Мастицкий С. Э., Шитиков В. К.

Статистический анализ и визуализация данных с помощью R

ДМК Пресс
издательство
Москва, 2015

УДК 311:004.9R
ББК 60.6с515
М32

Мастицкий С. Э., Шитиков В. К.
М32 Статистический анализ и визуализация данных с помощью R. – М.: ДМК Пресс, 2015. – 496 с.: цв. ил.

ISBN 978-5-97060-301-7

Сегодня язык R является безусловным лидером среди свободно распространяемых систем статистического анализа. Ведущие университеты мира, аналитики крупнейших компаний и исследовательских центров регулярно используют R при проведении научно-технических расчетов и создании крупных информационных проектов. Широкое преподавание статистики на базе этой системы и всемерная поддержка научным сообществом обусловили то, что приведение скриптов кода на языке R постепенно становится общепризнанным стандартом как в журнальных публикациях, так и при неформальном общении ученых всего мира. Настоящая книга дополняет небольшую (пока) коллекцию работ по R на русском языке, обобщая и значительно расширяя совокупность методических сообщений, опубликованных ранее одним из авторов в блоге «R: Анализ и визуализация данных» (http://r-analytics.blogspot.com).

Книга адресована студентам, аспирантам, а также молодым и состоявшимся ученым, желающим освоить классические и современные методы анализа данных с использованием языка R.

УДК 311:004.9R
ББК 60.6с515

ISBN 978-5-97060-301-7

Посвящаю эту книгу своим родителям.
Сергей Мастицкий

Всем заинтересованным читателям,
без которых книги вообще не имеют смысла...
Владимир Шитиков

Содержание

Предисловие 10

Глава 1. Основные компоненты статистической среды R 13

1.1. История возникновения и основные принципы организации среды R.....13
1.2. Работа с командной консолью.....17
1.3. Работа с меню R Commander.....20
1.4. Объекты, пакеты, функции, устройства.....24

Глава 2. Описание языка R 31

2.1. Типы данных.....31
2.2. Векторы и матрицы.....32
2.3. Факторы.....38
2.4. Списки и таблицы данных.....40
Заполнение пустых значений.....45
Сортировка таблиц.....46
Объединение таблиц.....46
2.5. Импортирование данных в R.....47
2.6. Представление дат и времени. Временные ряды.....51
Форматы представления дат и времени.....51
Вычисления с датами и временем.....52
Преобразование текстовых переменных в машинный формат времени.....53
Временные ряды.....54
2.7. Организация вычислений: функции, ветвления, циклы.....56
Написание собственных функций.....57
Условия и циклы.....59
2.8. Векторизованные вычисления в R.....61

Глава 3. Базовые графические возможности R 70

3.1. Функция `plot()` и ее параметры.....70
Управляющие параметры функции `plot()`.....73
Общие аргументы графических функций.....74
3.2. Гистограммы, функции ядерной плотности и функция `cdplot()`.....79
3.3. Диаграммы размахов.....87
3.4. Круговые и столбиковые диаграммы.....91

3.5. Диаграммы Кливленда и одномерные диаграммы рассеяния 99
3.6. Категоризованные графики 107

Глава 4. Описательная статистика, подгонка распределений и смежные задачи 114

4.1. Базовые функции для расчета параметров описательной статистики 114
4.2. `summary()` и функции из дополнительных пакетов 118
4.3. Анализ выбросов 121
4.4. Заполнение пропущенных значений в таблицах данных 125
4.5. Воспроизводимость результатов при использовании генератора случайных чисел 131
4.6. Законы распределения вероятностей, реализованные в R 134
4.7. Подбор закона и параметров распределения в R 136
4.8. Проверка на нормальность распределения 144
Графические способы 145
Формальные тесты 148

Глава 5. Классические методы статистики 151

5.1. Гипотеза о равенстве средних двух генеральных совокупностей 151
Одновыборочный t-критерий 151
Сравнение двух независимых выборок 153
Сравнение двух зависимых выборок 155
5.2. Ранговый критерий Уилкоксона-Манна-Уитни 157
Одновыборочный критерий Уилкоксона 157
Сравнение двух независимых выборок 158
Сравнение двух зависимых выборок 159
5.3. Рандомизация, бутстреп и оценка статистической мощности (на примере двухвыборочного t-критерия) 161
5.4. Гипотеза об однородности дисперсий 168
Проверка однородности дисперсии в двух группах 168
Проверка однородности дисперсии в нескольких группах 169
5.5. Введение в дисперсионный анализ 171
Постановка задачи 171
Две оценки генеральной дисперсии в дисперсионном анализе 174
Выполнение дисперсионного анализа в R 176
Двухфакторный дисперсионный анализ 176
5.6. Оценка корреляции двух случайных величин 180
5.7. Критерий хи-квадрат 184
Критерий хи-квадрат для таблиц сопряженности размером 2×2 184

Критерий хи-квадрат для таблиц сопряженности размером больше 2×2 187
5.8. Точный тест Фишера. Критерии Мак-Немара и Кохрана-Мантеля-Хензеля 187
Точный тест Фишера 187
Критерий Мак-Немара 190
Критерий Кохрана-Мантеля-Хензеля для таблиц сопряженности размером 2×2×K 193
5.9. Оценка статистической мощности при сравнении частот 197

Глава 6. Дисперсионный анализ 203

6.1. Протокол разведочного анализа данных 203
Выявление точек-выбросов 204
Проверка однородности групповых дисперсий 205
Проверка на нормальность распределения 206
Выявление избыточного числа нулевых значений 207
Выявление коллинеарности 207
Выявление формы связи между переменными 210
Выявление взаимодействий между предикторами 212
Влияние пространственно-временных факторов на анализируемую переменную 216
6.2. Дисперсионный анализ как линейная модель 219
6.3. Структура модельных объектов дисперсионного анализа 227
6.4. Оценка адекватности модели дисперсионного анализа 230
Проверка исходных предположений общей линейной модели 230
Проверка условия нормальности распределения 231
Проверка условия однородности групповых дисперсий 234
Что делать, когда однофакторный дисперсионный анализ неприменим? 237
6.5. Дисперсионный анализ по Краскелу-Уоллису 239
6.6. Модели двух- и многофакторного дисперсионного анализа 241
Синтаксис объекта «формула» 242
Выполнение двухфакторного дисперсионного анализа при помощи функции `lm()` 244
Порядок перечисления предикторов в формуле модели 246
Многофакторный дисперсионный анализ 248
6.7. Контрасты в линейных моделях, содержащих категориальные предикторы 249
Основные понятия 250
Контрасты комбинаций условий (treatment contrasts) 252
Контрасты сумм (sum contrasts) 254
Контрасты Хелмерта 255

Контрасты, задаваемые пользователем ... 257
6.8. Проблема множественных проверок статистических гипотез ... 258
Поправка Бонферрони ... 261
Метод Холма ... 262
Метод Беньямини-Хохберга ... 263
Метод Беньямини-Йекутили ... 266
6.9. Апостериорные сравнения групповых средних ... 267
Критерий Тьюки ... 268
Методы множественных проверок гипотез, реализованные в пакете `multcomp` ... 271

Глава 7. Регрессионные модели зависимостей между количественными переменными ... 279

7.1. О понятии «статистическая модель» ... 279
Пример простейшей статистической модели ... 279
Исследование свойств статистических моделей имитационными методами ... 282
Пример модели с одним количественным предиктором ... 287
Назначение регрессионных моделей ... 289
7.2. Простая линейная регрессия: каков возраст Вселенной? ... 290
Модель для оценки постоянной Хаббла ... 291
Доверительные интервалы ... 293
Оценка неопределенности в отношении параметров линейной регрессии ... 295
Оценка «качества» регрессионной модели ... 301
7.3. Стандартные методы диагностики линейных моделей ... 304
Проверка допущений в отношении остатков модели ... 304
Проверка адекватности структуры систематической части модели ... 308
Встроенные диагностические графики ... 313
Выявление необычных и влиятельных наблюдений ... 314
7.4. Модели регрессии при разных видах функции потерь ... 325
Два типа регрессионных моделей ... 325
Робастные процедуры ... 329
7.5. Критерии выбора моделей оптимальной сложности ... 331
7.6. Полиномиальные и нелинейные модели регрессии ... 335
Полиномиальная регрессия ... 335
Нелинейная регрессия ... 338
7.7. Модель множественной регрессии и выбор ее спецификации ... 344
Полная модель и обоснование необходимости ее оптимизации ... 345
Пошаговые алгоритмы селекции переменных ... 347

Построение «всех возможных моделей» ... 348
Пошаговогое включение предикторов в сочетании с перекрестной проверкой ... 350
7.8. Диагностика моделей множественной регрессии ... 353
Сравнение нескольких альтернативных моделей ... 353
Диагностика допущений в отношении остатков модели ... 354
Учет нелинейного характера влияния предикторов на отклик ... 359
7.9. Регуляризация множественной регрессии ... 361
Гребневая регрессия ... 362
Лассо-регрессия Тибширани ... 364
7.10. Регрессия на главные компоненты ... 366
7.11. Сравнение эффективности различных моделей при прогнозировании ... 372
Формирование исходных данных для построения моделей ... 372
Общая линейная модель и ее тестирование на проверочной выборке ... 374
Выбор информативного комплекса предикторов ... 375
Модели с использованием регуляризации ... 377
Регрессия на главные компоненты ... 380
Результаты и некоторые выводы ... 382

Глава 8. Обобщенные, структурные и иные модели регрессии ... 384

8.1. Модели сглаживания ... 384
Ядерная модель сглаживания ... 389
Сплайны ... 393
8.2. Обобщенные модели регрессии ... 395
8.3. Модели пробит- и логит-регрессии ... 399
Пробит-регрессия для моделирования зависимости «доза–эффект» ... 400
Логистическая регрессия ... 407
8.4. Пример использования обобщенных моделей для оценки экологической толерантности ... 411
Модели с нормально распределенным откликом ... 412
Модели с бинарным откликом ... 416
8.5. Ковариационный анализ ... 419
8.6. Модели со смешанными эффектами для иерархически организованных данных ... 424
Основные идеи ... 424
Пример с морскими животными: несколько частных моделей ... 426
8.7. Индуктивные модели (метод группового учета аргументов) ... 433

8.8. Моделирование структурными уравнениями 440

Глава 9. Пространственный анализ и создание картограмм 451

9.1. Простая карта: использование растрового рисунка и расчет расстояний 451
Использование географических расстояний в статистическом анализе 452
Расчет расстояния между объектами по их географическим координатам 457
9.2. Анализ пространственного размещения точек 460
9.3. Использование сервисов картографической системы Google Maps 466
9.4. Создание картограмм при помощи R 469
Шейп-файлы 470
Функция `spplot()` из пакета `sp` 474
Создание картограмм при помощи пакета `ggplot2` 478

Библиография и интернет-ресурсы 484

Основные литературные ссылки по тексту книги 484
Литература по R 484
Общеметодическая литература по статистическому анализу 485
Библиографический указатель литературы по R 485
Рекомендуемые интернет-ресурсы 494
Русскоязычные ресурсы 494
Англоязычные ресурсы 495

В целях природы обуздания,
Чтобы рассеять незнания тьму,
Берем картину мироздания
И тупо смотрим, что к чему....

А. и Б. Стругацкие.
«Понедельник начинается в субботу»

ПРЕДИСЛОВИЕ

Одним из основных инструментов познания мира является обработка данных, получаемых человеком из различных источников. Суть современного статистического анализа состоит в интерактивном процессе, состоящем из исследования, визуализации и интерпретации потоков поступаемой информации.

История последних 50 лет – это и история развития технологии анализа данных. Один из авторов этой книги с умилением вспоминает конец 60-х годов и свою первую программу расчета парной корреляции, которая набиралась металлическими штыречками на «операционном поле» из 150 ячеек персональной ЭВМ «Промінь-2» весом более 200 кг. В наше время высокопроизводительные компьютеры и доступное программное обеспечение позволяют реализовать полный цикл информационно-технологического процесса, состоящего, в общем случае, из следующих шагов:

- доступ к обрабатываемым данным (их загрузка из разных источников и комплектация совокупности взаимосвязанных исходных таблиц);
- редактирование загруженных показателей (замена или удаление пропущенных значений, преобразование признаков в более удобный вид);
- аннотирование данных (чтобы помнить, что представляет собой каждый их фрагмент);
- получение общих сведений о структуре данных (вычисление описательных статистик);
- графическое представление данных и результатов вычислений в понятной информативной форме (одна картинка на самом деле иногда стоит тысячи слов);
- моделирование данных (математическое описание зависимостей и тестирование статистических гипотез);
- оформление результатов (подготовка таблиц и диаграмм приемлемого публикационного качества).

В условиях, когда в распоряжении пользователя имеются десятки пакетов прикладных программ, актуальна проблема выбора (иногда трагичная, если вспомнить «буриданова осла»): какое программное обеспечение анализа данных следует предпочесть для своей практической работы? Здесь обычно принимаются во внимание специфика решаемой задачи, эффективность настройки алгоритмов обработки, издержки на покупку программ, а также вкусы и личные предпочте-

ния исследователя. При этом, например, шаблонная Statistica с ее механическим комплексом кнопок меню далеко не всегда может удовлетворить творческого исследователя, предпочитающего самостоятельно контролировать ход вычислительного процесса. Комбинировать различные типы анализа, иметь доступ к промежуточным результатам, управлять стилем отображения данных, добавлять собственные расширения программных модулей и оформлять итоговые отчеты в необходимом виде позволяют коммерческие вычислительные системы, включающие высокоуровневые средства командного языка, такие как Matlab, SPSS и др. Прекрасной альтернативой им является бесплатная программная среда R, являющаяся современной и постоянно развивающейся статистической платформой общего назначения.

Сегодня R является безусловным лидером среди свободно распространяемых систем статистического анализа, о чем говорит, например, тот факт, что в 2010 году система R стала победителем ежегодного конкурса открытых программных продуктов Bossie Awards в нескольких номинациях. Ведущие университеты мира, аналитики крупнейших компаний и исследовательских центров регулярно используют R при проведении научно-технических расчетов и создании крупных информационных проектов. Широкое преподавание статистики на базе пакетов этой среды и всемерная поддержка научным сообществом обусловили то, что приведение скриптов R постепенно становится общепризнанным «стандартом» как в журнальных публикациях, так и при неформальном общении ученых всего мира.

Главным препятствием для русскоязычных пользователей при освоении R, безусловно, является то, что почти вся документация по этой среде существует на английском языке. Лишь с 2008 года усилиями А. В. Шипунова, Е. М. Балдина, С. В. Петрова, И. С. Зарядова, А. Г. Буховца, П. А. Волковой и других энтузиастов появились методические пособия и книги на русском языке (ссылки на них можно найти в списке литературы в конце этой книги; там же представлены и ссылки на образовательные ресурсы, авторами которых делается посильный вклад в продвижение R среди русскоязычных пользователей). Настоящая книга дополняет эту небольшую (пока) коллекцию работ по R на русском языке, обобщая совокупность методических сообщений, опубликованных одним из авторов с 2011 года в блоге «R: Анализ и визуализация данных» (http://r-analytics.blogspot.com). Нам показалась целесообразной идея представить для удобства читателей весь этот несколько разобщенный материал в концентрированной форме, а также расширить некоторые разделы для полноты изложения.

В первых трех главах содержатся подробные указания по работе с интерактивными компонентами R, детальное описание языка и базовых графических возможностей среды. Эта часть книги вполне доступна новичкам в области программирования, хотя читатель, уже знакомый с языком R, может найти там интересные фрагменты кода или использовать приведенные описания графических параметров как справочное пособие.

В последующих главах (4–8) приведено описание распространенных процедур обработки данных и построения статистических моделей, которое иллюстрировано несколькими десятками примеров. Все эти главы включают краткие описания соответствующих алгоритмов анализа, основные полученные в примерах резуль-

таты и их возможную интерпретацию. Мы старались, по возможности, обойтись без злоупотребления «ритуальными» словооборотами, характерными для многочисленных руководств по статистике, цитирования общеизвестных теорем и приведения многоэтажных расчетных формул. Акцент делался в первую очередь на практическое применение – на то, чтобы читатель, руководствуясь прочитанным, мог проанализировать свои данные и изложить результаты коллегам.

Материал выстроен по мере усложнения. Так, главы 4 и 5 ориентированы на читателя, интересующегося статистикой лишь в объеме начального университетского курса. В главах 6 и 7 в рамках единой теории общих линейных моделей представлены дисперсионный и регрессионный анализы и приведены различные алгоритмы исследования и структурной идентификации моделей. Глава 8 посвящена некоторым современным методам построения и анализа обобщенных линейных и иных типов моделей.

Поскольку неизменный интерес у исследователей вызывают пространственный анализ и визуализация данных на географических картах и схемах, в главе 9 приведены некоторые примеры соответствующих приемов.

Отдельно стоит упомянуть обозначения, принятые в книге. Все команды языка R выделены моноширинным шрифтом. При этом имена функций дополнительно выделены полужирным шрифтом, как в этом примере: **mean**(**c**(1, 2, 3)). При упоминании каких-либо функций в тексте мы всегда добавляем к их именам круглые кавычки, что позволяет отличать функции от других объектов R (например, **summary**()). Результаты вычислений, полученные при использовании той или иной функции, выделены шрифтом синего цвета. Наконец, комментарии к коду представлены наклонным шрифтом серого цвета и начинаются со знака #:

```
# Расчет среднего значения:
mean(c(1, 2, 3))
[1] 2
```

Мы адресуем эту книгу студентам, аспирантам, а также молодым и состоявшимся ученым, желающим освоить анализ и визуализацию данных с использованием среды R. Мы надеемся, что по окончании чтения этого руководства вы получите некоторое представление о том, как работает R, где можно получить дальнейшую информацию, а также как справиться с широким спектром задач анализа данных.

Файлы со скриптами кода R по всем главам книги и таблицы исходных данных, необходимые для выполнения примеров, свободно доступны для скачивания с GitHub-репозитория https://github.com/ranalytics/r-tutorials, а также с сайта Института экологии Волжского бассейна РАН по ссылке http://www.ievbras.ru/ecostat/Kiril/R/Scripts.zip.

Мы будем благодарны Вам, Читатель, за любые замечания и пожелания касательно этой работы, которые Вы можете направлять по электронной почте rtutorialsbook@gmail.com.

Сергей Мастицкий, Лондон
Владимир Шитиков, Тольятти
май 2015 года

Глава 1

Основные компоненты статистической среды R

1.1. История возникновения и основные принципы организации среды R

Система статистического анализа и визуализации данных R состоит из следующих основных частей:

- языка программирования высокого уровня R, позволяющего одной строкой реализовать различные операции с объектами, векторами, матрицами, списками и т. д.;
- большого набора функций обработки данных, собранных в отдельные так называемые «пакеты»;
- развитой системой поддержки, включающей обновление компонентов среды, интерактивную помощь и различные образовательные ресурсы, предназначенные как для начального изучения R, так и для последующих консультаций по возникающим затруднениям.

Начало пути относится к 1993 г., когда двое молодых новозеландских ученых Росс Ихака (Ross Ihaka) и Роберт Джентльмен (Robert Gentleman) анонсировали свою новую разработку, которую назвали R. Они взяли за основу язык программирования развитой коммерческой системы статистической обработки данных S-PLUS и создали его бесплатную свободную реализацию, отличающуюся от своего прародителя легко расширяемой модульной архитектурой. В скором времени возникла распределенная система хранения и распространения пакетов к R, известная под аббревиатурой «CRAN» (Comprehensive R Archive Network – http://cran.r-project.org), основная идея организации которой – постоянное расширение, коллективное тестирование и оперативное распространение прикладных средств обработки данных.

Оказалось, что такой продукт непрерывных и хорошо скоординированных усилий мощного «коллективного разума» тысяч бескорыстных разработчиков-интеллектуалов оказался значительно эффективнее коммерческих статистических программ, стоимость лицензии на которые может составлять несколько тысяч долларов. Поскольку R является любимым языком профессиональных статистиков, все последние достижения статистической науки очень быстро становятся

доступными для пользователей R во всем мире в виде дополнительных пакетов. Ни одна коммерческая система статистического анализа так быстро сегодня не развивается. У R есть многочисленная армия пользователей, которые сообщают авторам дополнительных пакетов и самой системы R об обнаруженных ошибках, которые оперативно исправляются.

Язык вычислений R, хотя и требует определенных усилий для своего освоения, недюжинных поисковых навыков и энциклопедической памяти, позволяет оперативно выполнить расчеты, по своему разнообразию практически «столь же неисчерпаемые, как атом». По состоянию на конец февраля 2015 г., энтузиастами со всего мира было создано 7336 дополнительных библиотек для R, включающих 149 688 функций (см. http://www.rdocumentation.org), которые существенно расширяют базовые возможности системы. Очень сложно представить какой-либо класс статистических методов, который еще не реализован сегодня в виде пакетов R, включая, разумеется, весь «джентльменский набор»: линейные и обобщенные линейные модели, нелинейные регрессионные модели, планирование эксперимента, анализ временных рядов, классические параметрические и непараметрические тесты, байесовская статистика, кластерный анализ и методы сглаживания. При помощи мощных средств визуализации результаты анализа можно обобщать в виде всевозможных графиков и диаграмм. Кроме традиционной статистики, разработанный функционал включает большой набор алгоритмов численной математики, методов оптимизации, решения дифференциальных уравнений, распознавания образов и др. Свои специфические методы обработки данных могут обнаружить в составе пакетов R генетики и социологи, лингвисты и психологи, химики и медики, специалисты по ГИС- и веб-технологиям.

«Фирменная» документация по R весьма объемна и далеко не всегда толково написана (по странной традиции англоязычной литературы, слишком много слов расходуется на описание тривиальных истин, тогда как важные моменты пробегаются скороговоркой). Однако в дополнение к этому ведущими мировыми издательствами (Springer, Cambridge University Press и Chapman & Hall/CRC) или просто отдельными коллективами энтузиастов выпущено огромное число книг, описывающих различные аспекты анализа данных в R (см., например, список литературы на сайте «Энциклопедия психодиагностики», http://psylab.info/R:Литература). Кроме того, существует несколько активно действующих международных и российских форумов пользователей R, где любой может попросить о помощи в возникшей проблеме. В списке литературы мы приводим пару сотен книг и интернет-ссылок, на которые советуем обратить особое внимание в ходе изучения R.

Непосредственное обучение практической работе в R состоит из *а*) освоения конструкций языка R и знакомства с особенностями вызова функций, выполняющих анализ данных, и *б*) приобретения навыков работы с программами, реализующими специфические методы анализа и визуализации данных.

Вопрос выбора средств пользовательского интерфейса R неоднозначен и сильно зависит от вкусов пользователей. Единого мнения нет даже у авто-

ритетных специалистов. Одни считают, что нет ничего лучше стандартного консольного интерфейса R. Другие полагают, что для удобной работы стоит инсталлировать какую-либо из имеющихся интегрированных сред разработки (IDE) с богатым набором кнопочных меню. Например, отличным вариантом является бесплатная интегрированная среда разработки RStudio (http://www.rstudio.com). Ниже мы остановимся на описании консольного варианта и на работе с R Commander, но дальнейшим исканиям читателя может помочь обзор различных версий IDE, представленный в приложении к книге А. Шипунова с соавторами (2014).

Один из R-экспертов, Джозеф Рикерт, считает, что процесс изучения R можно разделить на следующие этапы (подробнее см. его статью на сайте http://bit.ly/1FlOx2E):

1. Знакомство с общими принципами культуры R-сообщества и программной среды, в которой разрабатывался и функционирует язык R. Посещение основных и вспомогательных ресурсов и освоение хорошего вводного учебника. Инсталляция R на компьютере пользователя и выполнение первых тестовых скриптов.
2. Считывание данных из стандартных файлов операционной системы и уверенное использование R-функций для выполнения ограниченного набора привычных пользователю процедур статистического анализа.
3. Использование базовых структур языка R для написания простых программ. Написание собственных функций. Ознакомление со структурами данных, с которыми может работать R, и более сложными возможностями языка. Работа с базами данных, веб-страницами и внешними источниками данных.
4. Написание сложных программ на языке R. Самостоятельная разработка и глубокое понимание структуры объектов так называемых S3- и S4-классов.
5. Разработка профессиональных программ на языке R. Самостоятельное создание дополнительных модулей для R.

Большинство рядовых пользователей R останавливаются на стадии 3, так как полученных к этому времени знаний им вполне достаточно для выполнения статистических задач по профилю их основной профессиональной деятельности. Примерно в этом объеме мы и приводим описание языка R в рамках настоящего руководства.

Установить и настроить базовую комплектацию статистической среды R весьма просто. На момент написания этой книги актуальной была версия R 3.1.2 для Windows (доступны также дистрибутивы для всех других распространенных операционных систем). Скачать дистрибутив системы вместе с базовым набором из 29 пакетов можно совершенно бесплатно с основного сайта проекта http://cran.r-project.org или его русского «зеркала» http://cran.gis-lab.info. Процесс инсталляции системы из скачанного дистрибутива затруднений не вызывает и не требует никаких особых комментариев.

Для удобства хранения скриптов, исходных данных и результатов расчетов стоит выделить на пользовательском компьютере специальный рабочий каталог. Весьма нежелательно использовать в названии рабочего каталога символы кириллицы.

Путь к рабочему каталогу и некоторые другие опции настроек целесообразно разместить, изменив в любом текстовом редакторе системный файл C:\Program Files\R\R-3.1.2\etc\Rprofile.site (на вашем компьютере он может иметь иной адрес). В представленном ниже примере модифицированные строки отмечены синим цветом. Помимо указания рабочего каталога, эти строки определяют ссылку на российский источник загрузки пакетов R и автоматический запуск R Commander.

Листинг файла Rprofile.site

```
# Все, что следует за символом комментария "#", средой игнорируется
# options(papersize="a4")
# options(editor="notepad")
# options(pager="internal")

# установить тип отображения справочной информации
# options(help_type="text")
  options(help_type="html")

# установить место расположения локальной библиотеки
# .Library.site <- file.path(chartr("\\", "/", R.home()), "site-library")

# При загрузке среды запустить меню R Commander
# Поставить знаки #, если запуск Rcmdr не нужен
 local({
 old <- getOption("defaultPackages")
 options(defaultPackages = c(old, "Rcmdr"))
 })

# Определить зеркало CRAN
 local({r <- getOption("repos")
       r["CRAN"] <- "http://cran.gis-lab"
       options(repos = r)})

# Определить путь к рабочему каталогу (любой иной на вашем компьютере)
setwd("D:/R/Process/Resampling")
```

Что касается «хорошего вводного учебника», то любые наши рекомендации неизбежно будут носить субъективный оттенок. Тем не менее следует упомянуть официальное введение в R (Venables & Smith, 2014) и книгу Kabacoff (2011), отчасти еще и потому, что имеется их русский перевод. Отметим также традиционное «наставление для чайников» Meys & Vries (2012) и руководства Dalgaard (2008) и Lam (2010). Из русскоязычных вводных книг наиболее полными являются работы И. Зарядова (2010а, б) и А. Шипунова с соавторами (2014).

