческие платформы для работы с большими данными в R

Подход	Описание
Amazon Web Services (AWS)	Существует несколько подходов к использованию R с AWS. Пакет `paws` (с поддержкой Amazon Web Services) предоставляет полный доступ к набору сервисов AWS из R (https://paws-r.github.io/). Пакет `aws.ec2` – это простой клиентский пакет для AWS EC2 REST API (https://github.com/cloudyr/aws.ec2). Луи Аслетт (Louis Aslett) поддерживает образы машин Amazon, что делает развертывание сервера RStudio в Amazon EC2 относительно простым делом (https://www.louisaslett.com/RStudio_AMI/)
Microsoft Azure	Виртуальные машины для обработки данных – это образы виртуальных машин на облачной платформе Azure, созданные для обработки данных. К ним относятся Microsoft R Open, Microsoft Machine Learning Server, RStudio Desktop и RStudio Server. Дополнительные сведения можно найти на странице https://azure.microsoft.com/ru-ru/services/virtual-machines/data-science-virtual-machines/. Также обратите внимание на AzureR – семейство пакетов для работы с Azure из R (https://github.com/Azure/AzureR)

Окончание табл. F.3

Подход	Описание
Google Cloud Services	Пакет `bigquery` предоставляет интерфейс к Google BigQuery API (https://github.com/r-dbi/bigrquery). Пакет `googleComputeEngineR` предоставляет интерфейс к API Google Cloud Compute Engine – он максимально упрощает развертывание облачных ресурсов для R (включая RStudio и Shiny) (https://cloudyr.github.io/googleComputeEngineR/)
Oracle R Advanced Analytics for Hadoop	Набор пакетов для R, предоставляющих интерфейс к таблицам HIVE, инфраструктуре Hadoop, таблицам базы данных Oracle и локальному окружению R. Включает широкий спектр методов прогнозной аналитики. Подробности ищите в файлах справки Oracle (http://mng.bz/K4jn)

В дополнение к ресурсам, указанным в табл. F.3, обратите внимание на проект cloudyr (https://cloudyr.github.io/), цель которого – упростить облачные вычисления в R. Он содержит широкий спектр пакетов, помогающих объединить преимущества R и облачных сервисов.

Обработка наборов данных с размерами от гигабайта до терабайта – сложная задача для любого языка. Каждый из этих подходов требует времени на изучение. Хорошим источником информации по использованию R с Hadoop может служить книга «Big Data Analytics with R and Hadoop» (Prajapati, 2013). Для знакомства с платформой Spark я рекомендую руководства «Mastering Spark with R» (https://therinspark.com/) и «Using Spark from R for Performance with Arbitrary Code» (https://sparkfromr.com/). Наконец, не забывайте про каталог задач CRAN Task View: «High-Performance and Parallel Computing with R» (https://cran.r-project.org/web/views/HighPerformanceComputing.html). Эта область продолжает быстро меняться и развиваться, поэтому почаще проверяйте каталог CRAN Task View.

Приложение G. Обновление версии R

С позиции потребителей, мы воспринимаем как должное возможность обновления программного обеспечения. В первой главе я отметил, что скачать и установить последние версии пакетов можно при помощи функции `update.packages()`. К сожалению, для обновления самой среды программирования подобной функции не существует.

Чтобы заменить версию R с 5.1.0 на 6.1.1, придется проявить изобретательность. (На момент написания книги текущая версия R имела номер 4.1.1, но я хочу, чтобы эта книга казалась современной спустя годы.) Далее я покажу два способа обновления: автоматизированный, с использованием пакета `installr`, и ручной, работающий на всех платформах.

G.1. Автоматизированное обновление R (только для Windows)

Для обновления версии R пользователи Windows могут использовать пакет `installr`. Сначала установите и загрузите пакет:

```
install.packages("installr")
library(installr)
```

После этого в меню графического интерфейса появится пункт Update (Обновить), как показано на рис. G.1.

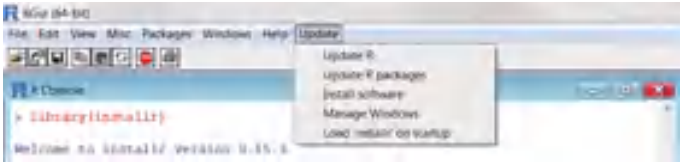


Рис. G.1. Меню Update (Обновить), появившееся в графическом интерфейсе R после установки пакета installr

Это меню позволяет установить новую версию R, обновить существующие пакеты и установить другое полезное программное обеспечение (например, RStudio). В настоящее время пакет `installr` доступен только для платформы Windows. Для пользователей macOS, а также для пользователей Windows, не желающих применять `installr`, можно рекомендовать обновлять R вручную.