1.2. Работа с командной консолью

Статистическая среда R выполняет любой набор осмысленных инструкций языка R, содержащихся в файле скрипта или представленных последовательностью команд, задаваемых с консоли. Работа с консолью может показаться трудной для современных пользователей, привыкших к кнопочным меню, поскольку надо запоминать синтаксис отдельных команд. Однако после приобретения уже некоторых первичных навыков окажется, что многие процедуры обработки данных можно выполнять быстрее и с меньшим трудом, чем, предположим, в той же программе Statistica.

Консоль R представляет собой диалоговое окно, в котором пользователь вводит команды и результаты их выполнения. Это окно возникает сразу при запуске среды (например, после клика мышью на ярлыке R на рабочем столе). Кроме того, стандартный графический пользовательский интерфейс R (RGui) включает окно редактирования скриптов и всплывающие окна с графической информацией (рисунками, диаграммами и прочим).

В командном режиме R может работать, например, как обычный калькулятор (рис. 1). Справа от символа приглашения («*prompt*») > пользователь может ввести произвольное арифметическое выражение, нажать клавишу **Enter** и тут же получить результат. Например, во второй команде на рис. 1 мы использовали функции факториала и синуса, а также встроенное число π. Результаты, полученные в текстовой форме, можно выделить мышью и скопировать через буфер обмена в любой текстовый файл (например, документ Word).

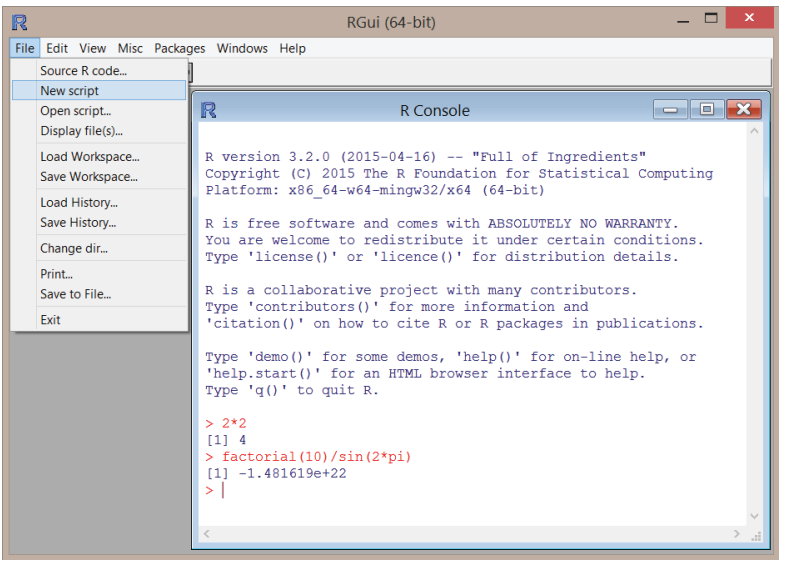


Рисунок 1

При работе с использованием RGui мы рекомендуем во всех случаях создавать скрипт (то есть файл с последовательностью команд языка R, выполняющей определенные действия). Как правило, это обычный текстовый файл с любым именем и, желательно, расширением `.r`, который можно создавать и редактировать обычным редактором типа «Блокнот». Если этот файл существует, его лучше всего поместить в рабочий каталог, и тогда после запуска R и выбора пункта меню **Файл** > **Открыть скрипт** (File > Open script) содержимое этого файла появится в окне **Редактор R** (R Editor). Выполнить последовательность команд скрипта можно из пункта меню **Правка** > **Запустить все** (Edit > Run all). Можно также выделить мышью осмысленный фрагмент из любого места подготовленного скрипта (от имени одной переменной до всего содержимого) и осуществить запуск этого блока на выполнение. Это можно сделать четырьмя возможными способами: из основного и контекстного меню, комбинацией клавиш **Ctrl**+**R** или кнопкой на панели инструментов.

Рисунок 2 описывает следующие действия:

- из бесплатного интернет-источника *Global Administrative Areas* (http://gadm.org/) был скачан R-объект `gadm` с данными по территориальному делению Республики Беларусь;
- переменная `NAME_1`, содержащая названия областных центров, преобразована в объект класса «фактор»;
- с использованием функции **`spplot`**`()` из пакета `sp` в графическое окно программы выведена административная карта, которую можно средствами

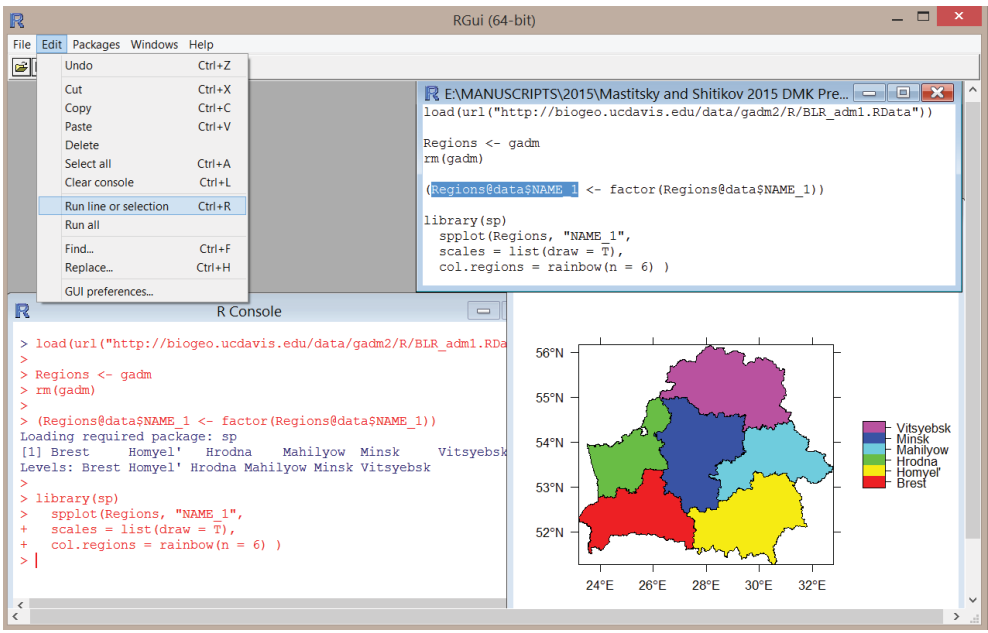


Рисунок 2

меню скопировать в буфер обмена или сохранить как стандартный мета- или растровый графический файл.

Подробнее смысл отдельных команд мы рассмотрим в последующих разделах, а здесь обратим внимание на то, что, выделив в скрипте и запустив на выполнение комбинацию символов `Regions@data`, мы получим в окне консоли весь набор данных `data` по объекту `gadm`, а команда, составленная из выделенных символов `Regions@data$NAME_1`, даст нам список наименований административных центров.

Таким образом, **Редактор R** позволяет легко выполнить навигацию по скрипту, редактирование и выполнение любой комбинации команд, поиск и замену определенных частей кода. Упомянутая выше интегрированная среда разработки RStudio позволяет дополнительно выполнять подсветку синтаксиса кода, его автоматическое завершение, «упаковку» последовательностей команд в функции для их последующего использования, работу с документами Sweave или TeX и другие операции, которые будут полезны продвинутому пользователю.

R обладает обширными встроенными справочными материалами, которые можно получить непосредственно в RGui. Если подать с консоли команду **`help.start()`**, то в вашем интернет-браузере откроется страница, предоставляющая доступ ко всем внутренним справочным ресурсам: основным руководствам, авторским материалам, ответам на распространенные вопросы, истории внесенных в R изменений и т. д. (рис. 3).

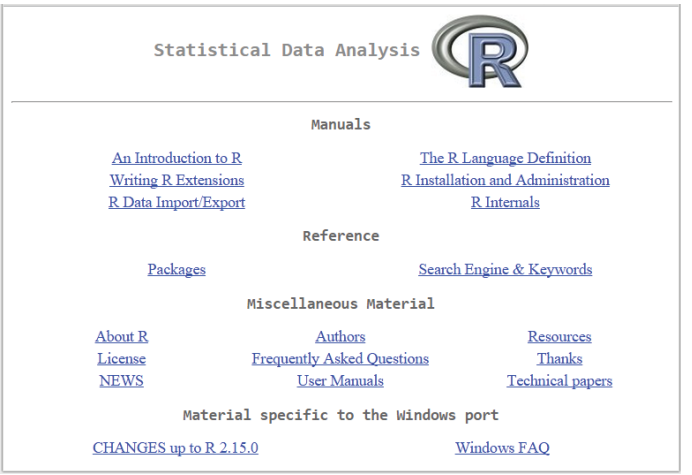


Рисунок 3

Справку по отдельным функциям можно получить с использованием следующих команд:

- **`help`**`("foo")` или `?foo` – справка по функции `foo` (кавычки необязательны);
- **`help.search`**`("foo")` или `??foo` – поиск всех справочных файлов, содержащих `foo`;

- **example**("foo") – примеры использования функции foo;
- **RSiteSearch**("foo") – поиск ссылок в онлайн-руководствах и архивах рассылок;
- **apropos**("foo", mode = "function") – список всех функций, в названии которых встречается foo;
- **vignette**("foo") – список руководств по теме foo.

1.3. Работа с меню R Commander

Удобным средством освоения вычислений в R для начинающего пользователя является R Commander – платформонезависимый графический интерфейс с кнопочным меню, реализованный в пакете Rcmdr. Он позволяет осуществить большой комплект процедур статистического анализа, не прибегая к предварительному заучиванию функций на командном языке, однако невольно способствует этому, поскольку отображает все выполняемые инструкции в консоли программы.

Установить Rcmdr, как и любые другие расширения, можно из меню **Пакеты** > **Установить пакет** (Packages > Install package(s)), но лучше выполнив команду:

```
install.packages("Rcmdr", dependencies = TRUE),
```

где включение опции dependencies вызовет гарантированную установку полного комплекта остальных пакетов, которые могут потребоваться при обработке данных через меню Rcmdr. Обратите внимание: выполнение приведенной команды предполагает, что ваш компьютер подключен к Интернету.

Запуск R Commander происходит при загрузке пакета Rcmdr через меню **Пакеты** > **Включить пакет** (Packages > Load package) или командой **library**(Rcmdr).

Если по какой-то причине было принято решение анализировать данные исключительно с помощью R Commander, то для автоматической загрузки этой графической оболочки при запуске R необходимо отредактировать файл Rprofile.site (см. раздел 1.1).

Работу в R Commander рассмотрим на примере корреляционного анализа данных по уровню инвазированности двустворчатого моллюска *Dreissena polymorpha* (http://bit.ly/1C0WvOW) инфузорией *Conchophthirus acuminatus* в трех озерах Беларуси (Mastitsky, 2013, http://bit.ly/1wsDKm4). В таблице с исходными данными, которую скачаем с сайта *figshare* (http://bit.ly/1wMGDOQ), нас будут интересовать две переменные: длина раковины моллюска (ZMlength, мм) и число обнаруженных в моллюске инфузорий (CAnumber). Подробно этот пример будет рассмотрен в главах 4 и 5, поэтому здесь мы не будем детально останавливаться на смысле анализа, а сосредоточимся на технике работы с Rcmdr.

Первый этап – загрузка нового набора данных, и мы выбираем из меню **Данные** > **Импорт данных… из URL** (Data > Import data ... from URL) (рис. 4).

Далее определяем во всплывающих окнах режим загрузки данных и адрес ссылки в Интернете. Нетрудно заметить, что те же данные мы могли легко загрузить

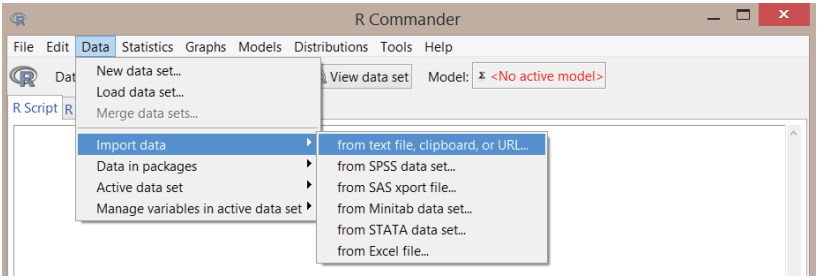
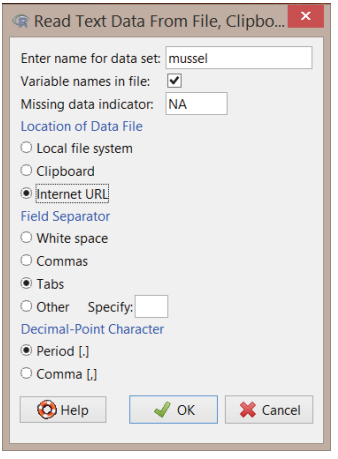
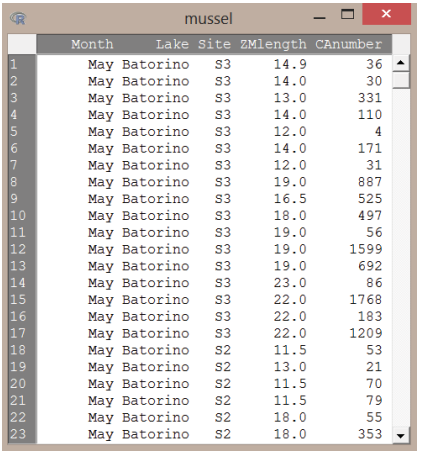


Рисунок 4

из локального текстового файла, книги Excel или таблицы базы данных. Чтобы убедиться в том, что наши данные загружены верно (и при необходимости их отредактировать), нажимаем кнопку **Посмотреть данные** (View data set) (рис. 5).



Окно спецификации данных



Фрагмент загруженной таблицы

Рисунок 5

На втором этапе в меню **Статистики** > **Итоги** (Statistics > Summaries) выбираем **Корреляционный тест** (Correlation test) (рис. 6).

Выделяем пару необходимых переменных, жмем **OK** и в окне вывода получаем коэффициент корреляции Пирсона (R = 0.467), уровень достигнутой статистической значимости (`p-value < 2.2e-16`) и 95%-ные доверительные пределы (рис. 7). Полученные результаты легко скопировать из окна вывода через буфер обмена.

Теперь получим графическое изображение корреляционной зависимости. В меню **Графики** (Graphs) выберем **Точечный график** (Scatterplot) зависимости `CAnumber` от `ZMlength` и снабдим ее краевыми диаграммами размахов, линией линейного тренда по методу наименьших квадратов (зеленым цветом) и сглаживающей линией по методу локальной регрессии (красным цветом), представленной с до-

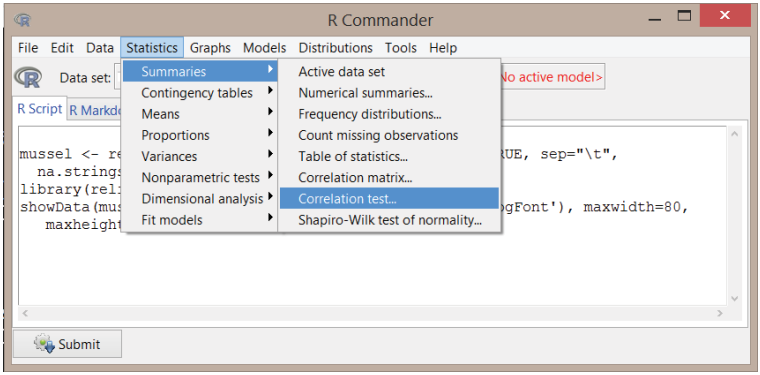
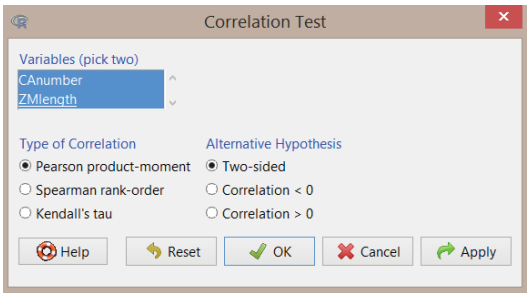


Рисунок 6




```
Pearson's product-moment correlation
data:  mussel$CAnumber and
mussel$ZMlength
t = 11.4964, df = 474
p-value < 2.2e-16
alternative hypothesis: true correlation
is not equal to 0
95 percent confidence interval:
 0.3935877 0.5343949
sample estimates:
     cor
0.466946
```

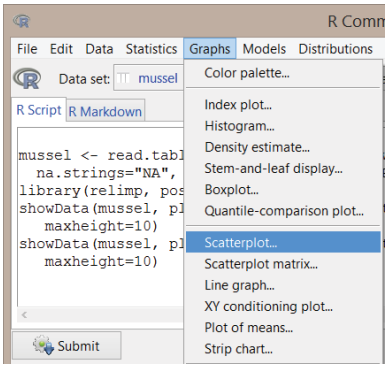
Окно спецификации вычислений **Полученные результаты**

Рисунок 7

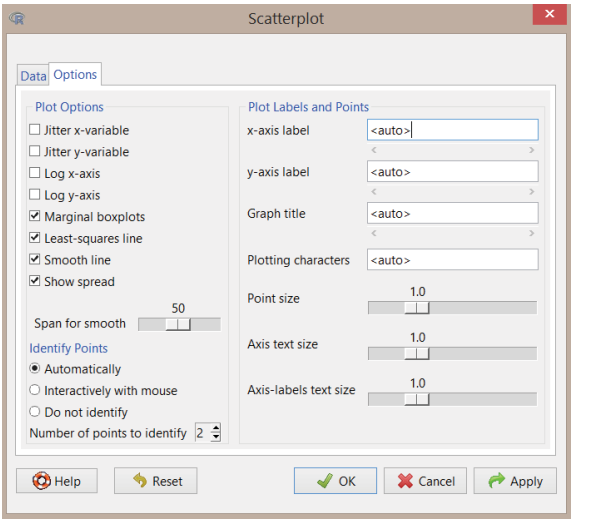
верительной областью (пунктир). Для каждого из трех озер (переменная Lake) экспериментальные точки будут представлены разными символами (рис. 8–9).

При нажатии кнопок меню R Commander в консоли R появляются эквивалентные инструкции языка R. В нашем случае они имеют следующий вид:

```
mussel <- read.table("http://bit.ly/1wMGDOQ", header=TRUE, sep="\t",
                     na.strings="NA", dec=".", strip.white=TRUE)
cor.test(mussel$CAnumber, mussel$ZMlength,
         alternative="two.sided", method="pearson")
scatterplot(CAnumber ~ ZMlength | Lake, reg.line=lm,
            smooth=TRUE, spread=TRUE, boxplots='xy', span=0.5,
            ylab="Численность инфузорий",
            xlab="Длина раковины",
            by.groups=FALSE, data=mussel)
```



Выбор типа графика



Спецификация графических опций

Рисунок 8

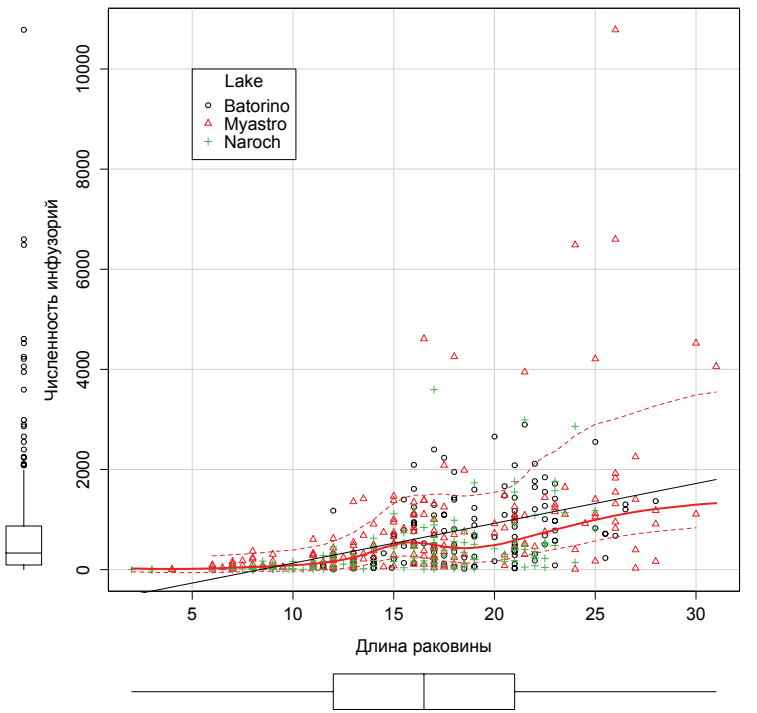


Рисунок 9

Все команды и/или выводимые результаты можно сохранить в файлах и в любой момент повторить. Тот же результат можно получить без запуска R Commander, загрузив сохраненный файл через консоль R.

По большому счету, не зная конструкций языка R (или просто не желая отягощать свою память их запоминанием), с использованием `Rcmdr` можно выполнить обработку данных с применением почти всех базовых статистических методов. Здесь представлены параметрические и непараметрические тесты, методы подгонки различных непрерывных и дискретных распределений, анализ многомерных таблиц сопряженности, одно- и многофакторный дисперсионный анализ, метод главных компонент и кластеризация, различные формы обобщенных регрессионных моделей и др. Достоин тщательного изучения развитый аппарат анализа и тестирования полученных моделей. Детальное описание техники работы с R Commander, а также особенности реализации алгоритмов обработки данных можно найти в руководствах Larson-Hall (2009) и Karp (2014).

Тем не менее как язык жестов не может заменить человеческого общения на естественном языке, так и знание языка R существенно расширяет границы возможностей пользователя и делает общение со средой R приятным и захватывающим. И тут автоматическая генерация команд в R Commander может оказаться для читателя прекрасным средством для знакомства с основами языка R и обучения специфике вызова отдельных функций. Последующие главы руководства мы посвятим обсуждению процедур обработки данных только на уровне языковых конструкций.

1.4. Объекты, пакеты, функции, устройства

Язык R принадлежит к семейству так называемых высокоуровневых объектно-ориентированных языков программирования. Для неспециалиста строгое определение понятия «объект» является достаточно абстрактным. Однако для простоты можно называть объектами все, что было создано в процессе работы с R.

Выделяют два основных типа объектов:

1) **объекты, предназначенные для хранения данных** («*data objects*») – это отдельные переменные, векторы, матрицы и массивы, списки, факторы, таблицы данных;
2) **функции** («*function objects*») – это поименованные программы, предназначенные для создания новых объектов или выполнения определенных действий над ними.

Объекты среды R, предназначенные для коллективного и свободного использования, часто комплектуются в пакеты, объединяемые сходной тематикой или методами обработки данных. Есть некоторое отличие между терминами *пакет* («*package*») и *библиотека* («*library*»). Термин «*library*» определяет директорию, которая может содержать один или несколько пакетов. Термин «*package*» обозначает совокупность функций, html-страниц руководств и наборов данных, предназначенных для тестирования или обучения.

Пакеты инсталлируются в определенной директории операционной системы или могут храниться в неустановленном виде и распространяться в заархивированных zip-файлах (версия пакета должна соответствовать конкретной версии вашей системы R). Полная информация о пакете (версия, основное тематическое направление, авторы, даты изменений, лицензии, другие функционально связанные пакеты, полный список функций с указанием на их назначение и прочее) может быть получена командой **library**(help = <имя_пакета>), например: **library**(help = Matrix).

Все пакеты R относятся к одной из трех категорий: базовые ("base"), рекомендуемые ("recommended") и прочие, установленные пользователем. Получить их список на конкретном компьютере можно, подав команду **library**() или:

```
installed.packages(priority = "base")
installed.packages(priority = "recommended")
# Получение полного списка пакетов
packlist <- rownames(installed.packages())
# Вывод информации в буфер обмена в формате для Excel
write.table(packlist, "clipboard", sep="\t", col.names=NA)
```

Базовые и рекомендуемые пакеты включены в инсталляционный файл R. Разумеется, нет необходимости сразу устанавливать «про запас» много разных пакетов. Для установки необходимого пакета достаточно в командном окне R Console выбрать пункт меню **Пакеты > Установить пакет(ы)** или ввести, например, команду (компьютер должен быть при этом подключен к Интернету):

```
install.packages(c("vegan", "xlsReadWrite", "car"))
```

Скачивание пакетов с русского «зеркала» http://cran.gis-lab.info может оказаться более быстрым, в связи чем полезно будет соответствующим образом отредактировать файл Rprofile.site (см. раздел 1.1).

Другой вариант установки пакетов – зайти на сайт http://cran.gis-lab.info/web/packages, выбрать нужный пакет в виде zip-файла и скачать в выбранную папку на своем компьютере. В этом случае на сайте можно предварительно посмотреть всю информацию по пакету, в частности описание входящих в него функций, и определиться, насколько он вам необходим. Далее нужно выполнить пункт командного меню **Пакеты > Установить пакеты из локальных zip-файлов**.

При запуске R загружаются только некоторые базовые пакеты. Для инициализации любого другого пакета перед его непосредственным использованием нужно ввести команду **library** (<*имя_пакета*>).

Выяснить, какие пакеты загружены в данной сессии работы с R, можно, выполнив следующую команду:

```
sessionInfo()
R version 2.13.2 (2011-09-30)
Platform: i386-pc-mingw32/i386 (32-bit)

attached base packages:
```

```
[1] stats  graphics  grDevices utils  datasets  methods  base

other attached packages:
[1] vegan_2.0-2   permute_0.6-3

loaded via a namespace (and not attached):
[1] grid_2.13.2     lattice_0.19-33 tools_2.13.2
```

В следующей таблице приведен список (возможно, не исчерпывающе полный) пакетов, которые использованы в скриптах, описанных в настоящей книге:

Пакеты R	Назначение
«Базовые» пакеты	
base	Базовые конструкции R
compiler	Компилятор R
datasets	Набор таблиц с данными для тестирования и демонстрации функций
graphics	Базовые графические функции
grDevices	Драйверы графических устройств, палитры цветов, шрифты
grid	Низкоуровневые функции создания графических объектов
methods	Компоненты объектно-ориентированного программирования (классы, методы)
splines	Функции для создания моделей типа «сплайн»
stats	Базовые функции статистического анализа
stats4	Методы статистических функций класса S4
tcltk	Компоненты пользовательского интерфейса
tools	Информационная поддержка, администрирование и документирование
utils	Различные утилиты отладки, ввода-вывода, архивирования и прочее
«Рекомендуемые» пакеты	
boot	Функции различных процедур бутстрепа и «складного ножа»
class	Различные алгоритмы неиерархической классификации и распознавания
cluster	Алгоритмы кластеризации
codetools	Анализ и отладка кода R
foreign	Чтение и запись файлов в разных форматах (DBF, SPSS, DTA, Stata)
KernSmooth	Функции, обслуживающие оптимизацию ядерного сглаживания
lattice	Графические функции (Sarkar, 2008)
MASS	Набор данных и статистических функций (Venables & Ripley, 2002)
Matrix	Операции с матрицами и векторами
mgcv	Обобщенные аддитивные модели и модели со смешанными эффектами
nlme	Линейные и нелинейные модели со смешанными эффектами
nnet	Нейронные сети
rpart	Деревья классификации и регрессии
spatial	Функции кригинга и анализа пространственного распределения точек
survival	Анализ выживаемости (модель Кокса и др.)
Дополнительно установленные пакеты	
adegenet	Алгоритмы анализа генетических расстояний

Пакеты R	Назначение
arm	Анализ регрессионных моделей – приложение к книге Gelman & Hill (2007)
car	Процедуры, связанные с прикладным регрессионным анализом
corrplot	Отображение корреляционных матриц в графическом виде
fitdistrplus	Подбор параметров статистических распределений
FWDselect, packfor	Селекция информативных переменных в регрессионных моделях
gamair	Наборы данных для тестирования аддитивных моделей
geosphere	Оценка географических расстояний
ggplot2	Графический пакет высокой функциональности
DAAG	Приложение к книге Maindonald & Braun (2010)
Hmisc	Набор полезных функций от профессора Ф. Харрела (F. Harrell)
HSAUR2	Приложение к книге Everitt & Hothorn (2010)
ISwR	Приложение к книге Dalgaard (2008)
jpeg	Работа с графическими файлами jpeg
lars	Специальные виды регрессии (LARS, Lasso и др.)
lavaan	Конфирматорный анализ и модели структурных уравнений
lmodel2	Реализация моделей регрессии I и II типов (MA, SMA, RMA)
maptools	Инструментарий для работы с географическими картами
mice	Процедуры анализа и восстановления пропущенных значений
moments	Функции расчета выборочных моментов
nortest	Критерии для проверки гипотез о нормальном распределении
outliers	Анализ выбросов в данных
pastecs	Анализ пространственных и временных рядов в экологии
pls	Регрессия на главные компоненты
pwr	Оценка статистической мощности
reshape2	Гибкие инструменты для преобразования таблиц данных
robustbase	Робастные методы построения регрессионных моделей
rootSolve	Нахождение корней функции с несколькими переменными
scales	Утилиты для формирования осей графиков
sem	Модели структурных уравнений
semPlot	Визуализация структурных связей
sm	Оценка плотности распределений и методы сглаживания
sp	Классы и методы доступа к пространственным данным
spatstat	Методы пространственной статистики, подгонка моделей
spdep	Пространственные зависимости: геостатистические методы и моделирование
stargazer	Вывод информации о статистических моделях в разных форматах
vcd	Визуализация категориальных данных
vegan	Методы анализа биологических сообществ (меры сходства, разнообразия и вложенности, ординация и многомерный анализ)

Если мы попробуем загрузить пакет, еще не установленный в R, или попытаемся использовать функции еще не загруженного пакета, то получим такие сообщения системы:

```
sem(model, data = PoliticalDemocracy)
Ошибка: не могу найти функцию "sem"
library(lavaan)
Ошибка в library(lavaan): нет пакета под названием 'lavaan'
```

Следующая функция, написанная K. Cichini (http://bit.ly/1DXtW48), принимает в качестве исходного параметра список используемых пользователем пакетов и сама разбирается, какие следует загрузить, а какие нужно предварительно установить. Для понимания работы скрипта необходимо знание конструкций языка R, описываемых в следующем разделе, но интересующийся читатель может вернуться к приведенным командам позднее.

```
instant_pkgs <- function(pkgs) {
pkgs_miss <- pkgs[which(!pkgs %in% installed.packages()[, 1])]
# Инсталлируем пакеты, не подготовленные к загрузке:
if (length(pkgs_miss) > 0) {
install.packages(pkgs_miss)
}
# Загружаем пакеты, которые еще не загружены:
attached <- search()
attached_pkgs <- attached[grepl("package", attached)]
need_to_attach <- pkgs[which(!pkgs %in% gsub("package:", "", attached_pkgs))]
if (length(need_to_attach) > 0) {
   for (i in 1:length(need_to_attach))
require(need_to_attach[i], character.only = TRUE)
}
}

# Пример вызова:
instant_pkgs(c("base", "jpeg", "vegan"))
```

Получить список функций каждого пакета можно, например, подав команду

```
ls(pos = "package:vegan")
```

Примечание: `ls()` является функцией общего назначения для вывода списка объектов, находящихся в заданном окружении, или среде («*environment*»). Приведенная выше команда устанавливает в качестве таковой среду пакета `vegan`. Если подать эту команду без параметров, то получим список объектов, созданных за время текущей сессии в глобальной среде R («*global environment*»).

Получить список аргументов любой функции загруженного пакета позволяет команда `args()`. Например, широко используемая нами впоследствии функция для подгонки линейных моделей `lm()` имеет следующие параметры:

```
args(lm)
function (formula, data, subset, weights, na.action,
 method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
 singular.ok = TRUE, contrasts = NULL, offset, ...)
```

Если ввести команду, состоящую только из названия функции (то есть без круглых скобок), то можно получить исходный код этой функции. Например, в случае с функцией **IQR**(), вычисляющей интерквартильный размах, получаем:

```
IQR
function (x, na.rm = FALSE)
diff(quantile(as.numeric(x), c(0.25, 0.75), na.rm = na.rm, names = FALSE))
```

Продвинутый пользователь может внести изменения в этот код и «перенаправить» вызов стандартной функции на свою версию.

Однако если мы захотим таким же образом посмотреть код функции **predict**(), которая используется для расчета прогнозируемых значений разных моделей, то получим:

```
predict
function (object, ...)
UseMethod("predict")
```

В данном случае **predict**() представляет собой «универсальную» функцию: в зависимости от класса подаваемой на нее модели (например, lm для линейной регрессии, glm для пуассоновской или логистической регрессии, lme для модели со смешанными эффектами и т. д.) вызывается соответствующий метод получения прогнозируемых значений. В частности, эта функция используется для реализации следующих методов:

```
methods("predict")
 [1] predict.ar*                predict.Arima*
 [3] predict.arima0*            predict.glm
 [5] predict.HoltWinters*       predict.lm
 [7] predict.loess*             predict.mlm
 [9] predict.nls*               predict.poly
[11] predict.ppr*               predict.prcomp*
[13] predict.princomp*          predict.smooth.spline*
[15] predict.smooth.spline.fit* predict.StructTS*
   Non-visible functions are asterisked
```

Этот пример связан с идеями объектно-ориентированного программирования (ООП), лежащими в основе среды R. Для ООП в так называемом стиле S3 *метод* – это, собственно говоря, функция, которая вызывается другой *универсальной* («*generic*») функцией (например, **print**(), **plot**() или **summary**()) в зависимости от класса объекта, подаваемого на эту универсальную функцию. При этом за «объектную ориентированность» отвечает атрибут class, который обеспечивает корректную диспетчеризацию и вызов необходимого метода для данного объекта. Так, «функция-метод» для получения прогнозируемых значений обобщенной линейной модели будет иметь вызов **predict.glm**(), при сглаживании сплайнами – **predict.smooth.spline**() и т. д. Подробную информацию о модели ООП S3 можно получить в разделе справки S3Methods, а по более продвинутой модели S4 – в разделе Methods (см. также Wickham, 2014).

Наконец, рассмотрим некоторые простейшие приемы сохранения и загрузки результатов работы:

- **sink**(file = *<имя файла>*) – выводит результаты выполнения последующих команд в режиме реального времени в файл с заданным именем; для прекращения действия этой команды необходимо выполнить команду **sink**() без параметров;
- **save**(file = *<имя файла>*, *<список сохраняемых объектов>*) – сохраняет указанные объекты в двоичном файле XDR-формата, с которым можно работать в любой операционной системе;
- **load**(file = *<имя файла>*) – загружает сохраненные ранее объекты в текущую среду;
- **save.image**(file= *<имя файла>*) – сохраняет все объекты, созданные в ходе работы, в виде специфичного для R rda-файла.

Пример передачи сформированной таблицы с данными в буфер обмена в формате, совместимом со структурой листа Excel, был приведен выше в настоящем разделе. В главе 6 будет приведен пример передачи данных из объекта линейной модели в файл Word.

Среда R может генерировать пиксельное изображение необходимого качества почти для любого разрешения дисплея или устройства печати, а также сохранять содержимое графических окон в файлах разного формата. Для каждого устройства графического вывода существует функция соответствующего драйвера: для получения полного списка драйверов можно воспользоваться командой **help**(Devices). Среди графических устройств наиболее употребительными являются:

- **windows**() – графическое окно Windows (экран, принтер или метафайл);
- **png**(), **jpeg**(), **bmp**(), **tiff**() – вывод в растровый файл соответствующего формата;
- **pdf**(), **postscript**() – вывод графической информации в файл PDF или PostScript.

По завершении работы с устройством вывода следует отключить его драйвер командой **dev.off**(). Существует возможность одновременного запуска нескольких устройств графического вывода и переключения между ними (см., например, Шипунов и др. (2012), с. 278).

Глава 2

Описание языка R

2.1. Типы данных

R работает со следующими элементарными типами данных (они же являются классами соответствующих переменных):

- **numeric** – переменные, содержащие целочисленные (integer) и действительные числа (double);
- **logical** – переменные, содержащие логические значения: `FALSE` (сокращенно `F`) и `TRUE` (`T`);
- **character** – символьные переменные (отдельные значения таких переменных задаются в двойных либо одинарных кавычках).

Имена различным объектам можно присваивать как на латинице, так и на кириллице, но следует учесть, что `a` (кириллица) и `a` (латиница) – это два разных символа. Во избежание путаницы и других (связанных с кодировкой) проблем мы настоятельно рекомендуем использовать только латиницу. Кроме того, среда R чувствительна к регистру, то есть строчные и заглавные буквы в ней различаются. Имя переменной должно начинаться с буквы (иногда допускается точка), за которой следует сочетание из букв, цифр и символов точки и нижнего подчеркивания. При помощи команды **?**<*имя*> можно проверить, существует ли переменная или функция с указанными именем.

Проверка на принадлежность переменной к определенному классу осуществляется функциями **is.numeric**(<*имя_объекта*>), **is.integer**(<*имя*>), **is.logical**(<*имя*>), **is.character**(<*имя*>), а для преобразования объекта в другой тип используются функции **as.numeric**(<*имя*>), **as.integer**(<*имя*>), **as.logical**(<*имя*>), **as.character**(<*имя*>).

Переменные в R могут принимать ряд специальных значений:

- `Inf` - положительная или отрицательная бесконечность (обычно результат деления вещественного числа на 0);
- `NA` - «отсутствующее значение» («*Not Available*»);
- `NaN` - «не число» («*Not a Number*»).

Проверить, относится ли то или иное значение конкретной переменной к какому-либо из этих специальных типов, можно функциями **is.finite**(<*имя*>), **is.na**(<*имя*>) и **is.nan**(<*имя*>) соответственно. Эти три функции возвращают вектор из логических значений `TRUE` или `FALSE`, которые отражают результат проверки.

Выражение («*expression*») языка R представляет собой сочетание таких элементов, как оператор присваивания, арифметические или логические операторы, имена объектов и имена функций. Результат выполнения выражения, как правило, сразу отображается в командном или графическом окне. Однако при выполнении операции присваивания результат сохраняется в соответствующем объекте и на экран не выводится.

В качестве оператора присваивания в R можно использовать либо символ =, либо пару символов <- (присваивание определенного значения объекту слева) или -> (присваивание значения объекту справа). Хорошим стилем программирования считается использование <-.

Выражения языка R организуются в скрипте по строкам. В одной строке можно ввести несколько команд, разделяя их символом ;. Одну команду можно также расположить на нескольких строках.

Объекты типа **numeric** могут составлять выражения с использованием традиционных арифметических операций + (сложение), - (вычитание), * (умножение), / (деление), ^ (возведение в степень), %/% (целочисленное деление), %% (остаток от деления). Операции имеют обычный приоритет, то есть сначала выполняется возведение в степень, затем умножение или деление, а потом уже сложение или вычитание. В выражениях могут использоваться круглые скобки, и заключенные в них операции имеют наибольший приоритет.

Логические выражения можно составлять с использованием следующих логических операторов:

- «Равно» ==;
- «Не равно» !=;
- «Меньше» <;
- «Больше» >;
- «Меньше либо равно» <=;
- «Больше либо равно» >=;
- «Логическое И» &;
- «Логическое ИЛИ» |;
- «Логическое НЕ» !.

2.2. Векторы и матрицы

Вектор представляет собой поименованный одномерный объект, содержащий набор однотипных элементов (числовые, логические либо текстовые значения – никакие их сочетания не допускаются). Для создания векторов небольшой длины в R используется функция конкатенации **c**() (от «*concatenate*» – объединять, связывать). В качестве аргументов этой функции через запятую перечисляют объединяемые в вектор значения, например:

```
my.vector <- c(1, 2, 3, 4, 5)
my.vector
[1] 1 2 3 4 5
```

Вектор можно создать также при помощи функции **scan**(), которая «считывает» последовательно вводимые с клавиатуры значения:

```
X <- scan()
1: 2.9 # после каждого нового значения нажать клавишу "Ввод"
2: 3.1
3: 3.4
4: 3.4
5: 3.7
6: 3.7
7: 2.8
8: 2.5
9: # выполнение команды scan завершают введением пустой строки
Read 8 items  # программа сообщает о считывании 8 значений
X
[1] 2.9 3.1 3.4 3.4 3.7 3.7 2.8 2.5
```

Один из недостатков создания векторов при помощи функции **scan**() состоит в том, что если при вводе значений с клавиатуры допущена ошибка, то приходится либо начать ввод заново, либо воспользоваться специальными инструментами корректировки (например, функцией **fix**(); здесь эти способы не рассматриваются).

Для создания векторов, содержащих последовательную совокупность чисел, удобна функция **seq**() (от «*sequence*» – последовательность). Так, вектор с именем S, содержащий совокупность целых чисел от 1 до 7, можно создать следующим образом:

```
S <- seq(1,7)
S
[1] 1 2 3 4 5 6 7
```

Идентичный результат будет получен при помощи команды

```
S <- 1:7
S
[1] 1 2 3 4 5 6 7
```

В качестве дополнительного аргумента функции **seq**() можно указать шаг приращения чисел:

```
S <- seq(from = 1, to = 5, by = 0.5)
S
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

Векторы, содержащие одинаковые значения, создают при помощи функции **rep**() (от «*repeat*» – повторять). Например, для формирования текстового вектора Text, содержащего пять значений "test", следует выполнить команду

```
Text <- rep("test", 5)
Text
[1] "test" "test" "test" "test" "test"
```

Система R способна выполнять самые разнообразные операции над векторами. Так, несколько векторов можно объединить в один, используя уже рассмотренную выше функцию конкатенации:

```
v1 <- c(1, 2, 3)
v2 <- c(4, 5, 6)
V <- c(v1, v2)
V
[1] 1 2 3 4 5 6
```

Если попытаться объединить, например, текстовый вектор с числовым, сообщение об ошибке не появится – программа просто преобразует все значения в текстовые:

```
# создаем текстовый вектор text.vect:
text.vect <- c("a", "b", "c")
# объединяем числовой вектор v1 (см. выше)
# с текстовым вектором text.vect:
new.vect <- c(v1, text.vect)
# просмотр содержимого нового вектора new.vect:
new.vect
[1] "1" "2" "3" "a" "b" "c"
# все значения нового вектора взяты в кавычки,
# что указывает на их текстовую природу;
# для подтверждения этого воспользуемся командой class():
class(new.vect)
[1] "character" # все верно: "character" значит "текстовый"
```

Для работы с определенным элементом вектора необходимо иметь способ отличать его от других элементов. Для этого при создании вектора всем его компонентам автоматически присваиваются индексные номера, начиная с 1. Чтобы обратиться к конкретному элементу, необходимо указать имя вектора и индекс этого элемента в квадратных скобках:

```
# создадим числовой вектор y, содержащий 5 числовых значений:
y <- c(5, 3, 2, 6, 1)
# проверим, чему равен третий элемент вектора y:
y[3]
[1] 2
```

Используя индексные номера, можно выполнять различные операции с избранными элементами разных векторов:

```
# создадим еще один числовой вектор z, содержащий 3 значения:
z <- c(0.5, 0.1, 0.6)
# умножим первый элемент вектора y на третий элемент вектора z # (то есть 5*0.6):
y[1]*z[3]
[1] 3
```

Индексирование является мощным инструментом, позволяющим создавать совокупности значений в соответствии с определенными критериями. Например,

для вывода на экран 3-го, 4-го и 5-го значений вектора y необходимо выполнить команду

```
y[3:5]
[1] 2 6 1
```

Из этого же вектора мы можем выбрать, например, только первое и четвертое значения, используя уже известную нам функцию конкатенации **c**():

```
y[c(1, 4)]
[1] 5 6
```

Похожим образом мы можем удалить первое и четвертое значения из вектора y, применив знак «минус» перед функцией конкатенации:

```
y[-c(1, 4)]
[1] 3 2 1
```

В качестве критерия для выбора значений может служить логическое выражение. Для примера выберем из вектора y все значения > 2:

```
y[y > 2]
[1] 5 3 6
```

Индексирование является также удобным инструментом для внесения исправлений в имеющихся векторах. Например, так можно исправить второе значение созданного нами ранее вектора z с 0.1 на 0.3:

```
z[2] <- 0.3
z
[1] 0.5 0.3 0.6
```

Для упорядочения значений вектора по возрастанию или убыванию используют функцию **sort**() в сочетании с аргументом decreasing = FALSE или decreasing = TRUE соответственно (*«decreasing»* значит «убывающий»):

```
sort(z)                 # по умолчанию  decreasing = FALSE
[1] 0.3 0.5 0.6
sort(z, decreasing = TRUE)
[1] 0.6 0.5 0.3
```

Матрица представляет собой двухмерный вектор. В R для создания матриц служит одноименная функция:

```
my.mat <- matrix(seq(1, 16), nrow = 4, ncol = 4)
my.mat
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

Обратите внимание на то, что по умолчанию заполнение матрицы происходит по столбцам, то есть первые четыре значения входят в первый столбец, следующие

четыре значения – во второй столбец и т. д. Такой порядок заполнения можно изменить, придав специальному аргументу `byrow` (от «*by row*» – по строкам) значение `TRUE`:

```
my.mat <- matrix(seq(1, 16), nrow = 4, ncol = 4, byrow = TRUE)
my.mat
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
```

В качестве заголовков строк и столбцов создаваемой матрицы автоматически выводятся соответствующие индексные номера (строки: `[1,]`, `[2,]` и т. д.; столбцы: `[,1]`, `[,2]` и т. д.). Для придания пользовательских заголовков строкам и столбцам матриц используют функции **`rownames`**`()` и **`colnames`**`()` соответственно. Например, для обозначения строк матрицы `my.mat` буквами `A`, `B`, `C` и `D` необходимо выполнить следующее:

```
rownames(my.mat) <- c("A", "B", "C", "D")
my.mat
  [,1] [,2] [,3] [,4]
A    1    2    3    4
B    5    6    7    8
C    9   10   11   12
D   13   14   15   16
```

В матрице `my.mat` имеются 16 значений, которые как раз вмещаются в имеющиеся четыре строки и четыре столбца. Но что произойдет, если, например, попытаться вместить вектор из 12 чисел в матрицу того же размера? В подобных случаях R заполняет недостающие значения за счет «зацикливания» («*recycling*») короткого вектора. Вот как это выглядит на примере:

```
my.mat2 <- matrix(seq(1, 12), nrow = 4, ncol = 4, byrow = TRUE)
my.mat2
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]    1    2    3    4
```

Как видим, для заполнения ячеек последней строки матрицы `my.mat2` программа снова использовала числа 1, 2, 3 и 4.

Альтернативный способ создания матриц заключается в применении функции **`dim`**`()` (от «*dimension*» – размерность). Так, матрицу `my.mat` мы могли бы сформировать из одномерного вектора следующим образом:

```
my.mat <- 1:16
# Задаем размерность 4×4 вектору my.mat:
```

```
dim(my.mat) <- c(4, 4)

my.mat
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16

# Функция dim() очень полезна. Она позволяет проверить
# размерность уже имеющейся матрицы (или таблицы данных):
dim(my.mat)
[1] 4 4
```

Матрицу можно собрать также из нескольких векторов, используя функции **cbind**() (от «*column*» – столбец и «*bind*» – связывать) или **rbind**() (от «*row*» – строка и «*bind*» – *связывать*):

```
# Создадим четыре вектора одинаковой длины:
a <- c(1, 2, 3, 4)
b <- c(5, 6, 7, 8)
d <- c(9, 10, 11, 12)
e <- c(13, 14, 15, 16)

# Объединим этим векторы при помощи функции cbind():
cbind(a, b, d, e)
     a b  d  e
[1,] 1 5  9 13
[2,] 2 6 10 14
[3,] 3 7 11 15
[4,] 4 8 12 16

# Объединим те же векторы при помощи функции rbind():
rbind(a, b, d, e)
  [,1] [,2] [,3] [,4]
a    1    2    3    4
b    5    6    7    8
d    9   10   11   12
e   13   14   15   16
```

Практически все векторные операции одинаково применимы в отношении матриц и массивов. Так, путем индексирования мы можем извлекать из матриц необходимые элементы и далее подвергать их требуемым преобразованиям. Рассмотрим лишь несколько примеров:

```
# Извлечем элемент матрицы my.mat, расположенный на
# пересечении 2-й строки и 3-го столбца:
my.mat[2, 3]
[1] 7
```

```
# Извлечем из матрицы все элементы, находящиеся в 4-м столбце
# (для этого номера строк перед запятой можно не указывать):
my.mat[, 4]
[1] 4  8 12 16
# Извлечем из матрицы все элементы, находящиеся в 1-й строке
# (в этом случае нет необходимости указывать номера столбцов):
my.mat[1, ]
[1] 1 2 3 4
# Перемножим 1-й и 4-й столбцы матрицы (поэлементно):
my.mat[, 1]*my.mat[, 4]
[1] 4 40 108 208
```

Отметим, наконец, что при необходимости матрицу можно транспонировать (то есть поменять местами строки и столбцы) при помощи функции **t**() (от «*transpose*»):

```
t(my.mat)
     A B  C  D
[1,] 1 5  9 13
[2,] 2 6 10 14
[3,] 3 7 11 15
[4,] 4 8 12 16
```

2.3. Факторы

В статистике данные очень часто группируют в соответствии с тем или иным признаком, например полом, социальным положением, стадией болезни, местом отбора проб и т. п. В R существует специальный класс векторов – **факторы** («*factors*»), которые предназначены для хранения меток соответствующих *уровней* номинальных переменных. Часто уровни факторов кодируют в виде чисел. В таких случаях очень важно проинструктировать программу так, чтобы она правильно «распознавала» уровни номинальной переменной от чисел как таковых.

Предположим, что в эксперименте по испытанию эффективности нового медицинского препарата было задействовано 10 пациентов-добровольцев, из которых шесть пациентов принимали новый препарат, а четверо остальных – плацебо (например, таблетки активированного угля). Для обозначения членов этих двух групп мы можем использовать метки 1 (препарат) и 0 (плацебо). Соответственно, информацию обо всех десяти участниках эксперимента мы могли бы сохранить в виде следующего вектора:

```
treatment <- c(1, 1, 1, 1, 1, 1, 0, 0, 0, 0)
treatment
[1] 1 1 1 1 1 1 0 0 0 0
```

При таком подходе, однако, программа будет «рассматривать» вектор `treatment` в качестве числового (проверьте при помощи команды **class**(treatment)). Это будет ошибкой с нашей стороны, поскольку ноль и единица обозначают лишь два

уровня номинальной переменной. С таким же успехом мы могли бы использовать, например, 10 для обозначения контрольной группы пациентов (то есть пациентов, принимавших плацебо) и 110 для обозначения пациентов, принимавших тестируемый препарат. Для преобразования числового (или любого другого) вектора в фактор в R существует одноименная функция **factor**():

```
treatment <- factor(treatment, levels = c(0, 1))
treatment
[1] 1 1 1 1 1 1 0 0 0 0
Levels: 0 1
```

Обратите внимание: теперь при выводе содержимого объекта treatment программа подсказывает нам, что этот объект является фактором с двумя уровнями (Levels: 0 1). Дополнительно убедиться в этом можно при помощи все той же команды **class**(treatment):

```
class(treatment))
[1] "factor"
```

Более надежным подходом, позволяющим не запутаться при выполнении анализа, является кодировка уровней факторов при помощи текстовых значений, а не чисел. Например, в нашем примере можно присвоить значение yes пациентам, принимавшим препарат, и значение no пациентам из контрольной группы. Мы можем перекодировать уровни уже имеющегося фактора treatment при помощи функции **levels**():

```
levels(treatment) <- c("no", "yes")
treatment
[1] yes yes yes yes yes yes no  no  no  no
Levels: no yes
```

Заметьте, что при выводе содержимого вектора treatment коды пациентов не заключены в двойные кавычки, как в случае с текстовыми значениями. Это является одним из внешних признаков того, что мы имеем дело именно с фактором, а не с текстовым вектором, содержащим шесть значений "yes" и четыре значения "no".