G.2. Обновление R вручную (для Windows и macOS)

Скачать и установить последнюю версию R из архива CRAN (http://cran.r-project.org/bin/) относительно просто. Все осложняется тем, что настройки (в том числе уже установленные дополнительные пакеты) не будут включены в новую версию. У меня установлено более 500 дополнительных пакетов, и я действительно не хочу выписывать их названия и загружать их вручную, когда придет время обновить R.

В Сети долго дискутировали о наиболее изящном и рациональном способе обновления R. Описанный ниже способ ни изящен, ни рационален, но зато я обнаружил, что он хорошо действует в разных операционных системах (Windows и macOS).

Этот подход предполагает сохранение списка всех установленных пакетов вне дерева каталогов R с помощью функции `installed.packages()` и его последующую передачу функции `install.packages()`, которая загрузит и установит последние версии этих пакетов. Вот этапы этой процедуры:

1) если вы изменяли файл Rprofile.site (приложение B), сохраните его копию за пределами дерева каталогов R;
2) запустите текущую версию R и выполните следующие инструкции:

```
oldip <- installed.packages()[,1]
save(oldip, fi le="path/installedPackages.Rdata")
```

   где `path` – это каталог, куда был сохранен список установленных пакетов;
3) скачайте и установите новую версию R;
4) если вы сохранили измененный файл Rprofile.site на этапе 1, скопируйте его в новую версию R;

5) запустите новую версию R и выполните следующие инструкции:

```
load("path/installedPackages.Rdata")
newip <- installed.packages()[,1]
for(i in setdiff(oldip, newip)){
  install.packages(i)
}
```

где path – это каталог, указанный в инструкции на этапе 2;

6) удалите старую версию программы (по желанию).

При помощи этого способа можно установить только те пакеты, которые находятся в архиве CRAN. Вам придется отдельно найти и установить пакеты, распространяемые через другие репозитории. К счастью, в конце описанной выше процедуры появится список пакетов, которые не могут быть установлены из CRAN. Когда я последний раз обновлял программу, то у меня не были обнаружены пакеты globaltest и Biobase. Поскольку я получил их с сайта Bioconductor, эти пакеты можно установить так:

```
source("http://bioconductor.org/biocLite.R")
biocLite("globaltest")
biocLite("Biobase")
```

На шестом этапе было предложено удалить старую версию программы. В Windows одновременно может быть установлено несколько версий R. При необходимости старую версию можно удалить, выбрав Start > Control Panel > Uninstall a Program (Пуск > Панель управления > Удаление программ). Чтобы удалить старую версию R в macOS, запустите программу Finder, перейдите в каталог /Library/Frameworks/R.frameworks/versions/ и удалите папку, соответствующую старой версии.

Очевидно, что обновление текущей версии R требует больше усилий, чем это необходимо для такой совершенной программы. Я надеюсь, что когда-нибудь в этом приложении будет просто написано «Для обновления текущей версии R выберите в меню пункт Check for Updates (Проверить наличие обновлений)».

G.3. Обновление R в Linux

Процесс обновления R в Linux сильно отличается от процесса, используемого на компьютерах с Windows и macOS. Кроме того, многое зависит от дистрибутива (Debian, Red Hat, SUSE или Ubuntu). Более полную информацию ищите на странице http://cran.r-project.org/bin/linux.

Список литературы

1 Allison, T. and D. Chichetti. 1976. «Sleep in Mammals: Ecological and Constitutional Correlates». *Science* 194 (4266): 732–734.

2 Anderson, M. J. 2006. «Distance-based Tests for Homogeneity of Multivariate Dispersions». *Biometrics* 62: 245–253.

3 Baade, R. and R. Dye. 1990. «The Impact of Stadiums and Professional Sports on Metropolitan Area Development». *Growth and Change* 21: 1–14.

4 Bandalos, D. L. and M. R. Boehm-Kaufman. 2009. «Four Common Misconceptions in Exploratory Factor Analysis». In *Statistical and Methodological Myths and Urban Legends*, edited by C. E. Lance and R. J. Vandenberg, 61–87. New York: Routledge.