Фактор легко преобразовать обратно в числовой вектор, состоящий из порядковых номеров уровней фактора:

```
as.numeric(treatment)
[1] 2 2 2 2 2 2 1 1 1 1
```

Существует также специальная команда для создания факторов (от «*generate levels*» – сгенерировать уровни):

```
gl(n, k, length = n*k, labels = 1:n),
```

где n – количество уровней фактора; k – число повторов для каждого уровня; length – размер итогового объекта; labels – необязательный аргумент, который можно использовать для указания названий каждого уровня фактора. Например,

выполнение следующей команды приведет к созданию вектора my.fac, являющегося фактором с двумя уровнями – Control и Treatment, причем каждая из меток "Control" и "Treatment" будет повторена по 8 раз:

```
my.fac = gl(2, 8, labels = c("Control", "Treatment"))
my.fac
[1] Control Control Control Control  Control  Control  Control
[8] Control Treatment Treatment Treatment Treatment Treatment Treatment
[15] Treatment Treatment
Levels: Control Treatment
```

Еще одна полезная команда создает факторы, разделяя диапазон значений некоторого числового вектора x на интервалы:

```
cut(x, breaks, labels),
```

где в качестве аргумента breaks может выступать либо необходимое число интервалов, либо вектор со списком «точек разрыва», а labels определяет названия уровней:

```
x <- c(1, 2, 3, 4, 5, 2, 3, 4, 5, 6, 7)
cut(x, breaks = 3)
 [1] (0.994,3] (0.994,3] (3,5]     (3,5]     (3,5]     (0.994,3] (3,5]
 [8] (3,5]     (3,5]     (5,7.01]  (5,7.01]
Levels: (0.994,3] (3,5] (5,7.01]

cut(x, breaks = 3, labels = letters[1:3])
 [1] a a b b b a b b b c c
Levels: a b c

cut(x, breaks = quantile(x, c(0, .25, .50, .75, 1)),
    labels = c("Q1", "Q2", "Q3", "Q4"), include.lowest = TRUE)
 [1] Q1 Q1 Q2 Q2 Q3 Q1 Q2 Q2 Q3 Q4 Q4
Levels: Q1 Q2 Q3 Q4
```

В третьем фрагменте кода числовой вектор «разрезан» по квартильным значениям, а параметр include.lowest указан, чтобы избежать появления отсутствующих значений (NA) для $x = 1$.

2.4. Списки и таблицы данных

В отличие от вектора или матрицы, которые могут содержать данные только одного типа, в *список* («*list*») или *таблицу* («*data frame*») можно включать сочетания любых типов данных. Это позволяет эффективно, то есть в одном объекте, хранить разнородную информацию.

Каждый компонент списка может являться переменной, вектором, матрицей, фактором или другим списком. Кроме того, и сами эти элементы могут принадлежать к различным типам: числа, строки символов, булевы переменные. Спис-

ки служат наиболее общим способом хранения внутрисистемной информации: в частности, результаты большинства статистических анализов в программе R хранятся в объектах-списках.

Для создания списков служит одноименная функция **list**(). Рассмотрим пример:

```
# Сначала создадим три разнотипных вектора - с текстовыми,
# числовыми и логическими значениями:
vector1 <- c("A", "B", "C")
vector2 <- seq(1, 3, 0.5)
vector3 <- c(FALSE, TRUE)

# Теперь объединим эти три вектора в один объект-список,
# компонентам которого присвоим имена Text, Number и Logic:
my.list <- list(Text=vector1, Number=vector2, Logic=vector3)

# Просмотрим содержимое созданного списка:
my.list
$Text
[1] "A" "B" "C"
$Number
[1] 1.0 1.5 2.0 2.5 3.0
$Logic
[1] FALSE  TRUE
```

К элементам списка можно получить доступ посредством трех различных операций индексации. Для обращения к поименованным компонентам применяют знак $. Так, для извлечения компонентов Text, Number и Logic из созданного нами списка my.list необходимо последовательно ввести следующие команды:

```
my.list$Text
[1] "A" "B" "C"

my.list$Number
[1] 1.0 1.5 2.0 2.5 3.0

my.list$Logic
[1] FALSE  TRUE
```

Имеется возможность извлекать из списка не только его поименованные компоненты-векторы, но и отдельные элементы, входящие в эти векторы. Для этого необходимо воспользоваться уже рассмотренным ранее способом – индексацией при помощи квадратных скобок. Единственная особенность работы со списками здесь состоит в том, что сначала необходимо указать имя компонента списка, используя знак $, а уже затем номер(а) отдельных элементов этого компонента:

```
my.list$Text[2]
[1] "B"

my.list$Number[3:5]
```

```
[1] 2.0 2.5 3.0

my.list$Logic[1]
[1] FALSE
```

Извлечение компонентов списка можно осуществлять также с использованием двойных квадратных скобок, в которые заключается номер компонента списка:

```
my.list[[1]]
[1] "A" "B" "C"

my.list[[2]]
[1] 1.0 1.5 2.0 2.5 3.0

my.list[[3]]
[1] FALSE  TRUE
```

После двойных квадратных скобок с индексным номером компонента списка можно также указать номер(а) отдельных элементов этого компонента:

```
my.list[[1]][2]
[1] "B"

my.list[[2]][3:5]
[1] 2.0 2.5 3.0

my.list [[3]][1]
[1] FALSE
```

Созданный нами список `my.list` содержал всего лишь три небольших вектора, и мы знали, какие это векторы и на каком месте в списке они стоят. Однако на практике можно столкнуться с гораздо более сложно организованными списками, индексирование которых затрудняется из-за отсутствия представлений об их структуре. Для выяснения структуры объектов в языке R имеется специальная и очень полезная функция **`str`**`()` (от «*str*ucture»):

```
str(my.list)
List of 3
 $ Text  : chr [1:3] "A" "B" "C"
 $ Number: num [1:5] 1 1.5 2 2.5 3
 $ Logic : logi [1:2] FALSE TRUE
```

Из приведенного примера следует, что список `my.list` включает 3 компонента (`List of 3`) с именами `Text`, `Number` и `Logic` (перечислены в отдельных строках после знака `$`). Эти компоненты относятся к символьному (`chr`), числовому (`num`) и логическому (`logic`) типам данных соответственно. Кроме того, команда **`str`**`()` выводит на экран первые несколько элементов каждого вектора.

Таблица данных представляет собой объект R, по структуре напоминающий лист электронной таблицы Microsoft Excel. Каждый столбец таблицы является вектором, содержащим данные определенного типа. При этом действует правило,

согласно которому все столбцы должны иметь одинаковую длину (собственно, с «точки зрения» R таблица данных является частным случаем списка, в котором все компоненты-векторы имеют одинаковый размер).

Таблицы данных – это основной класс объектов R, используемых для хранения данных. Обычно такие таблицы подготавливаются при помощи внешних приложений (особенно популярна и удобна программа Microsoft Excel) и затем загружаются в среду R. Подробнее об импортировании данных в R будет рассказано ниже. Тем не менее небольшую таблицу можно собрать из нескольких векторов средствами самой системы R. Для этого используют функцию **data.frame**().

Предположим, у нас есть наблюдения по общей численности мужского (Male) и женского (Female) населения в трех городах City1, City2 и City3. Представим эти данные в виде одной таблицы с именем CITY. Для начала создадим текстовые векторы с названиями городов (city) и пола (sex), а также вектор со значениями численности представителей каждого пола (number):

```
city <- c("City1", "City1", "City2", "City2", "City3", "City3")
sex <- c("Male", "Female", "Male", "Female", "Male", "Female")
number <- c(12450, 10345, 5670, 5800, 25129, 26000)
```

Теперь объединим эти три вектора в одну таблицу данных и посмотрим, что получилось:

```
CITY <- data.frame(City = city, Sex = sex, Number = number)
CITY
   City    Sex Number
1 City1   Male  12450
2 City1 Female  10345
3 City2   Male   5670
4 City2 Female   5800
5 City3   Male  25129
6 City3 Female  26000
```

Обратите внимание на синтаксис функции **data.frame**(): ее аргументы перечисляются в формате «заголовок столбца = добавляемый вектор». В качестве заголовков столбцов могут выступать любые пользовательские имена, удовлетворяющие требованиям R (см. об этом подробнее в разделе 2.1).

Как и в случае со списками, извлечь отдельные компоненты таблиц можно с использованием знака $, квадратных скобок с указанием двух индексов [<номер_строки>, номер_столбца>], двойных квадратных скобок [[]] либо непосредственно по имени столбца:

```
CITY$Sex
[1] Male Female Male Female Male Female
Levels: Female Male

CITY$Number
[1] 12450 10345  5670  5800 25129 26000

# Идентичные результаты можно получить при помощи команд:
```

```
CITY[, 2]
[1] Male Female Male Female Male Female
Levels: Female Male

CITY[[3]]
[1] 12450 10345  5670  5800 25129 26000

CITY["Sex"]
[1] Male Female Male Female Male Female
Levels: Female Male

CITY["Number"]
[1] 12450 10345  5670  5800 25129 26000
```

После имени или индексного номера столбца можно указывать индексные номера отдельных ячеек таблицы, что позволяет извлекать содержимое этих ячеек:

```
# Извлекаем 4-й элемент из столбца Number:
CITY$Number[4]
[1] 5800

# Извлекаем элементы 1-3 из столбца Number:
CITY$Number[1:3]
[1] 12450 10345  5670

# Извлекаем все значения численности, превышающие 10000
CITY$Number[CITY$Number > 10000]
[1] 12450 10345 25129 26000

# Извлекаем все значения численности мужского населения:
CITY$Number[CITY$Sex == "Male"]
[1] 12450  5670 25129

# Повторяем те же команды, но с использованием []:
CITY[4, 3]
[1] 5800

CITY[1:3, 3]
[1] 12450 10345  5670

CITY[CITY$Number > 10000, 3]
[1] 12450 10345 25129 26000

CITY[CITY$Sex == "Male", 3]
[1] 12450  5670 25129
```

При работе с большими таблицами данных бывает сложно визуально исследовать все их содержимое перед началом анализа. Однако визуального просмотра содержимого таблиц и не требуется – полную сводную информацию о них (равно

как и о других объектах R) можно легко получить при помощи упомянутой ранее функции **str**():

```
str(CITY)
'data.frame':   6 obs. of  3 variables:
 $ City  : Factor w/ 3 levels "City1","City2",..: 1 1 2 2 3 3
 $ Sex   : Factor w/ 2 levels "Female","Male": 2 1 2 1 2 1
 $ Number: num  12450 10345 5670 5800 25129 ...
```

Как следует из представленного отчета, объект CITY является таблицей данных, в состав которой входят три переменные с шестью наблюдениями каждая. Две из этих переменных – City и Sex – программа автоматически распознала как факторы с тремя и двумя уровнями соответственно. Переменная Number является количественной. Для удобства выводятся также несколько первых значений каждой переменной.

Часто возникает необходимость выяснить лишь имена переменных, входящих в таблицу данных. Это можно сделать при помощи команды **names**():

```
names(CITY)
[1] "City"   "Sex"    "Number"
```

Имеется также возможность быстро просмотреть несколько первых или несколько последних значений каждой переменной, входящей в состав таблицы данных. Для этого используются функции **head**() и **tail**() соответственно:

```
head(CITY, n = 3)
   City    Sex Number
1 City1   Male  12450
2 City1 Female  10345
3 City2   Male   5670

tail(CITY, n = 2)
   City    Sex Number
5 City3   Male  25129
6 City3 Female  26000
```

При необходимости внесения исправлений в таблицу можно воспользоваться встроенным в R редактором данных. Внешне этот редактор напоминает обычный лист Excel, однако имеет весьма ограниченные функциональные возможности. Все, что он позволяет делать, – это добавлять новые или исправлять уже введенные значения переменных, изменять заголовки столбцов, а также добавлять новые строки и столбцы. Работая в стандартной версии R, редактор данных можно запустить из меню **Редактировать** > **Редактор данных** либо выполнив команду **fix**() (от «*fix*» – исправлять, чинить) из командной строки (например, **fix**(CITY)). После внесения исправлений редактор просто закрывают – все изменения будут сохранены автоматически.

Заполнение пустых значений

Часто на практике некоторые значения в таблице отсутствуют, что может быть обусловлено множеством причин: на момент измерения прибор вышел из строя,

по невнимательности персонала измерение не было занесено в протокол исследования, испытуемый отказался отвечать на определенный(е) вопрос(ы) в анкете, была утеряна проба и т. п. Ячейки с такими отсутствующими значениями (*«missing values»*) в таблицах данных R не могут быть просто пустыми – иначе столбцы таблицы окажутся разной длины, что недопустимо. Для обозначения отсутствующих наблюдений в языке R, как было отмечено ранее, имеется специальное значение – `NA` (*«not available»* – не доступно). В разделе 4.4 мы остановимся на решении проблемы заполнения пропусков подробнее. Здесь же отметим, что если значение `NA` имеет смысл нуля (например, экземпляров некого вида обнаружено не было), то легко произвести эту замену в таблице `DF` командой `DF[is.na(DF)] <- 0`.

Сортировка таблиц

Сортировка строк таблицы по различным ключам не составляет труда. Для этого служит функция **order**():

```
DF <- data.frame(X1 = c(1, 15, 1, 3), X2 = c(1, 0, 7, 0),
                 X3 = c(1, 0, 1, 2), X4 = c(7, 4, 41, 0),
                 X5 = c(1, 0, 5, 3))

row.names(DF) <- c("A", "B", "C", "D")

# DF1 - таблица, столбцы которой отсортированы
# по убыванию суммы значений:
DF1 <- DF[, rev(order(colSums(DF)))]

# DF2 - таблица, строки которой отсортированы в восходящем
# порядке по первому столбцу, затем в нисходящем по второму:
DF2 <- DF[order(DF$X1, -DF$X2), ]
```

Объединение таблиц

Представим, что мы имеем две таблицы:

DF1				
Y	N	A	B	C
12	22	0	1	0
12	23	1	3	0
12	24	0	0	1

DF2				
Y	N	A	B	D
13	22	0	1	2
13	23	0	3	0
13	24	1	0	5

Объединить их столбцы можно с использованием уже известной нам функции **cbind**():

```
cbind(DF1, DF2)
   Y  N A B C  Y  N A B D
1 12 22 0 1 0 13 22 0 1 2
2 12 23 1 3 0 13 23 0 3 0
3 12 24 0 0 1 13 24 1 0 5
```

Для объединения строк мы должны предварительно преобразовать объединяемые таблицы к единому списку столбцов:

```
DF1[, names(DF2)[!(names(DF2) %in% names(DF1))]] <- NA
DF2[, names(DF1)[!(names(DF1) %in% names(DF2))]] <- NA
rbind(DF1, DF2)
   Y  N A B  C  D
1 12 22 0 1  0 NA
2 12 23 1 3  0 NA
3 12 24 0 0  1 NA
4 13 22 0 1 NA  2
5 13 23 0 3 NA  0
6 13 24 1 0 NA  5
```

В приведенных командах оператор `%in%` служит для проверки того, встречаются ли значения одного вектора среди значений другого вектора (например, значения вектора с именами таблицы `DF2` среди значений вектора с именами таблицы `DF1`: `names(DF2) %in% names(DF1)`).

Аналогичную операцию мы можем выполнить гораздо более элегантно с помощью команды

```
merge(DF1, DF2, all = TRUE)
```

Функция `merge()` позволяет выполнять объединение таблиц всеми распространенными способами `join`-операций, свойственных языку SQL.

2.5. Импортирование данных в R

В предыдущих разделах были рассмотрены способы создания небольших по объему объектов для хранения данных (векторы, матрицы, списки, таблицы данных) средствами самой системы R. Следует отметить, однако, что возможности системы R по вводу и редактированию данных умышленно ограничены ее создателями, которые предполагали, что для этого будут использоваться другие средства (например, программа Microsoft Excel или базы данных). Поэтому подлежащие анализу объемные таблицы данных обычно подготавливаются при помощи сторонних приложений и только потом загружаются в рабочую среду R из внешних файлов. Хотя предпочтение при этом отдается текстовым файлам, выше был упомянут специальный пакет `foreign`, функции которого позволяют импортировать таблицы, сохраненные во множестве других распространенных форматов (Excel, SPSS, SAS, STATA, Access, Matlab, SQL, Oracle и т. п.; см. также руководство «R Data Import/Export» – http://bit.ly/1awy3cO).

Импортирование данных в систему R часто вызывает проблемы у тех, кто только начинает работать с этой программой. Тем не менее ничего сложного в этом нет. Ниже будут подробно рассмотрены наиболее распространенные способы импорта таблиц данных в рабочую среду R, однако сначала стоит ознакомиться с правилами подготовки загружаемых файлов:

- в импортируемой таблице с данными не должно быть пустых ячеек. Если некоторые значения по тем или иным причинам отсутствуют, вместо них следует ввести NA;
- импортируемую таблицу с данными рекомендуется преобразовать в простой текстовый файл с одним из допустимых расширений. На практике обычно используются файлы с расширением .txt, в которых значения переменных разделены знаками табуляции («*tab-delimited files*»), а также файлы с расширением .csv («*comma separated values*»), в которых значения переменных разделены запятыми или другим разделяющим символом;
- в качестве первой строки в импортируемой таблице рекомендуется ввести заголовки столбцов-переменных. Такая строка – удобный, но не обязательный элемент загружаемого файла. Если она отсутствует, то об этом необходимо сообщить в описании команды, которая будет управлять загрузкой файла (например, **read.table**() – см. ниже). Все последующие строки файла в качестве первого элемента могут содержать заголовки строк (если таковые предусмотрены), после которых следуют значения каждой из имеющихся в таблице переменных.

Имена столбцов таблицы необходимо задавать с соблюдением рассмотренных ранее правил присваивания имен переменным R (в частности, исключить пробелы и другие специальные символы, кроме точки и нижнего подчеркивания). Во избежание проблем, связанных с кодировкой, все текстовые величины в импортируемых файлах стоит создавать с использованием букв латинского алфавита.

Подлежащий импортированию файл рекомендуется поместить в рабочую папку программы, то есть папку, в которой R по умолчанию будет «пытаться найти» этот файл (см. раздел 1.1). Для выяснения пути к текущей рабочей папке R служит команда **getwd**() (от «*get working directory*» – узнать рабочую директорию):

```
getwd()
[1] "C:/Temp/"
```

Изменить рабочую директорию можно при помощи команды **setwd**() (от «*set working directory*» – создать рабочую директорию):

```
setwd("C:/My Documents")
# при выполнении этой команды внешне ничего не произойдет,
# однако последующий вызов команды getwd() покажет,
# что путь к рабочей папке изменился:
getwd()
[1] "C:/My Documents/"
```

Ниже приведен фрагмент типичной таблицы данных, которую можно успешно загрузить для анализа в среду R:

```
        Group   Variable1   Variable2   Variable3
Ivan    A       102         1.3         14
Vitaliy A       98          1.4         11
Sergey  B       45          NA          8
Mikhail B       50          3.2         6
```

Как видим, приведенный фрагмент имеет размерность 5×5, то есть состоит из пяти строк и пяти столбцов. В первой строке представлены заголовки всех имеющихся в таблице столбцов, за исключением первого. Первый столбец, хотя и не имеет собственного заголовка, не является пустым – он содержит имена добровольцев, участвовавших в некотором эксперименте (`Ivan`, `Vitaliy` и т. д.). Второй столбец имеет заголовок `Group` и содержит метки, по которым можно выяснить принадлежность испытуемых к той или иной экспериментальной группе (`A`, `B` и т. д.). Мы уже знаем, что в терминах языка R переменная `Group` называется фактором. В последующих столбцах (с заголовками `Variable1`, `Variable2` и т. д.) содержатся значения измеренных в ходе исследования переменных. В приведенном фрагменте таблицы имеется одно отсутствующее значение, вместо которого введено `NA`.

Пожалуй, одним из наиболее доступных и удобных средств подготовки данных для их последующего анализа при помощи R является программа Microsoft Excel. Для сохранения Excel-таблиц в виде `txt`- или `csv`-файлов обычно используется опция **Сохранить как** (Save as) в разделе **Файл** (File) главного меню этой программы. Другим простым и надежным способом экспорта данных из Excel являются создание в редакторе Блокнот (Notepad) нового файла и перенос туда через буфер обмена всей таблицы или выделенной ее части.

Основной функцией для импортирования данных в рабочую среду R является **`read.table`**`()`. Эта мощная функция позволяет достаточно тонко настроить процесс загрузки внешних файлов, в связи с чем она имеет большое количество управляющих аргументов. Наиболее важные из этих аргументов перечислены ниже в таблице (подробнее см. файл помощи, доступный по команде `?read.table`).

Аргумент	Назначение
`file`	Служит для указания пути к импортируемому файлу. Путь приводят либо в абсолютном виде (например, `file = "C:/Temp/MyData.txt"`), либо указывают только имя импортируемого файла (например, `file = "MyData.txt"`), но при условии, что последний хранится в рабочей папке программы (см. выше). Можно указывать также полную URL-ссылку на файл, который предполагается загрузить из Сети (например: `file = "http://somesite.net/YourData.csv"`). Начиная с версии `R 2.10` появилась возможность импортировать архивированные файлы в zip-формате
`header`	Служит для сообщения программе о наличии в загружаемом файле строки с заголовками столбцов. По умолчанию принимает значение `FALSE`. Если строка с заголовками столбцов имеется, этому аргументу следует присвоить значение `TRUE`
`row.names`	Служит для указания номера столбца, в котором содержатся имена строк (например, в рассмотренном выше примере это был первый столбец, поэтому `row.names = 1`). Важно помнить, что все имена строк должны быть уникальными, то есть одинаковые имена для двух или более строк не допускаются
`sep`	Служит для указания разделителя значений переменных, используемого в файле (от «*separator*» – разделитель). По умолчанию предполагается, что значения переменных разделены «пустым пространством», например в виде пробела или знака табуляции (`sep = " "`). В файлах формата `csv` значения переменных разделены запятыми, и поэтому для них `sep = ","`
`dec`	Служит для указания знака, используемого в файле для отделения целой части числа от дроби. По умолчанию `dec = "."`. Однако во многих странах в качестве десятичного знака применяют запятую, о чем важно вспомнить перед загрузкой файла и, при необходимости, использовать `dec = ","`. Следите, чтобы `dec` и `sep` не были одинаковыми!

nrows	Выражается целым числом, указывающим количество строк, которое должно быть считано из загружаемой таблицы. Отрицательные и иные значения игнорируются. Пример: `nrows = 100`
skip	Выражается целым числом, указывающим количество строк в файле, которое должно быть пропущено перед началом импортирования. Пример: `skip = 5`

Для загрузки тщательно подготовленных файлов (см. правила выше) достаточно использовать минимальный набор аргументов функции **read.table**(). В качестве примера предположим, что нам необходимо загрузить файл hydro_chem.txt, который хранится в рабочей папке R и содержит данные по химическому составу воды некоторого водоема. Загружаемую таблицу данных мы намерены сохранить в виде объекта с именем chem. Функция **read.table**() в этом случае может быть применена следующим образом:

```
chem <- read.table(file = "hydro_chem.txt", header = TRUE)
```

Как отмечено выше, импортируемые в R файлы часто имеют формат csv. Для их загрузки можно воспользоваться той же функцией **read.table**(), но при этом следует указать символ, который используется в качестве разделителя значений переменных в файле:

```
chem <- read.table(file = "hydro_chem.csv", header = TRUE, sep = ",")
```

Аналогом **read.table**() для считывания csv-файлов является функция **read.csv**():

```
chem <- read.csv(file = "hydro_chem.csv", header = TRUE)
```

Если подлежащий загрузке файл хранится в папке, отличной от рабочей папки R, то следует указать полный путь к нему. При этом пользователям операционной системы Windows необходимо помнить, что для указания путей к файлам в программе R используется не обратный одинарный слэш (\), а прямой одинарный (/) либо двойной обратный слэш (\\). Например, следующие две команды будут успешно восприняты R и приведут к идентичному результату – загрузке файла hydro_chem.txt и сохранению его в виде объекта chem:

```
chem <- read.csv(file = "D:\\Documents\\hydrochem.txt", header = TRUE)
chem <- read.csv(file = "D:/Documents/hydrochem.txt", header = TRUE)
```

Для интерактивного выбора загружаемого файла, который хранится вне рабочей папки R, на компьютерах с операционной системой Windows можно применить также вспомогательную функцию **file.choose**() («выбрать файл»). Выполнение этой команды приводит к открытию обычного диалогового окна, в котором пользователь выбирает папку с необходимым файлом. Очень удобно совмещать **file.choose**() с командами **read.table**() или **read.csv**(), как в следующем примере:

```
chem <- read.table(file = file.choose(),
                   header = TRUE, sep = ",")
```

2.6. Представление дат и времени. Временные ряды

Форматы представления дат и времени

Анализ данных, содержащих даты и время, может иногда сопровождаться приличной головной болью. Причин этому несколько:

- разные годы начинаются в разные дни недели;
- високосные годы имеют дополнительный день в феврале;
- американцы и европейцы по-разному представляют даты (например, 8/9/2011 будет 9 августа 2011 г. для первых и 8 сентября 2011 г. для вторых);
- в некоторые годы добавляется так называемая «секунда координации»;
- страны различаются по временным поясам и в ряде случаев применяют переход на «зимнее» и «летнее» время.

К счастью, система дат и времени в R такова, что многие из указанных проблем относительно легко преодолеваются. С форматом представления дат и времени в R можно ознакомиться, выполнив команду

```
Sys.time()
[1] "2011-09-06 00:38:04 EEST"
```

Как видим, формат строго иерархичен: сначала идет наиболее крупная временная единица – год, потом месяц и день, разделенные дефисом, а затем пробел, час, минуты, секунды и, после еще одного пробела, аббревиатура временной шкалы.

Отдельные элементы из этого результата можно извлечь при помощи функции **substr**() (от «*substring*» – часть строки), указав позиции первого и последнего элементов извлекаемой строки:

```
substr(as.character(Sys.time()), 1, 10)
[1] "2011-09-06"
```

или

```
substr(as.character(Sys.time()), 12, 19)
[1] "00:38:04"
```

Функция **date**() позволяет выяснить текущую дату:

```
[1] "Tue Sep 06 01:02:24 2011"
```

Если выполнить команду

```
unclass(Sys.time())
[1] 1315258719
```

то получим время в так называемом формате POSIXct, то есть время, выраженное в секундах, прошедших с 1 января 1970 г. (его еще трактуют как Unix-время, по названию операционной системы). Такой «машинный» формат бывает удобным для включения соответствующих значений в таблицы данных.