5 Bates, D. 2005. «Fitting Linear Mixed Models in R». *R News 5* (1): 27–30. www.r-project.org/doc/Rnews/Rnews_2005-1.pdf.

6 Breslow, N. and D. Clayton. 1993. «Approximate Inference in Generalized Linear Mixed Models». *Journal of the American Statistical Association* 88:9–25.

7 Bretz, F., T. Hothorn, and P. Westfall. 2010. *Multiple Comparisons Using R*. Boca Raton, FL: Chapman & Hall.

8 Canty, A. J. 2002. «Resampling Methods in R: The boot Package». *R News 2* (3): 2–7. www.rproject.org/doc/Rnews/Rnews_2002-3.pdf.

9 Chambers, J. M. 2008. *Software for Data Analysis: Programming with R*. New York: Springer.

10 Chambers, J., W. Cleveland, B. Kleiner, and P. Tukey. 1983. *Graphical Methods for Data Analysis*. Wadsworth & Brooks/Cole Statistics/Probability Series. Boston: Duxbury Press.

11 Chollet, F., and J. Allaire. 2018. *Deep Learning with R*[1]. NY: Manning.

[1] *Шолле Франсуа*. Глубокое обучение на R. Руководство. Питер, 2018. ISBN: 978-5-4461-0902-9. – *Прим. перев.*

12 Ciaburro, G. and B. Venkateswaran. 2017. *Neural networks with R.* Birmingham: Packt.

13 Cleveland, W. 1981. «LOWESS: A Program for Smoothing Scatter Plots by Robust Locally Weighted Regression». *The American Statistician* 35:54.

14 1993. *Visualizing Data. Summit*, NJ: Hobart Press.

15 Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed. Hillsdale, NJ: Lawrence Erlbaum.

16 Cowpertwait, P. S. and A. V. Metcalfe. 2009. *Introductory Time Series with R.* Auckland, New Zealand: Springer.

17 Coxe, S., S. West, and L. Aiken. 2009. «The Analysis of Count Data: A Gentle Introduction to Poisson Regression and Its Alternatives». *Journal of Personality Assessment* 91: 121–136.

18 Culbertson, W. and D. Bradford. 1991. «The Price of Beer: Some Evidence for Interstate Comparisons». *International Journal of Industrial Organization* 9: 275–289.

19 DiStefano, C., M. Zhu, and D. Mîndrila. 2009. «Understanding and Using Factor Scores: Considerations for the Applied Researcher». *Practical Assessment, Research & Evaluation* 14 (20). http://pareonline.net/pdf/v14n20.pdf.

20 Dobson, A. and A. Barnett. 2008. *An Introduction to Generalized Linear Models*, 3rd ed. Boca Raton, FL: Chapman & Hall.

21 Dunteman, G. and M-H Ho. 2006. *An Introduction to Generalized Linear Models.* Thousand Oaks, CA: Sage.

22 Efron, B. and R. Tibshirani. 1998. *An Introduction to the Bootstrap.* New York: Chapman & Hall.

23 Enders, C. K. 2010. *Applied Missing Data Analysis.* New York: Guilford Press.

24 Everitt, B. S., S. Landau, M. Leese, and D. Stahl. 2011. *Cluster Analysis*, 5th ed. London: Wiley.

25 Fair, R. C. 1978. «A Theory of Extramarital Affairs». *Journal of Political Economy* 86: 45–61.

26 Faraway, J. 2006. *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models.* Boca Raton, FL: Chapman & Hall.

27 Fawcett, T. 2005. «An Introduction to ROC Analysis». *Pattern Recognition Letters* 27: 861–874.

28 Forte, R. M. 2015. *Mastering Predictive Analytics with R.* Birmingham: Packt.

29 Fox, J. 2002. *An R and S-Plus Companion to Applied Regression.* Thousand Oaks, CA: Sage.

30 2002. «Bootstrapping Regression Models». http://mng.bz/pY9m.

31 2008. *Applied Regression Analysis and Generalized Linear Models*. Thousand Oaks, CA: Sage.