Для человека более удобным является представление времени в формате класса POSIXlt. Объекты этого класса представляют собой списки, включающие такие элементы, как секунды, минуты, часы, дни, месяцы и годы. Например, мы можем конвертировать системное время в объект POSIXlt класса следующим образом:

```
date <- as.POSIXlt(Sys.time())
```

Из списка date далее легко можно извлечь такие содержащиеся в нем элементы, как sec (секунды), min (минуты), hour (часы), mday (день месяца), mon (месяц), year (год), wday (день недели, начиная с воскресенья = 0), yday (день года, начиная с 1 января = 0) и isdst («*is daylight savings time in operation?*» – логическая переменная, обозначающая, используется ли режим перехода на «зимнее» и «летнее» время: 1, если TRUE, и 0, если FALSE). Например:

```
date$wday
[1] 2

date$yday
[1] 248
```

Для просмотра всего содержимого списка date можно использовать функцию **unclass**() в сочетании с **unlist**():

```
unlist(unclass(date))
sec         min        hour        mday        mon
29.97798    26.00000   12.00000    6.00000     8.00000
year        wday       yday        isdst
111.00000   2.00000    248.00000   1.00000
```

Вычисления с датами и временем

В R можно выполнять следующие типы вычислительных операций с датами и временем:

- число + время;
- время – число;
- время_1 – время_2;
- время_1 «логический оператор» время_2 (в качестве логического оператора могут использоваться ==, !=, <=, <, > или >=).

Важной особенностью является то, что перед выполнением любых вычислений с датами или временем необходимо конвертировать их в объекты класса POSIXlt. Например, количество дней между 15 сентября 2011 г. и 15 сентября 2000 г. можно найти следующим образом:

```
t1 <- as.POSIXlt("2011-09-15")
t2 <- as.POSIXlt("2000-09-15")
t1 - t2
Time difference of 4017 days
```

Разницу во времени, выраженную в часах, можно рассчитать так:

```
t3<-as.POSIXlt("2010-09-22 08:30:30")
t4<-as.POSIXlt("2010-09-22 22:25:30")
t4-t3
Time difference of 13.91667 hours
```

Еще проще разницу между двумя датами можно найти при помощи готовой функции **difftime**() (от «*difference*» – разница и «*time*» – время):

```
difftime("2011-09-22", "2010-06-22")
Time difference of 457 days
```

Для извлечения самого числа дней из результата выполнения предыдущей команды используйте функцию **as.numeric**():

```
as.numeric(difftime("2011-09-22", "2010-06-22"))
[1] 457
```

Обратите внимание: в R отсутствует возможность для сложения двух дат.

Измерить продолжительность какого-нибудь вычислительного процесса можно с использованием функции «процессорного времени» **proc.time**(), которая, по существу, работает как секундомер: вы засекаете стартовое время, запускаете процесс, а после его завершения находите разность времен. Например, чтобы вычислить 10 000 значений арктангенса, потребуется ~0.02 секунды:

```
t1 <- proc.time()
for (x in 1:10000) y <- atan(x)
time.result <- proc.time() - t1
time.result["elapsed"]
elapsed
   0.02
```

Преобразование текстовых переменных в машинный формат времени

Функция **strptime**() (от «*strip*» – оголять и «*time*» – время) позволяет преобразовывать текстовые значения, отражающие даты и время, в машинный (то есть принятый в R) формат времени. При этом важно верно указать формат (при помощи аргумента `format`), в котором приведены временные величины в текстовых переменных. Приняты следующие условные обозначения для наиболее часто используемых форматов дат и времени (детали доступны по команде `?strptime`):

- `%a` – сокращенное название для недели (англ. яз.);
- `%A` – полное название для недели (англ. яз.);
- `%b` – сокращенное название месяца (англ. яз.);
- `%B` – полное название месяца (англ. яз.) ;
- `%d` – день месяца (01–31);
- `%H` – часы от 00 до 23;
- `%I` – часы от 01 до 12;

- %j – порядковый номер дня года (001–366);
- %m – порядковый номер месяца (01–12);
- %M – минуты (00–59);
- %S – секунды (00–61, с возможностью добавить «секунду координации»);
- %U – неделя года (00–53); первое вокресенье считается первым днем первой недели;
- %w – порядковый номер дня недели (0–6, воскресенье – 0);
- %W – неделя года (00–53); первый понедельник считается первым днем первой недели;
- %Y – год с указанием века;
- %y – год без указания века.

Рассмотрим пример. Предположим, у нас имеется текстовый вектор, в котором хранятся даты в формате программы Microsoft Excel:

```
dates.excel <- c("25/02/2008", "24/04/2009",
"14/06/2009", "25/07/2010", "04/03/2011")
```

Формат имеющихся Excel-дат таков, что сначала идет день месяца, затем порядковый номер самого месяца и, наконец, год с указанием века. Требуется преобразовать эти текстовые выражения в даты формата R. Используя приведенные выше обозначения форматов функции **strptime**(), параметр format можно представить в виде %d/%m/%Y. Тогда команда для преобразования Excel-дат в R-даты будет выглядеть следующим образом:

```
strptime(dates.excel, format = "%d/%m/%Y")
[1] "2008-02-25" "2009-04-24" "2009-06-14"
[2] "2010-07-25" "2011-03-04"
```

Вот еще один пример, в котором год приведен без указания века, а месяцы приведены в виде их сокращенных названий:

```
example2 <- c("1jan79", "2jan99", "31jan04", "30aug05")
strptime(other.dates, "%d%b%y")
 [1] "1979-01-01" "1999-01-02" "2004-01-31" "2005-08-30"
```

Для работы с разными форматами дат в R существует несколько специальных пакетов, среди которых особой рекомендации заслуживает lubridate.

Временные ряды

В R существует специальный класс объектов для работы с данными, представляющими собой временные ряды, – ts (от «*time series*» – временной ряд). Для создания объектов этого класса служит одноименная функция **ts**().

В качестве примера рассмотрим ежемесячные данные по рождаемости в г. Нью-Йорке, собранные в период с января 1946 г. по декабрь 1959 г. Пример заимствован из электронной книги «*A Little Book of R for Time Series*» (http://bit.ly/1vH9dQQ). Исходные данные можно загрузить с сайта профессора Роба Хиндмана (Rob J. Hyndman) следующим образом:

```
birth <- scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")
Read 168 items
```

Объект `birth` представляет собой вектор со всеми 168 ежемесячными значениями рождаемости (в тыс. человек), в чем можно убедиться при помощи функции

```
is.vector(birth)
[1] TRUE
```

Функция **`head`**() позволяет просмотреть первые несколько значений вектора `birth` (по умолчанию первые 6 значений):

```
head(birth)
[1] 26.663 23.598 26.931 24.740 25.806 24.364
```

Преобразовать объект `birth` во временной ряд очень просто:

```
birth.ts <- ts(birth, start = c(1946, 1), frequency = 12)
```

В приведенной команде аргумент `start` был использован для указания даты, с которой начинается временной ряд `birth.ts` (1946 год, 1-й месяц). Дополнительный аргумент `frequency` (частота) позволяет задать шаг приращения последующих дат – в рассматриваемом примере год разбивается на 12 промежутков, так что шаг приращения составляет 1 месяц. Созданный таким образом объект `birth.ts` при просмотре внешне напоминает матрицу. При этом строкам и столбцам этой матрицы были автоматически (исходя из значений аргументов `start` и `frequency`) присвоены соответствующие имена (данные по столбцам `Oct`, `Nov`, `Dec` для экономии места опущены):

```
birth.ts
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep
1946 26.663 23.598 26.931 24.740 25.806 24.364 24.477 23.901 23.175
1947 21.439 21.089 23.709 21.669 21.752 20.761 23.479 23.824 23.105
1948 21.937 20.035 23.590 21.672 22.222 22.123 23.950 23.504 22.238
1949 21.548 20.000 22.424 20.615 21.761 22.874 24.104 23.748 23.262
1950 22.604 20.894 24.677 23.673 25.320 23.583 24.671 24.454 24.122
1951 23.287 23.049 25.076 24.037 24.430 24.667 26.451 25.618 25.014
1952 23.798 22.270 24.775 22.646 23.988 24.737 26.276 25.816 25.210
1953 24.364 22.644 25.565 24.062 25.431 24.635 27.009 26.606 26.268
1954 24.657 23.304 26.982 26.199 27.210 26.122 26.706 26.878 26.152
1955 24.990 24.239 26.721 23.475 24.767 26.219 28.361 28.599 27.914
1956 26.217 24.218 27.914 26.975 28.527 27.139 28.982 28.169 28.056
1957 26.589 24.848 27.543 26.896 28.878 27.390 28.065 28.141 29.048
1958 27.132 24.924 28.963 26.589 27.931 28.009 29.229 28.759 28.405
1959 26.076 25.286 27.660 25.951 26.398 25.565 28.865 30.000 29.261
```

Функция **`is.ts`**() позволяет проверить, действительно ли созданный нами объект `birth.ts` является временным рядом:

```
is.ts(birth.ts)
[1] TRUE
```

В R имеется достаточно большой набор методов для работы с объектами класса ts. В частности, при помощи функции **plot**() можно быстро изобразить временной ряд графически (рис. 10):

```
plot(birth.ts, xlab = "", ylab = "Рождаемость, тыс. чел.")
```

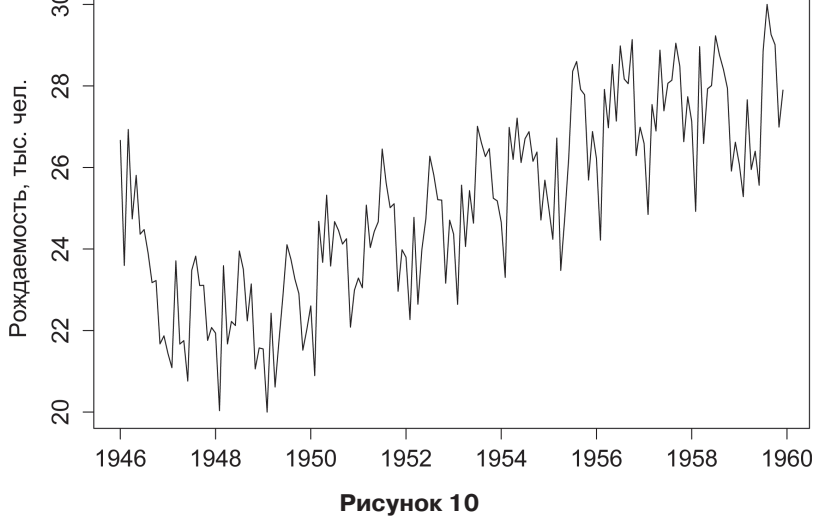


Рисунок 10

Существует большое количество R-пакетов для работы с временными рядами. С обзором основных из них можно ознакомиться на сайте CRAN Task View (http://bit.ly/1zafWnW).

2.7. Организация вычислений: функции, ветвления, циклы

Абсолютное большинство процедур обработки данных в R реализуется с помощью функций. Функции представляют собой поименованный программный код, состоящий из некоторого набора переменных, констант, операторов и других функций и предназначенный для выполнения конкретных операций и задач. Как правило (но не всегда), функции возвращают результат своего выполнения в виде объекта языка R – переменной определенного класса: вектора, списка, таблицы и т. д.

По своему назначению функции можно разделить на характерные группы: арифметические, символьные, статистические и прочие. Функции могут быть встроенными (то есть представленными в базовых или подгружаемых пакетах) и собственными (то есть написанными непосредственно самими пользователями). Примеры некоторых наиболее употребительных встроенных функций представлены ниже:

Вызов функции и описание	Пример и результат
Арифметические функции	
abs(x) – модуль величины x	**abs**(-1) ⇒ 1
ceiling(x) – округление до целого в большую сторону	**ceiling**(9.435) ⇒ 10
floor(x) – округление до целого в меньшую сторону	**floor**(2.975) ⇒ 2
round(x, digits = n) – округление до указанного числа digits знаков после десятичного разделителя дроби	**round**(5.475, 2) ⇒ 5.48
signif(x, digits = n) – округление до указанного числа digits значащих цифр	**signif**(3.475, 2) ⇒ 3.5
trunc(x) – округление до целого числа	**trunc**(4.99) ⇒ 4
exp(x) – e^x	**exp**(2.87) ⇒ 17.637
log(x) – логарифм натуральный x **log10**(x) – логарифм десятичный x	**log**(3.12) ⇒ 1.137 **log**(3.12) ⇒ 0.494
sqrt(x) – корень квадратный x	**sqrt**(2.12) ⇒ 1.456
cos(x), **sin**(x), **tan**(x), **acos**(x), **cosh**(x), **acosh**(x) – тригонометрические функции от x	**cos**(1.27*pi) ⇒ -0.661
Функции для работы с символьными типами данных	
grep(pattern, x, ignore.case = FALSE, fixed = FALSE) – возврат индекса первого найденного элемента pattern в x	**grep**("A", c("x", "y", "A", "z"), fixed = TRUE) ⇒ 3
substr(x, start = n1, stop = n2) – выбор или замена символов в строках символьного вектора x	**substr**("язык R", 2, 4) ⇒ "зык"
paste(..., sep="") – объединение символов или строк через значение разделителя sep	**paste**("x", 1:3, sep="") ⇒ "x1" "x2" "x3"
strsplit(x, split) – разделяет элементы вектора по разделителям split	**strsplit**("абв", "") ⇒ "а" "б" "в"
toupper(x) и **tolower**(x) – преобразуют буквы текстового вектора x в прописные и обратно	**toupper**("Мал") ⇒ "МАЛ" **toupper**("БАЛ") ⇒ "бал"

Примеры статистических функций будут нам встречаться практически во всех последующих главах.

Написание собственных функций

Тремя характерными чертами языка R как языка высокого уровня являются модульность построения, ориентация на работу с объектами и векторизация вычислений. Под модульностью понимается широкое использование групп выражений и функций. Выражения expr, состоящие из объектов данных, вызовов функций и операторов языка, могут группироваться в фигурных скобках: {expr_1; ...; expr_m}. Значение, которое возвращает эта группа, представляет собой результат выполнения последнего выражения. Поскольку такая группа также является выражением, то она может быть, например, включена в круглые скобки и использоваться как часть еще более общего выражения.

Представленная ниже группа команд выполняет расчет среднего и стандартного отклонения натурального ряда чисел от 1 до 10 и возвращает вектор из этих значений:

```
{aver <- mean(1:10); stdev <- sd(1:10); c(MEAN=aver, SD=stdev)}
   MEAN      SD
5.50000 3.02765
```

Однако если этот расчет необходимо выполнить неоднократно для различных наборов исходных данных, то его стоит оформить в виде функции. Общий синтаксис оформления любой функции таков:

```
имя_функции <-  function(arg1,  arg2, ...)   {
группа_выражений
return(object) },
```

где `имя_функции` – имя создаваемой функции, `arg1`, `arg2`, `...` – аргументы функции, а **`return`**`()` возвращает результаты вычислений, сохраненных в объекте `object`. В зависимости от хода выполняемых вычислений, использовать **`return`**`()` не обязательно (см. ниже).

Перед своим первым выполнением функция должна быть определена в текущем скрипте либо загружена с помощью команды **`source`**`()` из внешнего скриптового файла.

Для представленного выше примера можно оформить функцию (обратите внимание, что **`return`**`()` здесь не используется – результатом вычислений автоматически станет вектор из `MEAN` и `SD`, поскольку он создается последним)

```
stat_param <- function(x){
aver <- mean(x); stdev <- sd(x); c(MEAN = aver, SD = stdev)}
```

и включить ее в коллекцию собственных функций, хранящихся, например, в файле `my_func.R`. Тогда необходимый нам результат, приведенный выше, можно получить, выполнив

```
source("my_func.R")
stat_param(1:10)
```

Аргументы функций могут быть обязательными и необязательными. Например, следующая функция возводит числовой объект `x` в степень `n`, но если степень не указана, то автоматически происходит возведение в куб:

```
power <- function(x, n = 3){ x^n }
```

Аргументами функций могут быть объекты самого разного типа, например названия других функций. Так, следующая функция выполняет произвольные преобразования случайных равномерно распределенных величин:

```
my_exampl <- function(n, func_trans)
{ x <- runif(n); abs(func_trans(x)) }
```

Тогда сгенерировать 5 прологарифмированных значений можно, выполнив

```
my_exampl(5, log)
[1] 0.6345459 0.6522557 2.4180118 0.1007311 1.6983938
```

Условия и циклы

Как и в любом языке программирования, в R широко используются ветвления и циклы вычислительного процесса. Оператор условного выполнения команд `if` имеет следующую структуру:

```
if( логическое_выражение )
{ группа_выражений_1, если логическое_выражение равно TRUE }
else { группа_выражений_2 в противном случае }
```

Например, следующая функция сравнивает размеры двух векторов:

```
compare <- function(x, y){ nl <- length(x); n2 <- length(y)
if(nl!= n2){
if(nl > n2){ z = (nl - n2)
cat("Первый вектор имеет на ", z, " элементов больше \n") }
else{ z = (n2 - nl)
cat("Второй вектор имеет на ", z, " элементов больше \n")}}
else{cat("Количество элементов одинаково ", nl, "\n") }
}

x <-  c(1:4)
y <-  c(1:9)

compare(x,  y)
Первый вектор имеет на 5 элементов больше
```

Имеется также сокращенная форма реализации ветвлений:

```
ifelse(логическое_выражение,
       группа_выражений_1, группа_выражений_2)
```

Повторение в цикле одних и тех же вычислительных операций осуществляется с использованием конструкций **for**(), **while**() или **repeat**(), которые имеют следующий синтаксис:

```
for (index in for_object) { группа_выражений }
while(логическое_выражение) { группа_выражений }
repeat { группа_выражений; break }
```

Здесь объект `for_object` может быть вектором, массивом, таблицей или списком, а `группа_выражений` выполняется каждый раз для каждого элемента `index` этого объекта.

Рассмотрим в качестве примера функцию оценки доверительного интервала среднего значения для выборки размером *n* с использованием непараметрического бутстрепа. Необходимо отметить, что устоявшегося перевода термина «*bootstrap*» с английского языка на русский не существует. Используются разные варианты: «бутстреп», «бутстрэп», «бутстрап», «размножение выборок», «метод псевдовыборок» и даже «ресамплинг» (от англ. «*resampling*»). Несмотря на сложности с русскоязычным названием, суть метода тем менее весьма проста и подробно из-

ложена в оригинальных работах Б. Эфрона (Efron, 1979–1988). Бутстреп-метод с использованием R для решения различных задач представлен в ряде монографий (Chernick & LaBudde, 2011; Zieffer et al. 2011; Fox & Weisberg, 2011; Шитиков, Розенберг, 2014).

Предположим, что у нас есть выборка некоторого ограниченного объема, и мы имеем основания полагать, что эта выборка является репрезентативной (то есть хорошо отражает свойства генеральной совокупности, из которой она была взята). Идея бутстреп-метода заключается в том, что мы можем рассматривать саму эту выборку в качестве «генеральной совокупности» и, соответственно, можем извлечь большое число случайных выборок из этой исходной совокупности для расчета интересующего нас параметра (или параметров). Очевидно, что благодаря случайному процессу формирования этих новых выборок будет наблюдаться определенная вариация значений оцениваемого параметра. Другими словами, мы получим некоторое распределение значений этого параметра. Рассчитав стандартное отклонение этого распределения, мы получим оценку стандартной ошибки параметра, которая при большом числе наблюдений будет асимптотически приближаться к истинной стандартной ошибке. Аналогично можно получить оценки границ доверительного интервала.

Итак, будем генерировать из исходной выборки множество псевдовыборок того же размера, состоящих из случайных комбинаций исходного набора элементов. При этом используем алгоритм «случайного выбора с возвратом» («*random sampling with replacement*»), то есть извлеченный элемент возвращается в исходную совокупность и имеет шанс быть выбранным снова. В результате некоторые элементы в каждой отдельной псевдовыборке могут повторяться два или более раз, тогда как другие – отсутствовать. Этот алгоритм в R реализован в функции **sample**(data, replace = T). Для каждой псевдовыборки мы рассчитаем значение среднего, а в качестве границ 95%-го доверительного интервала («*bootstrap percentile interval*») примем 2.5% и 97.5% квантили бутстреп-распределения:

```
boot_np  <- function(data, Nboot=5000) {
 boots <- numeric(Nboot) # Пустой вектор для хранения результатов
  for (i in 1:Nboot) {boots[i] <- mean(sample(data, replace=T))}
  CI <-  quantile(boots, prob = c(0.025, 0.975))
  return (c(m = mean(data), CI))
}

x <- c(5, 5, 8, 10, 10, 10, 19, 20, 20, 20, 30, 40, 42, 50, 50)
boot_np(x)
    m  2.5% 97.5%
 22.6  15.0  30.8
```

Здесь конструкция **for**() осуществляет формирование Nboot = 5000 значений средних для генерируемых псевдовыборок.

Покажем, пользуясь случаем, как можно оценить выборочный доверительный интервал обычным параметрическим методом на основе процентилей распределения Стьюдента:

```
param_CI <- function(data) {
   n = length(data); m = mean(data)
   SE = sd(data)/sqrt(n); E = qt(.975, df = n-1)*SE
   CI <- m + c(-E, E)
   return (c(m , CI))
}
param_CI(x)
[1] 22.6  13.74741  31.45259
```

2.8. Векторизованные вычисления в R

Использование оператора циклов **for**(), что характерно для языков низкого уровня типа Basic, не считается хорошим стилем программирования на языке R, ориентированном на векторизацию вычислений. Вместо того чтобы выполнять последовательно скалярные операции над каждым из элементов массива, гораздо эффективнее выполнять параллельные вычисления, при которых программа обрабатывает одновременно весь массив (вектор) целиком или по нескольку элементов вектора в каждый момент времени. Очевидно, что такой подход потенциально может привести к значительному ускорению однотипных вычислений над большими массивами данных.

Рассмотрим простейший пример векторизованных вычислений в R. Допустим, у нас имеется вектор из 10 положительных чисел, и мы хотим извлечь квадратный корень из каждого из них. Вместо написания цикла для поочередного выполнения этой операции над каждым элементом мы просто подаем *весь* этот вектор на вход функции **sqrt**(), которая возвращает *вектор* с результатами вычислений:

```
x <- 1:10
sqrt(x)
[1] 1.000 1.414 1.732 2.000 2.236 2.449 2.645 2.828 3.000 3.162
```

Принцип векторизованных вычислений применим не только к векторам как таковым, но и к более сложным объектам R – матрицам, спискам и таблицам данных. В базовой комплектации R имеется целое семейство функций, предназначенных для организации векторизованных вычислений над такими объектами. В названии всех этих функций имеется слово apply («применить»), которому предшествует буква, указывающая на принцип работы той или иной функции (см. подробнее в справочном файле ?apply). При этом:

- **apply**(), в отличие от **for**(), можно легко распараллелить, задействовав все имеющиеся процессоры (путем использования «параллельной» версии этой функции из пакета snow);
- алгоритмы, записанные без циклов, легче модифицируются, содержат меньше ошибок и легче набираются в командной строке;
- результат работы **apply**() может быть аргументом функции, не говоря уже о том, что любая функция может быть вставлена в качестве аргумента в **apply**().

В ответе на один из вопросов, опубликованных на сайте StackOverflow (http://bit.ly/1vH9dQQ), который мы регулярно используем для поиска информации по R, был дан замечательный обзор apply-функций с примерами их использования. Ниже приводится перевод этого сообщения (с некоторыми изменениями и дополнениями).

Функция **apply**() применяется в случаях, когда необходимо применить какую-либо функцию ко всем строкам или столбцам матрицы (или массивам большей размерности):

```
apply(x, MARGIN, FUN, ...),
```

где x – это преобразуемый объект, MARGIN – индекс, обозначающий направление процесса вычислений (по столбцам или строкам), FUN – применяемая для вычислений функция, а ... – это любые параметры, управляющие поведением функции FUN. Для матрицы или таблицы данных MARGIN = 1 обозначает строки, а MARGIN = 2 – столбцы. Поскольку FUN означает любую функцию R, в том числе и ту, которую вы сами написали (см. раздел 2.7), функция **apply**() – это мощное средство модульной обработки данных.

```
# Создадим обычную матрицу:
M <- matrix(seq(1, 16), 4, 4)

# Найдем минимальные значения в каждой строке матрицы
apply(M, 1, min)
[1] 1 2 3 4

# Найдем минимальные значения в каждом столбце матрицы
apply(M, 2, max)
[1]  4  8 12 16

# Пример с трехмерным массивом:
M <- array(seq(32), dim = c(4,4,2))

# Применим функцию sum() к каждому элементу M[*, ,],
# то есть выполним суммирование по измерениям 2 и 3:
apply(M, 1, sum)
# Результат - одномерный вектор:
[1] 120 128 136 144

# Применим функцию sum() к каждому элементу  M[*, *, ],
#  то есть выполним суммирование по третьему измерению:
apply(M, c(1,2), sum)
# Результат - матрица:
     [,1] [,2] [,3] [,4]
[1,]   18   26   34   42
[2,]   20   28   36   44
[3,]   22   30   38   46
[4,]   24   32   40   48
```

При необходимости вычисления сумм и средних значений по строкам или столбцам матриц рекомендуется также использовать специально оптимизированные для этого функции **colSums**(), **rowSums**(), **colMeans**() и **rowMeans**().