32 Fwa, T., ed. 2006. *The Handbook of Highway Engineering*, 2nd ed. Boca Raton, FL: CRC Press.

33 Gentleman, R. 2009. *R Programming for Bioinformatics*. Boca Raton, FL: Chapman &Hall/CRC.

34 Good, P. 2006. *Resampling Methods: A Practical Guide to Data Analysis*, 3rd ed. Boston: Birkhäuser.

35 Gorsuch, R. L. 1983. *Factor Analysis*, 2nd ed. Hillsdale, NJ: Lawrence Erlbaum.

36 Greene, W. H. 2003. *Econometric Analysis*, 5th ed. Upper Saddle River, NJ: Prentice Hall.

37 Hand, D. J. and C. C. Taylor. 1987. *Multivariate Analysis of Variance and Repeated Measures*. London: Chapman & Hall.

38 Harlow, L., S. Mulaik, and J. Steiger. 1997. *What If There Were No Significance Tests?* Mahwah, NJ: Lawrence Erlbaum.

39 Hartigan, J. A. and M. A. Wong. 1979. «A K-Means Clustering Algorithm». *Applied Statistics* 28: 100–108.

40 Hayton, J. C., D. G. Allen, and V. Scarpello. 2004. «Factor Retention Decisions in Exploratory Factor Analysis: A Tutorial on Parallel Analysis». *Organizational Research Methods* 7: 191–204.

41 Hsu, S., M. Wen, and M. Wu. 2009. «Exploring User Experiences as Predictors of MMORPG Addiction». *Computers and Education* 53: 990–999.

42 Johnson, J. 2000. «Multivariate Behavioral Research». 35: 1–19. ResearchGate.

43 2004. «Factors Affecting Relative Weights: The Influence of Sample and Measurement Error». *Organizational Research Methods* 7: 283–299.

44 Johnson, J. and J. LeBreton. 2004. «History and Use of Relative Importance Indices in Organizational Research». Organizational Research Methods 7: 238–257.

45 Kirk, R. 2008. *Statistics: An Introduction*, 5TH ed. California: Wadworth.

46 Kowarik, A. and M. Templ. 2016. «Imputation with R Package VIM». *Journal of Statistical Software*, 74: 1-16.

47 Koch, G. and S. Edwards. 1988. «Clinical Efficiency Trials with Categorical Data». *In Statistical Analysis with Missing Data*, 2nd ed., by R. J. A. Little and D. Rubin. Hoboken, NJ: John Wiley & Sons, 2002.

48 Kuhn, M. and K. Johnson. 2013. *Applied Predictive Modeling*[1]. New York: Springer.

[1] *Кун Джонсон*. Предиктивное моделирование на практике. Питер, 2019. ISBN: 978-5-4461-1039-1. – *Прим. перев.*

49 LeBreton, J. M and S. Tonidandel. 2008. «Multivariate Relative Importance: Extending Relative Weight Analysis to Multivariate Criterion Spaces». *Journal of Applied Psychology* 93: 329–345.

50 Lanz, B. 2015. *Machine Learning with R*[1], 2nd ed. Birmingham: Packt.

51 Licht, M. 1995. «Multiple Regression and Correlation». *In Reading and Understanding Multivariate Statistics*, edited by L. Grimm and P. Yarnold. Washington, DC: American Psychological Association, 19–64.

52 Little, R. J. A., and D. B. Rubin. (2002). *Statistical Analysis with Missing Data* (2nd ed.). New Jersey: John Wiley.

53 Mangasarian, O. L. and W. H. Wolberg. 1990. «Cancer Diagnosis via Linear Programming». *SIAM News*, 23: 1–18.

54 McCall, R. B. 2000. *Fundamental Statistics for the Behavioral Sciences*, 8th ed. New York: Wadsworth.

55 McCullagh, P. and J. Nelder. 1989. *Generalized Linear Models*, 2nd ed. Boca Raton, FL: Chapman & Hall.

56 Meyer, D., A. Zeileis, and K. Hornick. 2006. «The Strucplot Framework: Visualizing Multi-way Contingency Tables with vcd». *Journal of Statistical Software* 17 (3): 1–48. www.jstatsoft.org/v17/i03/paper.

57 Montgomery, D. C. 2007. *Engineering Statistics*. Hoboken, NJ: John Wiley & Sons.

58 Mooney, C. and R. Duval. 1993. *Bootstrapping: A Nonparametric Approach to Statistical Inference*. Monterey, CA: Sage.

59 Mulaik, S. 2009. *Foundations of Factor Analysis*, 2nd ed. Boca Raton, FL: Chapman & Hall.

60 Nenadic´, O. and M. Greenacre. 2007. «Correspondence Analysis in R, with Two- and Three-Dimensional Graphics: The ca Package». *Journal of Statistical Software* 20 (3). www.jstatsoft.org/v20/i03/paper.