Функция **lapply**() используется в случаях, когда необходимо применить какую-либо функцию к каждому компоненту списка и получить результат также в виде списка (буква «l» в названии **lapply**() означает «*list*» – список).

```
# Создадим список с тремя компонентами-векторами:
x <- list(a = 1, b = 1:3, c = 10:100)

# Выясним размер каждого компонента списка x
#  (функция length()):
lapply(x, FUN = length)
$a
[1] 1
$b
[1] 3
$c
[1] 91

# Выполним суммирование элементов в каждом компоненте списка x:
lapply(x, FUN = sum)
$a
[1] 1
$b
[1] 6
$c
[1] 5005
```

Функция **sapply**() используется в случаях, когда необходимо применить какую-либо функцию к каждому компоненту списка, но результат вывести в виде вектора (буква «s» в названии **sapply**() означает «*simplify*» – упрощать).

```
# Список из трех компонентов:
x <- list(a = 1, b = 1:3, c = 10:100)

# Выясним размер каждого компонента списка x:
sapply(x, FUN = length)
a  b  c
1  3 91 # результат возвращен в виде вектора

# Суммирование всех элементов в каждом компоненте списка x:
sapply(x, FUN = sum)
a    b    c
1    6 5005 # результат возвращен в виде вектора
```

В некоторых более «продвинутых» ситуациях **sapply**() может выдать результат в виде многомерного массива. Например, если применяемая нами функция воз-

вращает векторы одинаковой длины, **sapply**() объединит эти векторы в матрицу (по столбцам):

```
sapply(1:5, function(x) rnorm(3, x))
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 1.699876 1.898511 2.655672 2.908397 3.841733
[2,] 1.309984 3.279127 2.720922 4.794944 4.723460
[3,] 1.755939 2.780790 2.781063 4.193394 4.220807
```

Если применяемая функция возвращает матрицу, то **sapply**() преобразует каждую матрицу в вектор и объединит такие векторы в одну большую матрицу (звучит сложно, но пример хорошо поясняет эту идею):

```
sapply(1:5, function(x) matrix(x, 2, 2))
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    1    2    3    4    5
[3,]    1    2    3    4    5
[4,]    1    2    3    4    5
```

Поведение **sapply**(), продемонстрированное в последнем примере, можно отменить при помощи аргумента `simplify = "array"` – в этом случае матрицы будут объединены в один многомерный массив:

```
sapply(1:5, function(x) matrix(x, 2, 2), simplify = "array")
, , 1
     [,1] [,2]
[1,]    1    1
[2,]    1    1
, , 2
     [,1] [,2]
[1,]    2    2
[2,]    2    2
, , 3
     [,1] [,2]
[1,]    3    3
[2,]    3    3
, , 4
     [,1] [,2]
[1,]    4    4
[2,]    4    4
, , 5
     [,1] [,2]
[1,]    5    5
[2,]    5    5
```

Функция **replicate**() является своего рода обобщением функции **sapply**() и позволяет провести серию вычислений с целью генерации набора чисел по заданному алгоритму. Синтаксис функции имеет вид:

```
replicate(n, expr, simplify = TRUE),
```

где n – число повторов, expr – функция или группа выражений, которые надо повторить n раз, simplify = TRUE - необязательный параметр, использование которого позволяет (по возможности) упростить результат и представить его в виде вектора или матрицы значений.

Рассмотрим пример использования функции **replicate**() для проверки статистической гипотезы о равенстве медиан двух выборок бутстреп-методом (см. раздел 2.7). Созданная нами ранее функция **boot_med**() извлекает из каждой исходной выборки x и y по N псевдовыборок, используя алгоритм «случайного выбора с возвратом», и находит разность их медиан. Здесь функция **sample.int**() формирует целочисленные наборы случайных индексов indx и indy для каждого из сравниваемых векторов:

```
boot_med <- function(x, y, N=100) {
  replicate(N, {
    indx <- sample.int(length(x), length(x), replace = T)
    indy <- sample.int(length(y), length(y), replace = T)
    median(x[indx]) - median(y[indy])
  })
}
```

Для тестирования функции проверим равенство медиан двух случайных выборок из нормального распределения со средним mean = 10 и стандартным отклонением sd = 4:

```
Y <- rnorm(100, sd = 4, mean = 10)
X <- rnorm(100, sd = 4, mean = 10)
quantile(boot_med(Y, X, 10000), probs = c(0.025, 0.975))
      2.5%      97.5%
-0.3834803  2.5177006
```

Поскольку доверительный интервал разности медиан включает 0, то нулевую гипотезу об отсутствии разницы между медианами генеральных совокупностей отклонять не следует.

Функция **vapply**() схожа с **sapply**(), но работает несколько быстрее за счет того, что пользователь однозначно указывает тип возвращаемых значений (буква «v» в названии **vapply**() означает «*velocity*» – скорость; пример сравнения **sapply**() и **vapply**() можно найти на одной из страниц StackOverflow – http://bit.ly/1E0XQV4). Такой подход позволяет также избегать сообщений об ошибках (и прерывания вычислений), возникающих при работе с **sapply**() в некоторых ситуациях. При вызове **vapply**() пользователь должен привести пример ожидаемого типа возвращаемых значений. Для этого служит аргумент FUN.VALUE:

```
# Аргументу FUN.VALUE присвоено логическое значение FALSE.
# Этим задается тип возвращаемых функцией значений,
# который ожидает пользователь
a <- vapply(NULL, is.factor, FUN.VALUE = FALSE)
```

```
# Функция sapply() применена к тому же NULL-объекту:
b <- sapply(NULL, is.factor)

# Проверка типа переменных:
is.logical(a)
[1] TRUE
is.logical(b)
[1] FALSE
```

Функция **mapply**() используется в случаях, когда необходимо поэлементно применить какую-либо функцию одновременно к нескольким объектам (например, получить сумму первых элементов векторов, затем сумму вторых элементов векторов и т. д.). Результат возвращается в виде вектора или массива другой размерности (см. примеры для **sapply**() выше). Буква «m» в названии **mapply**() означает «*multivariate*» – многомерный (имеется в виду одновременное выполнение вычислений над элементами нескольких объектов).

```
mapply(sum, 1:5, 1:5, 1:5)
[1]  3  6  9 12 15
```

Функция **rapply**() используется в случаях, когда необходимо применить какую-либо функцию к компонентам вложенного списка (буква «r» в названии **rapply**() означает «*recursively*» – рекурсивно).

```
# Пользовательская функция, которая добавляет ! к элементу
# объекта x, если этот элемент является текстовым выражением,
# или добавляет 1, если этот элемент является числом:
myFun <- function(x){
    if (is.character(x)){
    return(paste(x, "!", sep = ""))
    }
    else{
    return(x + 1)
    }
}

# Пример вложенного списка:
l <- list(a = list(a1 = "Boo", b1 = 2, c1 = "Eeek"),
          b = 3, c = "Yikes",
          d = list(a2 = 1, b2 = list(a3 = "Hey", b3 = 5)))
l
$a
$a$a1
[1] "Boo"
$a$b1
[1] 2
$a$c1
[1] "Eeek"
$b
```

```
[1] 3
$c
[1] "Yikes"
$d
$d$a2
[1] 1
$d$b2
$d$b2$a3
[1] "Hey"
$d$b2$b3
[1] 5

# Рекурсивное применение функции myFun к списку l:
rapply(l, myFun)
    a.a1     a.b1     a.c1        b        c     d.a2  d.b2.a3  d.b2.b3
  "Boo!"      "3"  "Eeek!"      "4" "Yikes!"      "2"   "Hey!"      "6"

# Если необходимо вернуть результат в виде вложенного
# списка, можно воспользоваться аргументом how = "replace".
# В этом случае исходные значения в списке l будут
# заменены на новые:
rapply(l, myFun, how = "replace")
$a
$a$a1
[1] "Boo!"
$a$b1
[1] 3
$a$c1
[1] "Eeek!"
$b
[1] 4
$c
[1] "Yikes!"
$d
$d$a2
[1] 2
$d$b2
$d$b2$a3
[1] "Hey!"

$d$b2$b3
[1] 6
```

Функция **tapply**() применяется, когда необходимо применить какую-либо функцию fun к отдельным группам элементов вектора x, заданным в соответствии с уровнями какого-либо фактора group:

```
tapply(x, group, fun,   ...)
```

Например, в следующем фрагменте кода функция **sample**() используется для создания двух случайных выборок из 50 значений целых чисел от 1 до 4 и связанных с ними меток четырех групп A–D. Функция **tapply**() подсчитывает суммы x для каждого из уровней фактора:

```
x <- sample(1:4, size = 50, replace = T)
gr <- as.factor(sample(c("A","B","C","D"), size = 50, replace = T))
tapply(x, gr,sum)
 A   B   C   D
29  25  30  37
```

Другие примеры использования **tapply**() мы приведем позднее в разделе 4.1, посвященном расчету параметров описательной статистики.

Функция **by**() является своего рода аналогом функции **tapply**(), с той разницей, что она применяется для таблиц. Таблица data разделяется в соответствии с уровнями некоторого столбца-фактора group на отдельные подтаблицы, и для обработки каждой такой части определяется функция fun:

```
by(data, group, fun,   ...)
```

Рассмотрим пример с таблицей mussel, представленной в разделе 1.3, и рассчитаем средние значения длины раковины дрейссены ZMlength и численности инфузорий CAnumber для каждого из трех обследованных озер Lake (Баторино, Мястро и Нарочь):

```
mussel <- read.table("http://bit.ly/1wMGDOQ",
                     header = TRUE, sep = "\t",
                     na.strings = "NA", strip.white = TRUE)

by(mussel[, 4:5], mussel$Lake, colMeans)
mussel$Lake: Batorino
 ZMlength  CAnumber
 18.61339 772.80315
------------------------
mussel$Lake: Myastro
 ZMlength  CAnumber
 16.36944 803.08889
------------------------
mussel$Lake: Naroch
ZMlength CAnumber
 14.4929 340.9941
```

Функция **outer**() позволяет выполнить комбинаторную операцию fun над элементами двух массивов или векторов x и y, не прибегая к явному использованию «двойного» цикла:

```
outer(x, y, fun = "*", ...)
```

По умолчанию осуществляется операция попарного перемножения:

```
x <- 1:5; y <- 1:5
outer(x,y)
 [,1] [,2] [,3] [,4] [,5]
[1,]  1 2 3 4 5
[2,]  2 4 6 8 10
[3,]  3 6 9 12  15
[4,]  4 8 12  16  20
[5,]  5 10  15  20  25
```

Используя, например, функцию **outer**() вместе с функцией **paste**(), можно сгенерировать все возможные попарные комбинации элементов символьного и целочисленного векторов:

```
x  <-   c("A", "B", "C", "D")
y  <-   1:10
outer(x, y, paste, sep = "")
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9]  [,10]
[1,]  "A1"  "A2"  "A3"  "A4"  "A5"  "A6"  "A7"  "A8"  "A9"  "A10"
[2,]  "B1"  "B2"  "B3"  "B4"  "B5"  "B6"  "B7"  "B8"  "B9"  "B10"
[3,]  "C1"  "C2"  "C3"  "C4"  "C5"  "C6"  "C7"  "C8"  "C9"  "C10"
[4,]  "D1"  "D2"  "D3"  "D4"  "D5"  "D6"  "D7"  "D8"  "D9"  "D10"
```

Глава 3

Базовые графические возможности R

Графическое представление данных играет очень важную роль в статистике. Как сказано в известной книге Дж. Чемберса с соавторами (Chambers et al., 1983): «*...нет статистического метода более мощного, чем хорошо подобранный график*». Так, графики являются неотъемлемой частью разведочного анализа данных, позволяют выявлять закономерности и тренды в сложных наборах данных, а также могут быть непосредственно результатом статистического анализа (например, при построении деревьев классификации).

Читатель, интересующийся спектром графических возможностей R, может посетить сайт R Graph Gallery (http://bit.ly/1E6Lkn1), где представлены не только примеры всевозможных графиков, но и исходный код, использованный для их построения.

Как правило, создание графика начинается с некоторой функции высокого уровня, которая определяет его общую структуру: размерность (1D, 2D, 3D), масштабы осей, названия и др. Одни из наиболее часто используемых графических функций высокого уровня – **plot**(), **hist**(), **boxplot**(), **scatterplot**() (из пакета `lattice`) и **pairs**(). С помощью богатого набора функций низкого уровня к построенному графику могут быть добавлены дополнительные элементы: текст, линии, легенда и прочее (примерами таких функций являются **lines**(), **points**(), **text**(), **axis**() и др.).

Настройка отдельных деталей диаграмм выполняется при помощи целого ряда параметров, которые, как правило, одинаковы для большинства высокоуровневых и низкоуровневых функций. Такие параметры, в частности, определяют цвет линий и символов, тип и размер символов, толщину и характер линий, штриховку, рамку графика и прочее.

3.1. Функция `plot()` и ее параметры

Функция **plot**() – главная «рабочая лошадка», используемая для построения графиков в R. Поведение этой базовой *функции высокого уровня* определяется классом подаваемых на нее объектов. Соответственно, с помощью **plot**() можно создать очень большой набор разнотипных графиков.

В качестве примера используем данные Kwan et al. (1976, http://bit.ly/1E7gqwl) по скорости выведения из организма человека индометацина – одного из наиболее активных противовоспалительных препаратов. В эксперименте приняли участие шесть испытуемых. Результаты этого исследования входят в базовый набор данных R и доступны по команде

```
data(Indometh)
```

Применив команду

```
names(Indometh)
[1] "Subject"   "time"   "conc"
```

мы увидим, что в состав таблицы `Indometh` входят переменные `Subject` (испытуемый), `time` (время с момента введения препарата) и `conc` (концентрация препарата в крови). Для облегчения дальнейшей работы прикрепим таблицу `Indometh` к *поисковому пути* R:

```
attach(Indometh)
```

Благодаря этой команде теперь мы можем напрямую обращаться к переменным таблицы `Indometh` (то есть использовать их имена непосредственно – например, `time` вместо `Indometh$time`).

Зависимость концентрации индометацина в крови от времени можно легко изобразить при помощи простой команды **`plot`**`(time, conc)` (рис. 11).

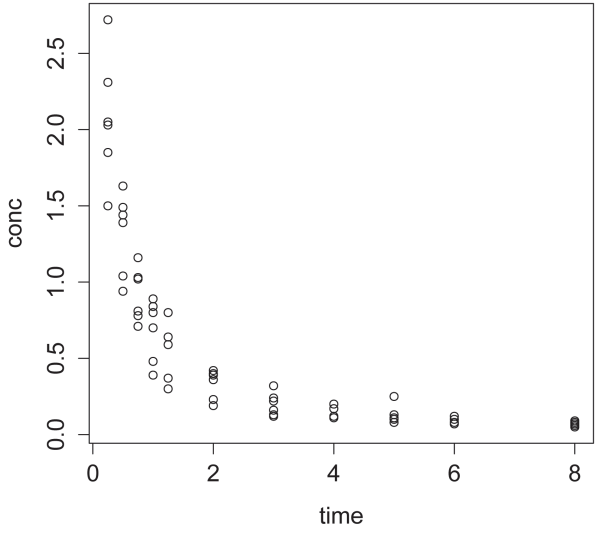


Рисунок 11

Предположим, что перед нами стоит задача отобразить на графике не все исходные данные, а только средние значения концентрации индометацина для каждой

временной точки (рис. 12). Рассчитать средние значения (или любые другие количественные величины) для отдельных групп данных позволяет функция **tapply**() (см. раздел 2.8):

```
(means <- tapply(conc, time, mean))
      0.25        0.5       0.75          1       1.25          2          3
2.07666667 1.32166667 0.91833333 0.68333333 0.55666667 0.33166667 0.19833333
         4          5          6          8
0.13666667 0.12500000 0.09000000 0.07166667
```

Обратите внимание на то, что при создании вектора means функция **tapply**() автоматически присвоила каждому из средних значений имя, соответствующее времени учета концентрации индометацина. Это легко проверить:

```
names(means)
[1] "0.25" "0.5"  "0.75" "1"  "1.25" "2"  "3"  "4"  "5"  "6"  "8"
```

Мы можем воспользоваться этим обстоятельством при построении графика и легко создать числовой вектор со значениями времени учета концентрации препарата:

```
indo.times <- as.numeric(names(means))
# строим график типа "точки с линиями":
plot(indo.times, means, type = "b", xlab = "Время", ylab = "Концентрация")
```

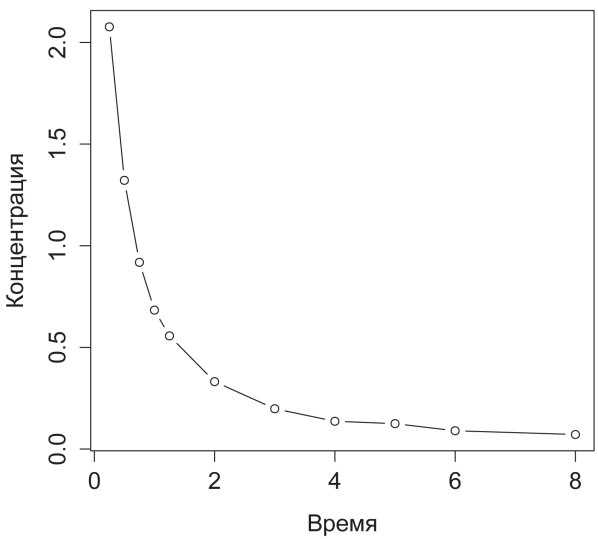


Рисунок 12

Управляющие параметры функции plot()

Функция **plot**() имеет большое количество управляющих параметров, которые позволяют осуществить тонкую настройку внешнего вида графика. Ниже рассмотрены лишь некоторые из них.

Параметры xlab и ylab

Параметры xlab и ylab служат для указания названий осей X и Y соответственно (см. предыдущую команду):

Параметр type

Параметр type позволяет изменять внешний вид точек на графике. Он принимает одно из следующих значений:

- "p" – точки («*points*»; используется по умолчанию);
- "l" – линии («*lines*»);
- "b" – изображаются и точки, и линии («*both points and lines*»);
- "o" – точки изображаются поверх линий («*points over lines*»);
- "h" – гистограмма («*histogram*»);
- "s" – ступенчатая кривая («*steps*»);
- "n" – данные не отображаются («*no points*»).

Параметры xlim и ylim

Эти два аргумента контролируют диапазон значений на каждой из осей графика. По умолчанию они оба принимают значение NULL – в этом случае размах выбирается программой автоматически. Для отмены автоматических настроек соответствующему параметру необходимо присвоить значение в виде числового вектора, содержащего минимальное и максимальное значения, которые должны отображаться на оси. Например:

```
plot(indo.times, means, xlab = "Время",
     ylab = "Концентрация", xlim = c(0, 15))
plot(indo.times, means, xlab = "Время",
     ylab = "Концентрация", ylim = c(0, 5))
```

Параметры axes и ann

Эти два параметра контролируют отображение осей и их названий соответственно. Каждый из них может принимать одно из двух возможных значений – TRUE или FALSE:

```
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     axes = TRUE, ann = TRUE)
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     axes = FALSE, ann = TRUE)
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     axes = TRUE, ann = FALSE)
```

Параметр `log`

При помощи аргумента `log` можно перевести одну или обе оси графика на логарифмическую шкалу, например:

```
plot(indo.times, means, xlab = "Время", ylab = "Концентрация", log = "x")
plot(indo.times, means, xlab = "Время", ylab = "Концентрация", log = "y")
plot(indo.times, means, xlab = "Время", ylab = "Концентрация", log = "xy")
```

Параметр `main`

Аргумент `main` служит для создания заголовка графика. По умолчанию название размещается в верхней части рисунка:

```
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "Скорость выведения индометацина", type = "o")
```

Далее будут рассмотрены такие особенности внешнего вида графиков, как тип, размер и цвет символов и линий, тип и размер шрифта в названиях графика и его осей, использование математических символов в названиях, размещение легенды и т. п. Они применяются в качестве аргументов при вызове не только **plot**(), но и многих других функций.

Общие аргументы графических функций

Тип символа

Как видно по рис. 12, отдельные измерения по умолчанию изображаются в виде кружков. Другие символы можно задать при помощи аргумента `pch` (от «*plotting character*» – символ изображения). В стандартных случаях этот аргумент принимает численные значения от 1 до 25. Например, при `pch = 2` символы превратятся из кружков в незакрашенные треугольники:

```
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "Скорость выведения индометацина",
     type = "o", pch = 2)
```

Таблица 25 стандартных символов и соответствующие им численные коды представлены на рис. 13.

Набор символов, используемых для отображения данных, может быть значительно расширен в случае, когда аргумент `pch` используется в комбинации с другим аргументом – `font`, задающим шрифт. Параметр `pch` может при этом принимать любое целое число от 1 до 128 и от 160 до 254. Например, при `font = 5` маркеру в виде «сердечка» соответствует код 169:

```
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "Скорость выведения индометацина", type = "o",
     pch = 169, font = 5)
```

В качестве маркеров можно также использовать обычные печатные символы, например буквы:

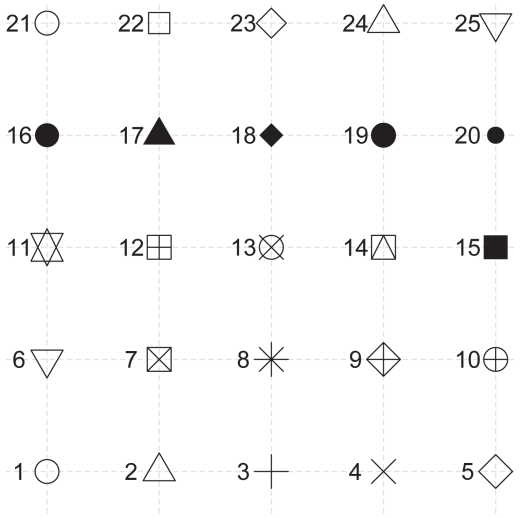


Рисунок 13

```
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "Скорость выведения индометацина", type = "o",
     pch = "A")
```

Размер символа

Размер сиволов задается при помощи аргумента `cex` (от «*character extension*» – размер символа), который по умолчанию равен 1. Уменьшение или увеличение этого параметра приводит к соответствующим пропорциональным изменениям.

При необходимости мы можем также изменить ширину линии обводки символа. Для этого служит параметр `lwd` (от «*line width*» – ширина линии). Далее будет приведен пример для символа с кодом 21 («заполненный кружок»).

Цвет символа

Цвет любого графического объекта может быть задан несколькими способами:

- по названию цвета: например, `col = "red"` (красный), `col = "green"` (зеленый) или `col = "black"` (черный). Всего в R имеется 675 стандартных цветов. Их названия доступны по команде **`colors`**`()`;
- путем непосредственного указания красного, зеленого и синего компонентов RGB-спектра с использованием шестнадцатеричной записи в формате `"#RRGGBB"` (подробнее см. http://bit.ly/1NdqruN и http://bit.ly/1F64wzO);
- по численному коду, например: `col = 2` (красный), `col = 3` (зеленый) или `col = 1` (черный).

Цвет маркеров задается аргументом `col` (от «*color*» – цвет). Номер необходимого цвета можно подобрать, например, при помощи диаграммы, представленной

на рис. 14 (код для этой диаграммы, а также много дополнительной информации о работе с цветовыми палитрами в R можно найти на сайте http://bit.ly/1wisAsW). Полезными функциями для подбора цвета могут оказаться **rgb**(), **col2rgb**(), **palette**(), **terrain.colors**() и некоторые другие.

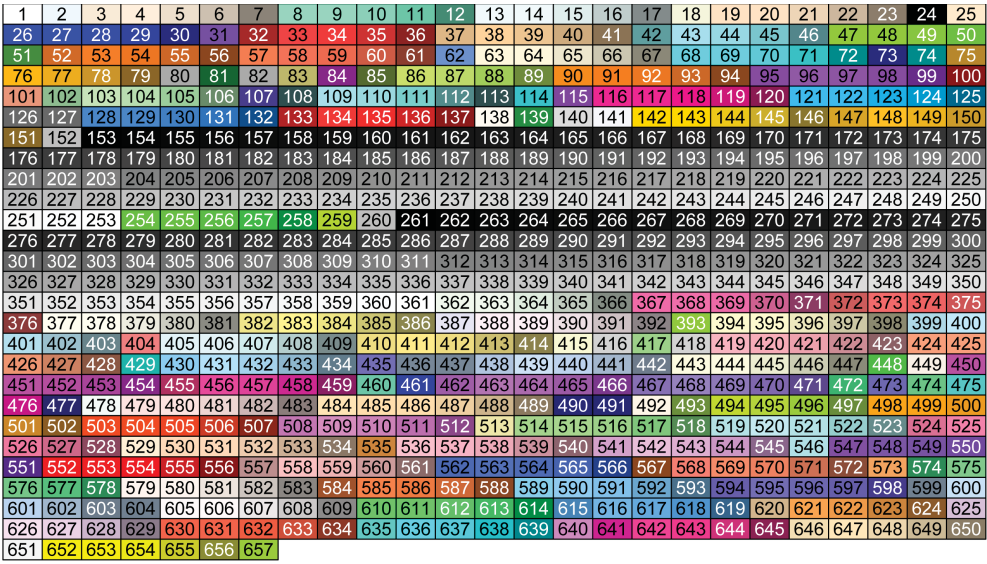


Рисунок 14

Имеются также отдельные аргументы для настройки цвета других элементов графика, таких как заголовок (`col.main`), названия осей (`col.lab`), метки осей (`col.axes`) и др.

Ниже приведено несколько примеров, иллюстрирующих эффекты параметра `col`.

```
# символы в виде синих треугольников: pch = 2, col = "blue"
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "Скорость выведения индометацина", type = "o",
     pch = 2, cex = 1.2, col = "blue")
# символы в виде ромбиков (pch = 5) красного цвета (col = 2)
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "Скорость выведения индометацина", type = "o",
     pch = 5, cex = 1.2, col = 2)
# заголовок графика синего цвета (col.main = "blue")
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "Скорость выведения индометацина", type = "o",
     col.main = "blue", cex = 1.2)
# названия осей выполнены красным цветом (col.lab = "red")
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
```

```
     main = "Скорость выведения индометацина", type = "o",
     col.main = "blue", col.lab = "red", cex = 1.2)
```

При работе с символами 21–25 (см. рис. 13 выше) мы можем использовать аргумент `bg` (от «*background*» – фон) для указания цвета, которым они должны быть закрашены (рис. 15):

```
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "Скорость выведения индометацина", type = "o",
     pch = 21, cex = 1.2, bg = "red", lwd = 2, col.main = "blue")
```

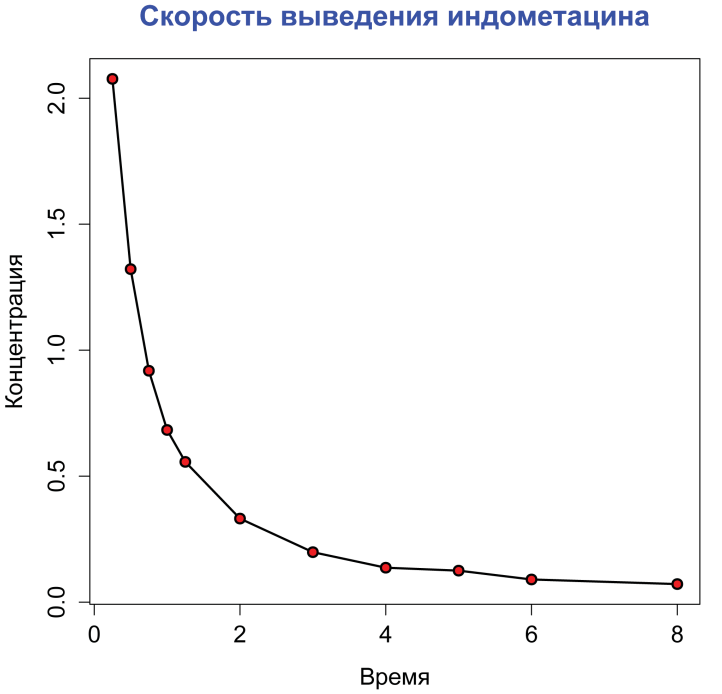


Рисунок 15

Ширина линии

Ширина линии задается при помощи упомянутого выше аргумента `lwd` функции **`plot`**`()`. Этот аргумент принимает положительные числовые значения, показывающие, во сколько *раз* ширина линии должна быть больше относительно ширины, заданной по умолчанию. Ширина линии (по умолчанию равна 1) является безразмерной величиной, поскольку на разных графических устройствах линии с одинаковыми параметрами могут выглядеть по-разному. Ниже приведены примеры трех графиков с разными значениями параметра `lwd`:

```
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "lwd = 2", type = "l", lwd = 2)
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "lwd = 5", type = "l", lwd = 5)
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "lwd = 10", type = "l", lwd = 10)
```

Концы и места соединения линий

Аргумент `lend` (от «*line end*») функции **`plot`**`()` позволяет настроить внешний вид концов линии. Этот аргумент принимает значения 0 (по умолчанию), 1 или 2, что соответствует округлым, усеченным квадратным и квадратным концам соответственно. Места соединения линий также могут выглядеть по-разному, что определяется аргументом `ljoin` (от «*line*» – линия и «*join*» – место соединения). Аргумент `ljoin` принимает значения 0 (по умолчанию), 1 или 2, что соответствует округлому, остроугольному и усеченному соединениям соответственно.

Тип линии

Тип линии настраивается при помощи аргумента `lty` (от «*line*» – линия и «*type*» – тип) функции **`plot`**`()`. Существует шесть предустановленных типов линий, которые задаются числами от 1 до 6 соответственно.

При необходимости можно создать также пользовательские типы линий. В таких случаях в качестве значения аргумента `lty` выступает текстовая последовательность из четырех цифр. Эти числа (от 1 до 9) определяют размер четырех элементов, составляющих повторяющийся паттерн «штрих – пробел – штрих – пробел». Например, при `lty = "4241"` линия будет состоять из повторяющегося паттерна, в котором имеются штрих длиной 4 единицы, пробел длиной 2 единицы, опять штрих длиной 4 единицы и пробел в 1 единицу. Примеры стандартных и пользовательских типов линий приведены на рис. 16.

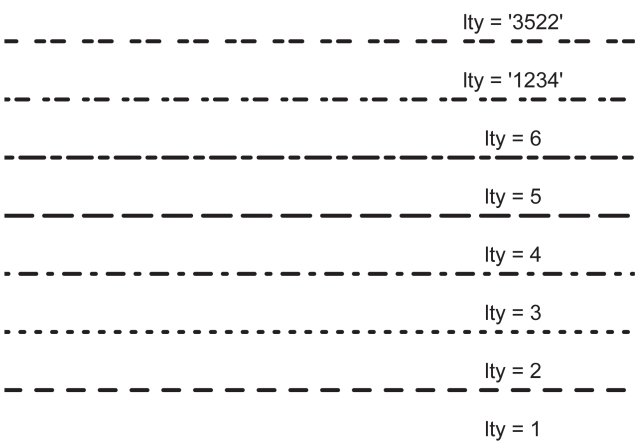


Рисунок 16

Цвет линий задается при помощи аргумента `col`. Использование параметра `col` в отношении линий ничем не отличается от его использования в отношении графических символов.

Рамка графика

Для настройки внешнего вида рамки графика служит аргумент `bty` (от «*box*» – коробка и «*type*» – тип) функции **`plot`**`()`. Этот аргумент принимает одно из следующих шести текстовых значений: `"O"`, `"L"`, `"7"`, `"C"`, `"U"`, `"["`. Рамка будет принимать вид в соответствии с формой указанного символа (допускается использование также строчных букв `o`, `l`, `c` и `u`). Ниже приведен пример использования одной из перечисленных опций (рис. 17).

```
plot(indo.times, means, xlab = "Время", ylab = "Концентрация",
     main = "", type = "l", lwd = 10, lty = 2, col = 6,
     bty = "L")
```

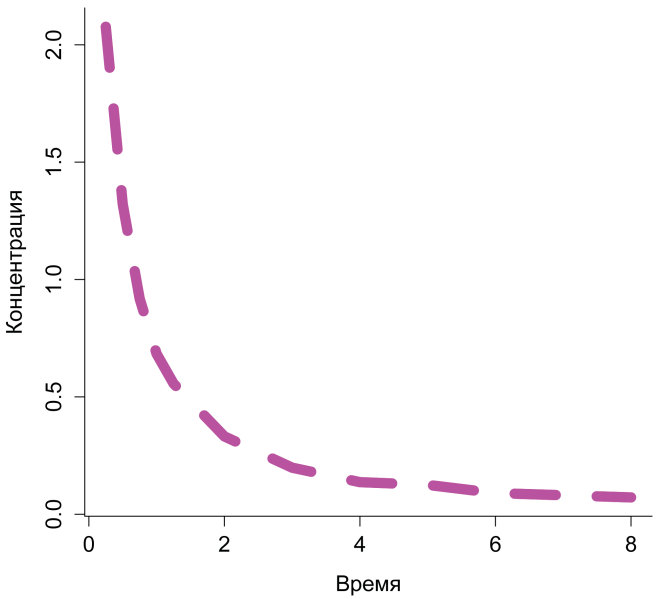


Рисунок 17

3.2. Гистограммы, функции ядерной плотности и функция cdplot()

Гистограмма является важным инструментом статистики, позволяющим наглядно представить эмпирическое распределение значений анализируемой переменной. В системе R для построения гистограмм служит функция **`hist`**`()`. Ее основным аргументом выступает имя анализируемой переменной.

В качестве примера создадим нормально распределенную совокупность X из 100 наблюдений со средним значением 15 и стандартным отклонением 5:

```
X <- rnorm(n = 100, mean = 15, sd = 5)
```

Для создания переменной X использована функция **rnorm**() (от «*random*» – случайный и «*norm*» – нормальный). Используя генератор (псевдо)случайных чисел, эта функция формирует нормально распределенные совокупности с заданными размером n, средним значением mean и стандартным отклонением sd. Изобразить значения переменной X в виде гистограммы очень просто (рис. 18):

```
hist(X)
```

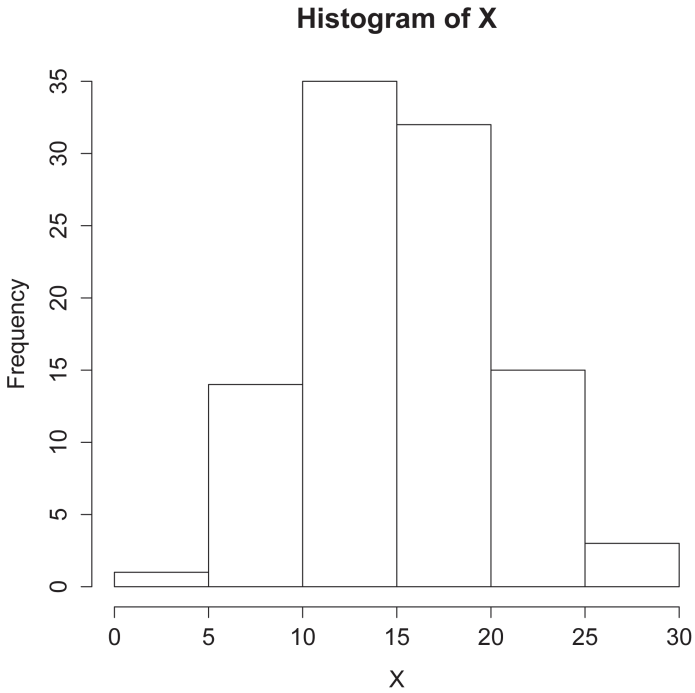


Рисунок 18

Как видно по рис. 18, функция **hist**() автоматически подбирает количество столбцов для отображения на графике, а также создает названия осей и заголовок графика. Такого рисунка, получаемого с использованием автоматических настроек, может оказаться вполне достаточно (например, при проведении быстрого разведочного анализа данных). Однако часто требуется его дополнительная доработка.

Прежде всего важно обратить внимание на размер шага, используемого для разбиения данных на классы при построении гистограммы. В приведенном выше примере программа разбила значения переменной X на 6 классов. Однако такое

грубое разбиение может недостаточно точно отражать свойства анализируемой совокупности. Для более детального изучения этих свойств можно увеличить дробность деления данных на классы (то есть использовать меньший классовый промежуток). Сделать это позволяет аргумент `breaks` (разломы) функции **`hist`**`()`. При необходимости столбцы гистограммы можно залить желаемым цветом. Для этого следует воспользоваться аргументом `col` – это тот же аргумент, который мы использовали при рассмотрении настроек функции **`plot`**`()` (раздел 3.1). В приведенном ниже примере выбран светло-голубой цвет столбцов (`lightblue`):

```
hist(X, breaks = 20, col = "lightblue")
```

Как видим, результатом выполнения предыдущей команды стала гистограмма с двадцатью столбцами (рис. 19), позволяющая более детально проанализировать распределение значений переменной `X`.

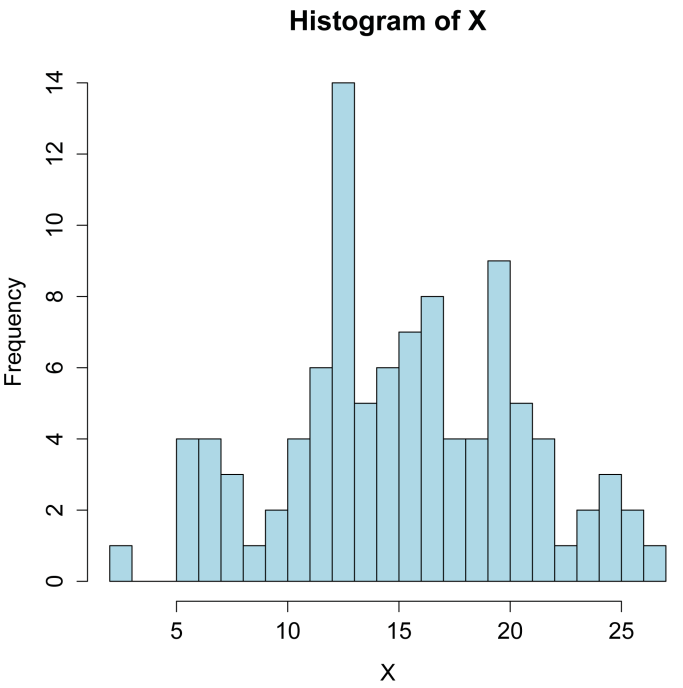


Рисунок 19

По умолчанию функция **`hist`**`()` отображает по оси ординат частоты встречаемости каждого класса значений `X`. Такое поведение функции можно изменить, придав аргументу `freq` (от «*frequency*» – частота) значение `FALSE`. В этом случае ось ординат будет отражать плотность вероятности каждого класса так, что суммарная площадь под гистограммой составит 1:

```
hist(X, breaks = 20, freq = FALSE)
```

В ряде случаев, в частности при небольшом числе наблюдений, гистограммы могут давать неверное представление о свойствах совокупности, например из-за небольшого числа редко расположенных столбцов:

```
X <- rnorm(n = 50, mean = 15, sd  = 5)
hist(X, breaks = 20, freq = FALSE, col = "lightblue")
```

Вместо гистограммы (или параллельно с ней) в таких случаях рекомендуется воспользоваться *кривой плотности вероятности*. Оценка плотности вероятности выполняется при помощи функции **density**(), которую можно применить в качестве аргумента функции **plot**() для графического изображения результата:

```
plot(density(X))
```

Гладкость получаемой кривой регулируется при помощи аргумента bw (от «*band*width» – ширина окна), например:

```
plot(density(X, bw = 0.8))
```

Для полноты картины гистограмму можно совместить с кривой плотности вероятности (рис. 20). При этом сначала необходимо построить саму гистограмму, а затем добавить к ней кривую плотности при помощи функции **lines**() (подробнее эту функцию мы рассмотрим позднее):

```
hist(X, breaks = 20, freq = FALSE, col = "lightblue",
       xlab = "Переменная X",
       ylab = "Плотность вероятности",
       main = "Гистограмма, совмещенная с кривой плотности")
lines(density(X), col = "red", lwd = 2)
lines(density(X, bw = 0.8), col = "blue", lwd = 2)
```

Обратите внимание на дополнительные аргументы, использованные при вызове функциии **hist**(): xlab и ylab – для создания названий осей и main – для создания заголовка рисунка. В случае с функцией **lines**() были применены аргументы col (для назначения цвета линии) и lwd (для установки толщины линии). Эти же аргументы мы использовали ранее при построении графиков с помощью функции **plot**(). Приведенные примеры показывают универсальность этих и целого ряда других аргументов, управляющих поведением **plot**(), **hist**() и иных графических функций R высокого уровня.

Особенности использования функции **density**() для визуализации группированных данных рассмотрим на примере данных, полученных в ходе эксперимента по изучению эффективности шести видов инсектицидных средств. Каждым из этих средств обработали по 12 растений, после чего подсчитали количество выживших на растениях насекомых. Данные этого эксперимента входят в состав стандартного набора данных R и еще не раз встретятся нам. Они доступны по команде **data**(InsectSprays). В таблице InsectSprays имеются два столбца: count, содержащий результаты подсчета насекомых, и spray, содержащий коды инсектицидных средств (от A до F):

Рисунок 20

```
data(InsectSprays)
head(InsectSprays)
   count spray
1     10     A
2      7     A
3     20     A
4     14     A
5     14     A
6     12     A
```

Изобразим кривые ядерной плотности для каждого из изученных препаратов (рис. 21). Для этого воспользуемся функцией **sm.density.compare**() из пакета sm («*smoothing methods*»), посвященного методам сглаживания и оценкам плотности.

```
attach(InsectSprays)
# Сравнение всех препаратов по кривым ядерной плотности
library(sm)
sm.density.compare(count, spray, lwd = 2, xlab = "Число насекомых")
title(main = "Кривые ядерной плотности")
# Составляем вектор с кодами использованных цветов
Colfill <- c(2:(2 + length(levels(spray))))
# добавляем легенду туда, куда мы кликнем мышью
legend(locator(1), levels(spray), fill = Colfill)
```

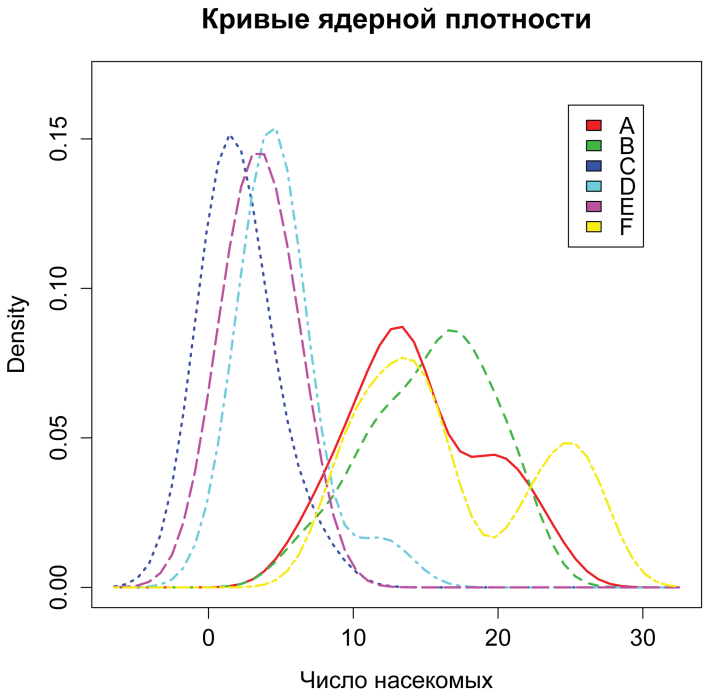


Рисунок 21

Обратите внимание на использование функции **title**(), добавляющей к графику заголовок, а также опции **locator**(1), позволяющей пользователю выбрать место размещения легенды.

Наконец, для оценки взаимодействия между двумя переменными представляет интерес построение графика поверхности ядерной плотности распределения двумерной случайной величины $z = f(x, y)$. Это можно сделать с использованием функции **kde2d**() из популярного пакета MASS. В качестве примера создадим диаграмму совместного распределения показателей time и conc в эксперименте по скорости выведения индометацина (см. раздел 3.1):

```
data(Indometh); attach(Indometh)
library(MASS)
f <- kde2d(time, conc)
image(f, xlab="Время выведения", ylab="Концентрация индометацина")
contour(f, add = TRUE)
```

Здесь функция **image**() создает окрашенную прямоугольную сетку, цвет которой зависит от значения переменной f (в нашем случае это плотность вероятности), а функция **contour**() добавляет на график изолинии (рис. 22).

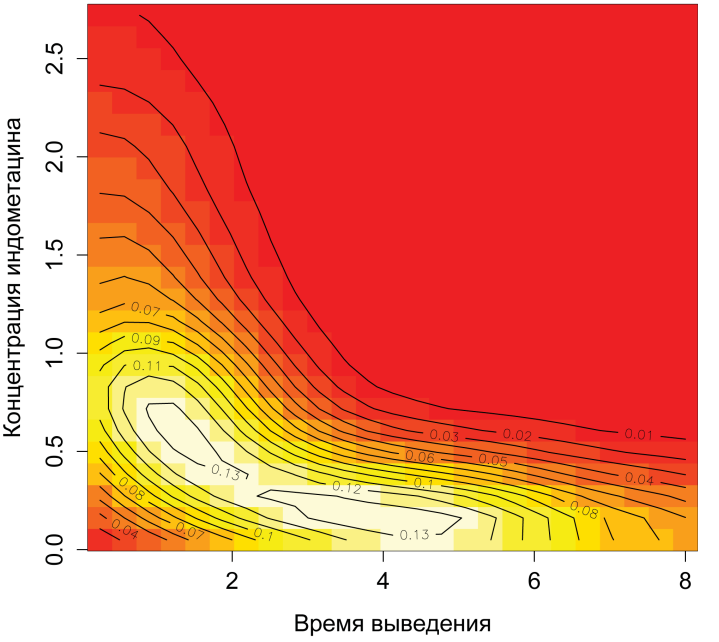


Рисунок 22

Для визуализации связи между двумя переменными, одна из которых является количественной, а другая качественной (фактором), можно использовать диаграммы размахов («*box plots*»). При таком подходе ось ординат обычно соответствует уровням количественной переменной. Но что, если зависимой является качественная переменная? Полезной здесь может оказаться еще одна базовая графическая функция R – **cdplot**(), позволяющая совмещать на одном графике плотности вероятности для каждого уровня интересующей исследователя качественной переменной («*conditional density plot*»).

Приведем пример, заимствованный из книги Everitt & Hothorn (2010) (одного из лучших, на наш взгляд, руководств по биостатистическому анализу с использованием R). Нам потребуется пакет HSAUR2, сопровождающий указанную книгу. Его легко установить при помощи команды **install.packages**("HSAUR2").

В состав пакета HSAUR2 входит набор данных plasma:

```
library(HSAUR2)
data(plasma)
summary(plasma)
   fibrinogen       globulin           ESR
 Min.   :2.090   Min.   :28.00   ESR < 20:26
 1st Qu.:2.290   1st Qu.:31.75   ESR > 20: 6
```

```
Median :2.600   Median :36.00
Mean   :2.789   Mean   :35.66
3rd Qu.:3.167   3rd Qu.:38.00
Max.   :5.060   Max.   :46.00
```

Видно, что таблица plasma содержит две количественные переменные – fibrinogen и globulin и одну качественную переменную с двумя уровнями – ESR.

Речь здесь идет о клинических данных. ESR расшифровывается как «*erythrocyte sedimentation rate*», то есть скорость оседания эритроцитов (СОЭ, мм/ч). Многие заболевания сопровождаются повышенными значениями СОЭ в связи с увеличением концентрации определенных белков в плазме крови, и благодаря этому СОЭ потенциально может служить критерием для диагностики таких заболеваний. Конкретное значение СОЭ у того или иного пациента не очень важно, но известно, что у здоровых людей этот показатель обычно не превышает 20 мм/ч. Поэтому в рассматриваемом исследовании СОЭ регистрировали как качественную переменную с двумя уровнями – "ESR < 20" и "ESR > 20". Вопрос заключался в том, есть ли связь между СОЭ и концентрацией двух конкретных белков в плазме крови пациентов – фибриногена и глобулина? При отсутствии такой связи СОЭ не может служить надежным диагностическим критерием.

Метод, при помощи которого был получен ответ на приведенный выше вопрос (логистическая регрессия), мы опишем в разделе 8.2 при рассмотрении статистических моделей. Здесь же сосредоточимся на разведочном анализе данных: с помощью функции **cdplot**() изобразим графически зависимость вероятности наблюдения "ESR < 20" и "ESR > 20" при разных концентрациях фибриногена и глобулина. Графики для обоих белков будут изображены на одном рисунке, и поэтому для начала разобьем область графического устройства R на две части при помощи функции **layout**() (подробнее см. ?layout):

```
layout(matrix(1:2, ncol = 2))
```

Теперь создадим сами графики (рис. 23):

```
cdplot(ESR ~ fibrinogen, data = plasma)
cdplot(ESR ~ globulin, data = plasma)
```

Из полученных графиков видно, что повышение концентрации обоих белков ведет к увеличению вероятности измерения СОЭ > 20 мм/ч, особенно в случае с фибриногеном.

У функции **cdplot**() есть несколько управляющих аргументов, большинство из которых являются стандартными для графических функций в R. Особенно полезными могут оказаться следующие аргументы (подробно см. справочный файл, доступный по команде ?cdplot):

- при помощи аргумента col можно задать текстовый вектор с названиями цветов, которые будут использованы для заливки полигонов на графике:

```
cdplot(ESR ~ fibrinogen, col = c("coral", "skyblue"), data = plasma)
cdplot(ESR ~ globulin, col = c("coral", "skyblue"), data = plasma)
```

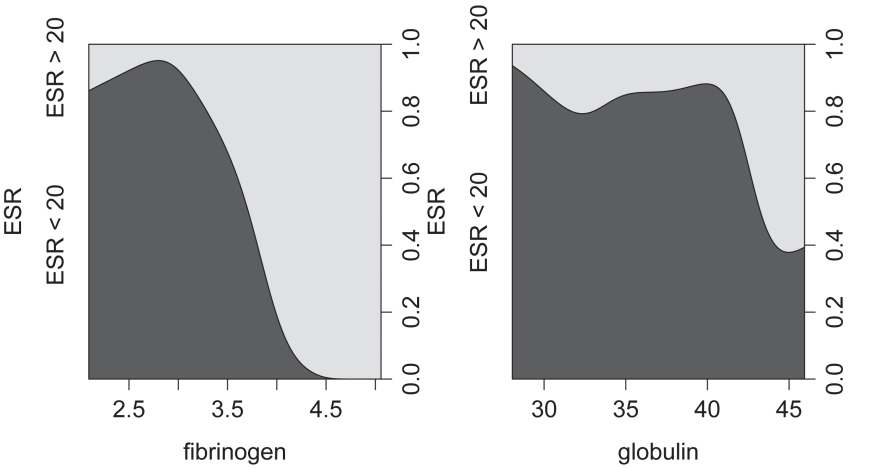


Рисунок 23

- аргумент yaxlabels позволяет «на лету» изменить метки анализируемой зависимой переменной:

```
cdplot(ESR ~ fibrinogen, col = c("coral", "skyblue"),
      yaxlabels = c("< 20 mm/h", "> 20 mm/h"), data = plasma)
cdplot(ESR ~ globulin, col = c("coral", "skyblue"),
      yaxlabels = c("< 20 mm/h", "> 20 mm/h"), data = plasma)
```

- при расчете кривых плотности вероятности функция **cdplot**() обращается к описанной ранее функции **density**(), и поэтому непосредственно при вызове **cdplot**() можно изменять такой параметр **density**(), как уровень сглаживания кривой (аргумент bw) (рис. 24):

```
cdplot(ESR ~ fibrinogen, col = c("coral", "skyblue"),
    yaxlabels = c("< 20 mm/h", "> 20 mm/h"), bw = 0.9, data = plasma)
cdplot(ESR ~ globulin, col = c("coral", "skyblue"),
     yaxlabels = c("< 20 mm/h", "> 20 mm/h"), bw = 0.9, data = plasma)
```

3.3. Диаграммы размахов

Диаграммы размахов, или «ящики с усами» (англ. «*box plots*», «*box-whisker plots*»), получили свое название за характерный вид: точку или линию, соответствующую некоторой мере центральной тенденции в данных, окружает прямоугольник («ящик»), длина которого соответствует определенному показателю разброса. Дополнительно от этого прямоугольника отходят «усы», также отражающие разброс в данных или, в некоторых случаях, точность оценки меры центральной тенденции. Графики этого типа очень популярны, поскольку позволяют дать очень полную статистическую характеристику анализируемой совокупности. Кроме того, диаграммы размахов можно использовать для визуальной экспресс-оценки раз-

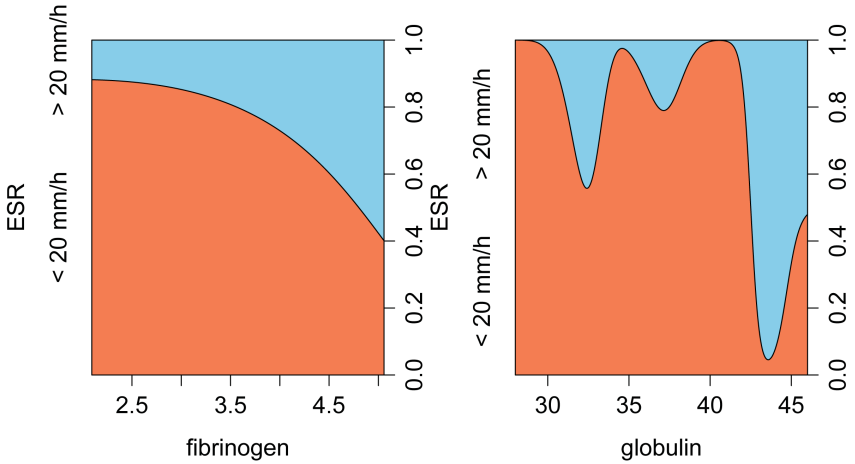


Рисунок 24

ницы между двумя и более группами (например, между датами отбора проб, экспериментальными группами, участками пространства и т. п.).