61 Pinheiro, J. C. and D. M. Bates. 2000. *Mixed-Effects Models in S and S-PLUS*. New York: Springer.

62 Potvin, C., M. J. Lechowicz, and S. Tardif. 1990. «The Statistical Analysis of Ecophysiological Response Curves Obtained from Experiments Involving Repeated Measures». *Ecology* 71: 1389–1400.

63 Schafer, J. and J. Graham. 2002. «Missing Data: Our View of the State of the Art». *Psychological Methods* 7: 147–177.

64 Schlomer, G., S. Bauman, and N. Card. 2010. «Best Practices for Missing Data Management in Counseling Psychology». *Journal of Counseling Psychology* 57: 1–10.

65 Shah, A. 2005. «Getting Started with the boot Package in R for Statistical Inference». www.mayin.org/ajayshah/KB/R/documents/boot.html.

[1] *Ланц Бретт.* Машинное обучение на R: экспертные техники для прогностического анализа. Питер, 2020. ISBN: 978-5-4461-1512-9. – *Прим. перев.*

66 Shumway, R. H. and D. S. Stoffer. 2010. *Time Series Analysis and Its Applications*. New York: Springer.

67 Silva, R. B., D. F. Ferreirra, and D. A. Nogueira. 2008. «Robustness of Asymptotic and Bootstrap Tests for Multivariate Homogeneity of Covariance Matrices». *Ci nc. agrotec.* 32: 157–166.

68 Simon, J. 1997. «Resampling: The New Statistics». www.resample.com/intro-text-online/.

69 Snedecor, G. W. and W. G. Cochran. 1988. *Statistical Methods*, 8th ed. Ames, IA: Iowa State University Press.

70 Statnikov, A., C. F. Aliferis, D. P. Hardin, and I. Guyon. 2011. *A Gentle Introduction to Support Vector Machines in Biomedicine* (vol. 1: *Theory and Methods*). Hackensack, NJ: World Scientific Publishing.

71 Torgo, L. 2017. *Data Mining with R: Learning with Case Studies* (2nd ed.). Boca Raton, Florida: Chapman & Hall/CRC.

72 UCLA: Academic Technology Services, Statistical Consulting Group. 2009. «Repeated Measures Analysis with R». http://mng.bz/a9c7.

73 van Buuren, S. and K. Groothuis-Oudshoorn. 2011. «MICE: Multivariate Imputation by Chained Equations in R». *Journal of Statistical Software*, 45 (3), 1–67. http://mng.bz/3EH5.

74 Venables, W. N. and B. D. Ripley. 1999. *Modern Applied Statistics with S-PLUS*, 3rd ed. New York: Springer.

75 2000. *S Programming*. New York: Springer.

76 Westfall, P. H., Y. Hochberg, D. Rom, R. Wolfinger, and R. Tobias. 1999. *Multiple Comparisons and Multiple Tests Using the SAS System*. Cary, NC: SAS Institute.

77 Wickham, H. 2009a. *ggplot2: Elegant Graphics for Data Analysis*. New York: Springer.

78 2009b. «A Layered Grammar of Graphics». *Journal of Computational and Graphical Statistics* 19: 3–28.

79 2019. *Advanced R*, 2nd ed. Boca Raton, Florida: Chapman & Hall/CRC.

80 Wilkinson, L. 2005. *The Grammar of Graphics*. New York: Springer-Verlag.

81 Yu, C. H. 2003. «Resampling Methods: Concepts, Applications, and Justification». *Practical Assessment, Research & Evaluation* 8 (19). http://pareonline.net/getvn.asp?v=8&n=19.

82 Yu-Sung, S., A. Gelman, J. Hill, and M. Yajima. 2011. «Multiple Imputation with Diagnostics (mi) in R: Opening Windows into the Black Box». *Journal of Statistical Software* 45 (2). www.jstatsoft.org/v45/i02/paper.

83 Zuur, A. F., E. Ieno, N. Walker, A. A. Saveliev, and G. M. Smith. 2009. *Mixed Effects Models and Extensions in Ecology with R*. New York: Springer.

Предметный указатель

З

И

К

Л

М

Н

О

П

P

С

Т

У

A

B

C

D

E

F

G

H

I

J

K

L

Q

R

S