В R для построения диаграмм размахов служит функция **boxplot**(). В отличие от многих других статистических программ, в R при построении диаграмм размахов используются устойчивые (робастные) оценки центральной тенденции (медиана) и разброса (интерквартильный размах – ИКР). Верхний «ус» простирается от верхней границы «ящика» до наибольшего выборочного значения, находящегося в пределах расстояния 1.5×ИКР от этой границы. Аналогично нижний «ус» простирается от нижней границы «ящика» до наименьшего выборочного значения, находящегося в пределах расстояния 1.5×ИКР от этой границы. Длину данного интервала (то есть 1.5×ИКР) можно изменить при помощи аргумента `range` функции **boxplot**(). Строение получаемых при помощи этой функции «ящиков с усами» представлено на рис. 25:

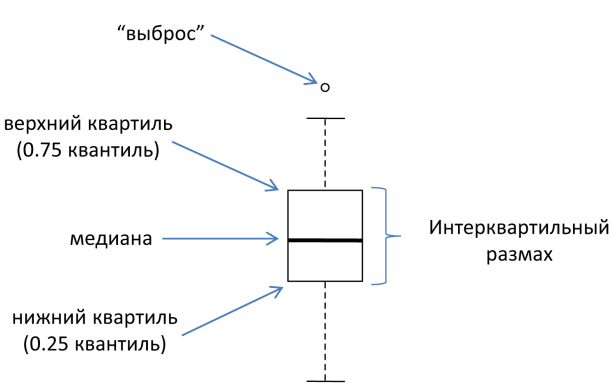


Рисунок 25

Наблюдения, находящиеся за пределами «усов», потенциально могут быть выбросами. Однако всегда следует внимательно относиться к такого рода нестандартным наблюдениям – они вполне могут оказаться «нормальными» для исследуемой совокупности и поэтому не должны удаляться из анализа без дополнительного расследования причин их появления.

Особенности использования функции **boxplot**() рассмотрим на примере данных InsectSprays, описывающих эксперимент по изучению эффективности шести видов инсектицидных средств (см. раздел 3.3). Для построения диаграммы размахов по этим данным достаточно выполнить команду

```
boxplot(count ~ spray, data = InsectSprays)
```

Как всегда, мы можем поработать над автоматически построенным графиком и несколько улучшить его внешний вид (рис. 26). Например, можно добавить заголовки осей и самого рисунка (аргументы xlab, ylab и main), а также залить «ящики» каким-нибудь цветом (аргумент col):

```
boxplot(count ~ spray,
        xlab = "Инсектициды",
        ylab = "Количество выживших насекомых",
        main = "Эффективность инсектицидов",
        col = "coral", data = InsectSprays)
```

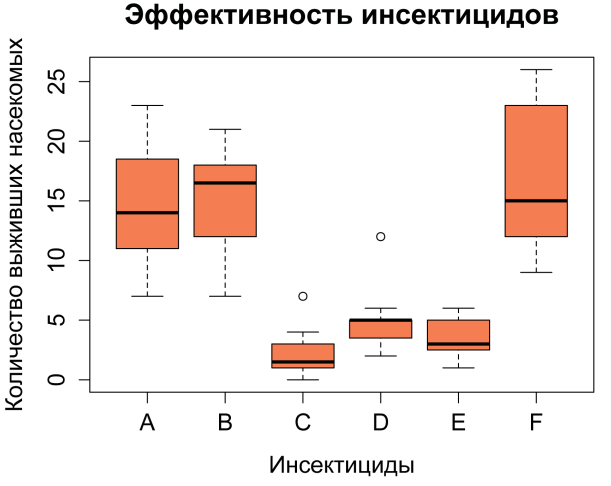


Рисунок 26

Как видим, количество насекомых на растениях, обработанных инсектицидами C, D и E, было наиболее низким, что говорит о высокой эффективности этих препаратов, по сравнению с тремя другими средствами. На растениях, обработанных средствами C и D, были отмечены необычно высокие количества насекомых (см. точки над «усами»). Однако для насекомых характерно пятнистое пространствен-

ное распределение, и поэтому вряд ли эти высокие наблюдения являются истинными выбросами.

Обратите внимание на то, как были указаны переменные для построения графика – в виде так называемой «*формулы*»: `count ~ spray`. Это стандартный способ, используемый в R для формулировки статистических моделей. По левую сторону от знака ~ (он называется «тильда») указывается зависимая переменная, по правую – предикторы.

Подобно **`plot`**`()`, функция **`boxplot`**`()` обладает большим числом управляющих аргументов. Например, используя аргумент `log`, можно изобразить данные на логарифмической шкале. Аргумент `varwidth` (от «*variable*» – переменная и «*width*» – ширина) позволяет сделать так, что ширина «ящиков» будет пропорциональна квадратному корню из числа наблюдений в каждой группе (для этого необходимо использовать `varwidth = TRUE`). Это может оказаться полезной оцпией для визуализации выборок, заметно различающихся по размеру (в нашем примере смысла в `varwidth = TRUE` нет, поскольку в каждой группе имеется по 12 наблюдений). Аргумент `horizontal` со значением `TRUE` позволяет изобразить «ящики» горизонтально (см. пример ниже). Подробнее об аргументах **`boxplot`**`()` можно узнать из справочного файла по этой функции (доступен по команде `?boxplot`).

```
boxplot(count ~ spray,
        ylab = "Инсектициды",
        xlab = "Количество выживших насекомых",
        main = "Эффективность инсектицидов",
        col = "coral", horizontal = TRUE,
        data = InsectSprays)
```

Интересно, что построить диаграммы размаха можно не только при помощи специализированной функции **`boxplot`**`()`, но также и функции **`plot`**`()` – например, по команде **`plot`**`(count ~ spray, data = InsectSprays)`. Дело в том, что **`plot`**`()` – очень «смышленая» функция: она автоматически распознает, что переменная `count` является количественной, а `spray` – номинальной, и поэтому начинает вести себя как **`boxplot`**`()`.

По аналогии с ядерной функцией, описывающей плотность распределения вероятности двумерной случайной величины, при наличии двух измерений «ящик с усами» превращается в «мешок в мешке» (*«bag plots»*). Как и в одномерном случае, внешний мешок ограничивается экстремальными выборочными значениями, а внутренний мешок содержит 50% всех наблюдений. За пределами внешнего мешка могут находиться выбросы. В центре диаграммы размещается область аппроксимации двумерной медианы. В качестве примера покажем диаграмму совместного распределения показателей `time` и `conc` в эксперименте со скоростью выведения индометацина (см. раздел 3.1), полученную с использованием функции **`bagplot`**`()` из пакета `aplpack` (рис. 27):

```
data(Indometh); attach(Indometh)
library(aplpack)
bagplot(time, conc, xlab = "Время выведения",
        ylab = "Концентрация индометацина", main = "Мешок с усами")
```

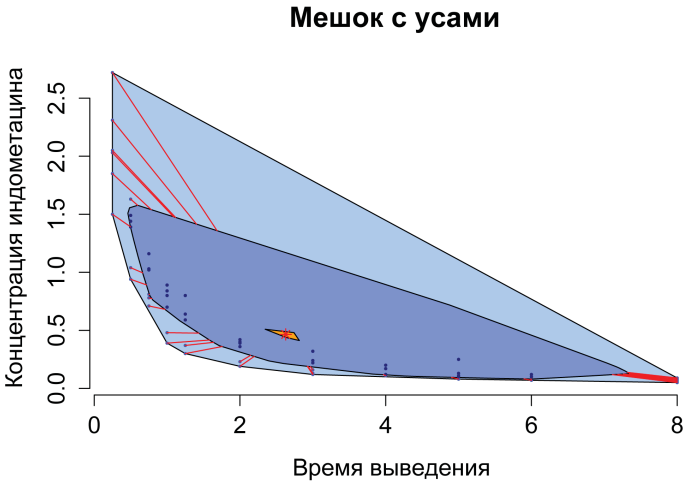


Рисунок 27

3.4. Круговые и столбиковые диаграммы

Круговые диаграммы (англ. «*pie charts*»), мягко говоря, не в почете у профессиональных статистиков. Информация, представляемая при помощи круговой диаграммы, плохо воспринимается визуально, и практически всегда лучшей альтернативой этому способу визуализации данных будет рассмотренная ниже точечная диаграмма Кливленда или, в крайнем случае, столбиковая диаграмма. Неудивительно поэтому, что в первых версиях R даже не было отдельной функции для построения круговых диаграмм. Позднее такая функция появилась, поскольку в ряде случаев этот вид диаграмм все же может оказаться полезным. Несложно догадаться, что соответствующая функция называется **`pie`**`()`.

Функция **`pie`**`()` имеет несколько аргументов (подробнее см. `?pie`). Основными из них являются следующие:

- `x` – вектор из положительных чисел, на основе которых строится диаграмма;
- `labels` – текстовый вектор, содержащий подписи секторов диаграммы; если значения `x` уже имеют атрибут `names` (имена), то аргумент `labels` указывать не обязательно (см. ниже);
- `radius` – изменяет размер квадрата, внутри которого строится диаграмма; в случаях, когда подписи секторов диаграммы слишком длинные, размер этого квадрата можно уменьшить (возможные значения: от –1 до 1; см. ниже);
- `init.angle` – угол поворота диаграммы;
- `col` – вектор (числовой или текстовый), содержащий коды цветов для заливки секторов диаграммы;
- `main` – текстовый вектор, содержащий заголовок диаграммы;

- ... – другие графические параметры (например, параметры, определяющие размер подписей секторов диаграммы, цвет линий и т. п.).

В качестве примера рассмотрим результаты голосования в Госдуму Российской Федерации (http://bit.ly/1wWb8fu). Изобразим на одном рисунке две круговые диаграммы, отображающие явку избирателей и распределение их голосов (рис. 28). Для начала создадим два числовых вектора с данными:

```
# Данные по явке избирателей:
percent.voted <- c(60, 40)
# Распределение голосов:
votes <- c(49.3, 19.2, 13.2, 11.7, 3.4, 1.0, 0.6)
```

Рисунок 28

Каждому элементу созданных векторов присвоим соответствующие имена при помощи функции **names**(). Эти имена потом будут автоматически использованы программой в качестве подписей секторов диаграммы:

```
names(percent.voted) <- c("Проголосовали", "Не явились")
names(votes) <- c("Единая Россия", "КПРФ",
     "Справедливая Россия", "ЛДПР", "Яблоко",
     "Патриоты России", "Правое дело")
```

Посмотрим, что получилось:

```
percent.voted
Проголосовали    Не явились
           60            40

votes
  Единая Россия    КПРФ    Справедливая Россия        ЛДПР
           49.3    19.2                   13.2        11.7
         Яблоко     Патриоты России         Правое дело
            3.4                 1.0                 0.6
```

Поскольку стоит задача изобразить обе диаграммы на одном рисунке, графическое окно R необходимо разбить на две части. Для этого можно использовать команду **par**() (от «*parameters*» – параметры) и один из ее многочисленных аргументов – mfrow:

```
par(mfrow = c(1,2))
```

Данная команда приведет к разбиению графического окна R на 2 ячейки, формирующие одну строку. Круговые диаграммы будут последовательно добавлены программой в каждую из этих ячеек:

```
pie(percent.voted, radius = 0.9, cex = 0.6, main = "Явка")
pie(votes, cex = 0.6, radius = 0.9, init.angle = -10,
    main = "Распределение голосов")
```

Во второй из приведенных команд можно слегка повернуть диаграмму влево при помощи аргумента init.angle, что позволит избежать перекрывания подписей «Правое дело» и «Патриоты России». Цвета заливки секторов диаграммы, используемые программой по умолчанию, можно изменить при помощи аргумента col. Например:

```
pie(percent.voted, radius = 0.9, cex = 0.6, main = "Явка",
    col = c("black", "gray80"))
pie(votes, cex = 0.6, radius = 0.9, init.angle = -10,
    main = "Распределение голосов", col = c(2:8))
```

Для создания *столбиковых диаграмм* (также «*столбчатых*», реже «*линейчатых*»; англ. «*bar plots*» или «*bar charts*») в системе R служит функция **barplot**(). У этой функции имеется большое количество аргументов (подробнее см. ?barplot), к основным из которых относятся:

- hight (высота) – числовой вектор или матрица со значениями, используемыми для построения диаграммы. Если аргумент hight указан в виде вектора, то строится график из последовательно расположенных столбцов, высоты которых соответствуют значениям из этого вектора. Если hight указан в виде матрицы и аргумент beside = FALSE, то будет построена столбчатая *диаграмма с накоплением*. Если же hight указан в виде матрицы и аргумент beside = TRUE, то столбцы диаграммы будут *сгруппированы* в соответствии со столбцами матрицы;
- width (ширина) – необязательный параметр, позволяющий регулировать ширину столбцов на диаграмме. Указывается в виде числового вектора, значения которого соответствуют ширине столбцов;
- space (пространство) – величина зазора между столбцами (пропорционально их средней ширине). Может быть указана либо в виде одного числа, либо в виде вектора из чисел, соответствующих каждому столбцу диаграммы;
- names.arg – текстовый вектор, содержащий подписи (вдоль оси X) для каждого столбца или группы столбцов. Если этот аргумент не указан, то в качестве подписей автоматически будут использованы имена элементов век-

тора height (если таковые имеются) либо заголовки столбцов, если height представляет собой матрицу;

- legend.text – вектор, содержащий текстовые элементы легенды графика. Этот аргумент полезен, только если height является матрицей. В этом случае метки легенды должны соответствовать строкам матрицы. Аргументу legend.text можно также присвоить значение TRUE, и тогда имена строк матрицы (если таковые имеются) будут использованы в качестве меток легенды автоматически;
- beside – принимает логическое значение и имеет смысл, только если height является матрицей. Значение FALSE приведет к построению диаграммы с накоплением. При значении TRUE столбцы будут сгруппированы;
- horiz – принимает логическое значение: TRUE для горизонтального расположения столбцов и FALSE – для вертикального;
- density – числовой вектор, задающий плотность заштриховки столбцов;
- angle – угол наклона штрихов (в градусах);
- col – вектор цветовых кодов для столбцов или их элементов. По умолчанию столбцы закрашиваются серым цветом, если height – вектор, и разными градациями серого, если height – матрица;
- border – код цвета для обводки столбцов. Если границу столбцов обводить не предполагается, можно указать border = NA;
- ... – другие графические параметры (см., например, ?plot и ?par).

В качестве первого примера используем рассмотренные ранее данные по эффективности шести инсектицидных средств (A–F), входящие в базовую комплектацию R. Загрузим таблицу с этими данными в рабочую среду R:

```
data(InsectSprays)
InsectSprays
```

Стоит задача отобразить в виде столбиковой диаграммы средние значения числа насекомых, учтенных на экспериментальных растениях после обработки каждым инсектицидом. Эти средние значения можно быстро рассчитать для каждой группы при помощи функции **tapply**(). Результат вычислений сохраним в векторе Means:

```
attach(InsectSprays)
Means <- tapply(count, spray, mean)

Means
    A     B     C     D     E     F
14.50 15.33  2.08  4.92  3.50 16.67
```

Для построения столбчатой диаграммы в ее простейшем виде достаточно выполнить следующую команду:

```
barplot(height = Means) # или просто barplot(Means)
```

Добавим подписи осей и закрасим столбцы диаграммы голубым цветом:

```
barplot(Means, col = "steelblue",
        xlab = "Инсектицид",
        ylab = "Количество выживших насекомых")
```

Теперь продемонстрируем действие разных аргументов функции **barplot**(), изменяющих внешний вид диаграммы:

- изменены ширина столбцов (теперь она пропорциональна квадратному корню из высоты соответствующего столбца – аргумент width) и цвет их обводки (на красный – аргумент border) (рис. 29):

```
barplot(Means, col = "steelblue",
        xlab = "Инсектицид",
        ylab = "Количество выживших насекомых",
        border = "red", width = sqrt(Means))
```

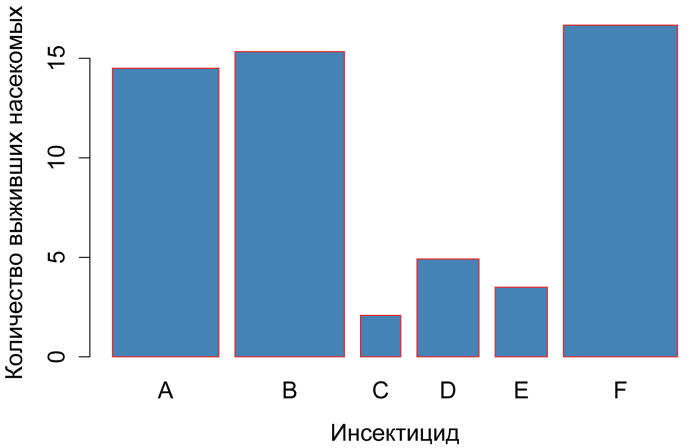


Рисунок 29

- столбцы имеют исходную ширину и цвет обводки, но теперь они заштрихованы (аргумент density) и изображены горизонтально (аргумент horiz); аргумент las = 1 заставляет названия инсектицидов принять вертикальное положение (рис. 30):

```
barplot(Means, density = 20, col = "red", horiz = T, las = 1,
        ylab = "Инсектицид",
        xlab = "Количество выживших насекомых")
```

- предыдущий график изменен путем увеличения зазора между столбцами (аргумент space) и изменения угла наклона штрихов (аргумент angle):

```
barplot(Means, density = 20, angle = -45, space = 2,
        col = "red", horiz = TRUE, las = 1,
        ylab = "Инсектицид",
        xlab = "Количество выживших насекомых")
```

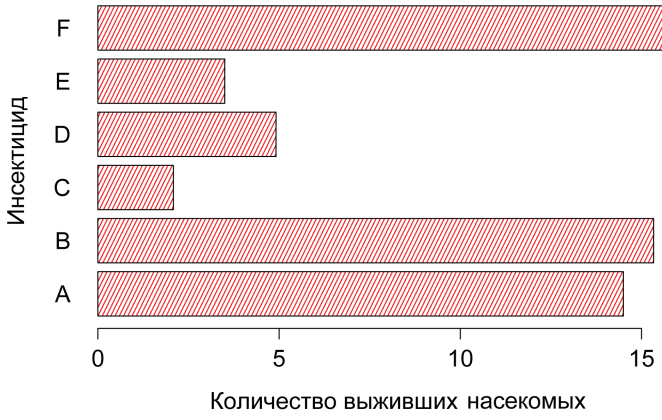


Рисунок 30

Построение диаграммы, где несколько столбцов содержат группировку в соответствии с уровнями какого-либо фактора, рассмотрим на примере данных из таблицы genotype, входящей в состав пакета MASS. В эту таблицу входят данные из эксперимента, в котором были задействованы лабораторные крысы четырех разных генотипов (A, B, C, D). Выводок (Litter), полученный самками (Mother) каждого генотипа, изолировался от матерей и отдавался на вскармливание самкам других генотипов. В конце эксперимента (на 28-й день) был измерен вес у крысят (Wt), кормившихся молоком от разных самок, с целью установить влияние генотипа на этот показатель:

```
library(MASS)
data(genotype)
head(genotype)
```

```
  Litter Mother   Wt
1      A      A 61.5
2      A      A 68.2
3      A      A 64.0
4      A      A 65.0
5      A      A 59.7
6      A      B 55.0
```

Рассчитаем средние значения веса для каждой экспериментальной группы при помощи функции **tapply**():

```
means = with(genotype, tapply(Wt, list(Litter, Mother), mean))
# Функция with() позволяет указать таблицу, из которой
# функция tapply() должна брать данные для вычислений

# Просмотрим содержимое матрицы means:
```

```
means
      A     B     I     J
A 63.68 52.40 54.12 48.96
B 52.33 60.64 53.92 45.90
I 47.10 64.37 51.60 49.43
J 54.35 56.10 54.53 49.06
```

Матрицу means можно уже непосредственно использовать для построения столбчатой диаграммы (рис. 31):

```
barplot(means, beside = TRUE,
        col = topo.colors(4),
        legend.text = rownames(means),
        xlab = "Выводок", ylab = "Вес, г",
        ylim = c(0, 100))
```

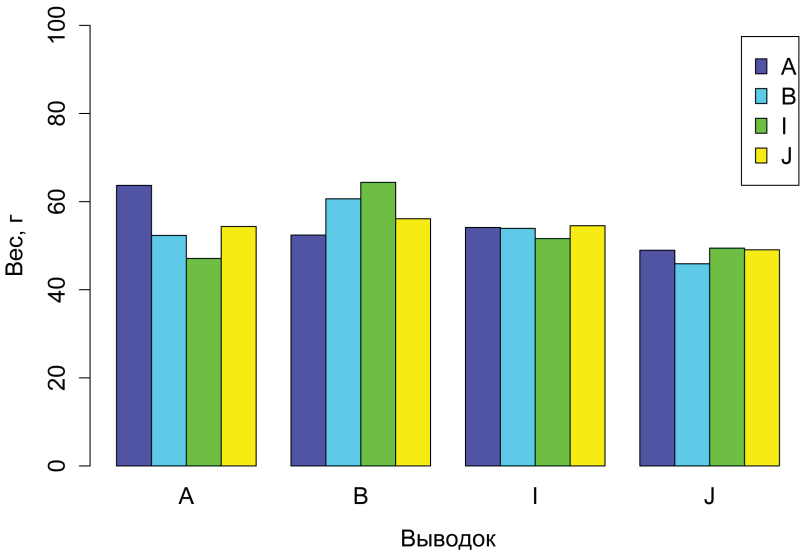


Рисунок 31

В приведенном коде цвет столбцов задан при помощи функции **topo.colors**(), которая автоматически подбирает необходимое количество хорошо сочетающихся цветов в стиле «топографическая карта». Легенда графика включена при помощи агрумента legend.text, которому присвоены значения в виде заголовков строк матрицы means. Для того чтобы легенда не «наползла» на столбцы, пришлось несколько растянуть ось Y при помощи аргумента ylim.

Обратите внимание на то, что для построения приведенного на рис. 31 графика аргументу beside было присвоено значение TRUE. Если бы мы присвоили этому аргументу значение FALSE, то получили бы столбиковую *диаграмму с накоплением*:

```
barplot(means, beside = FALSE,
        col = topo.colors(4),
        xlab = "Выводок", ylab = "Вес, г")
```

К полученным столбцам диаграмм мы можем добавить «усы», соответствующие, например, стандартным отклонениям (рис. 32). Стандартные отклонения легко рассчитать, используя все ту же функцию **tapply**() (здесь – в связке с **with**()):

```
sds = with(genotype, tapply(Wt, list(Litter, Mother), sd))
```

```
sds
      A    B    I    J
A  3.27 9.37 5.32 8.76
B  5.53 5.65 5.11 7.64
I 18.10 7.12 8.62 5.37
J  5.33 3.35 8.38 5.34
```

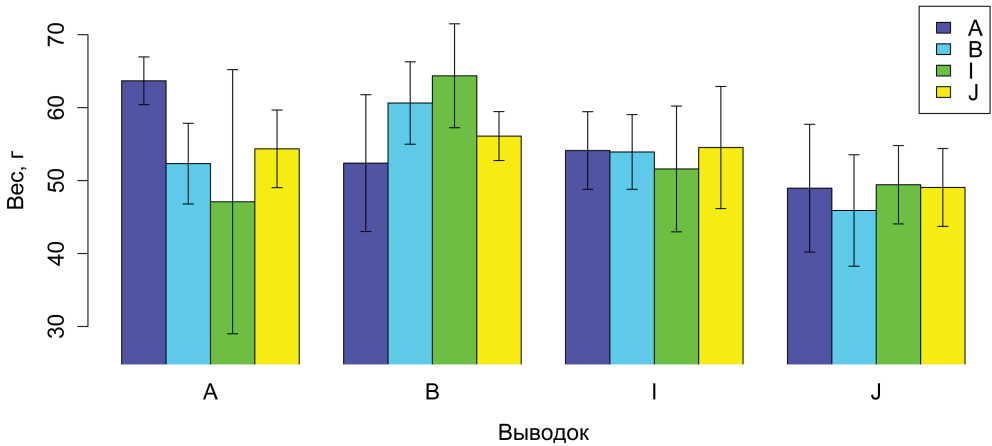


Рисунок 32

Построим и сохраним диаграмму в виде самостоятельного объекта b. Помимо прочего, этот объект будет хранить информацию о высоте столбцов, что потребуется далее при добавлении «усов» к этим столбцам при помощи функции **arrows**(). Аргумент ylim в связке с функциями **min**(), **max**() и **pretty**() используется для автоматического подбора оптимального диапазона значений, отображаемых на оси ординат (подробнее см. ?min, ?max, ?pretty):

```
b <- barplot(means, ylim = c(min(pretty(means-sds)),
             max(pretty(means + sds))),
             col = topo.colors(4),
             beside = TRUE, xpd = FALSE,
             ylab = "Вес, г", xlab = "Выводок",
             legend.text = rownames(means))
# Добавляем "усы" (подробнее см. ?arrows):
arrows(b, means + sds, b, means - sds, angle = 90, code = 3, length = 0.05)
```

3.5. Диаграммы Кливленда и одномерные диаграммы рассеяния

Точечные диаграммы Кливленда представляют собой графики, на которых точки используются для отображения значений некоторой количественной переменной (или переменных), разбитой(ых) на группы в соответствии с уровнями некоторой номинальной переменной (или переменных). Этот инструмент графического анализа данных получил свое название в честь предложившего его Уильяма Кливленда (William Cleveland). В работе Cleveland & McGill (1984, http://bit.ly/1Kt7oh2) было показано, что столбиковые диаграммы, используемые для изображения сгруппированных значений количественных переменных, визуально плохо воспринимаются людьми (*впрочем, это еще и дело вкуса...*). В качестве альтернативы были предложены точечные диаграммы.

Для пояснения этой идеи приведем ниже столбиковую диаграмму, изображающую распределение 32 моделей автомобилей 1973–1974 годов выпуска по экономичности двигателя (выражается в количестве миль, которое автомобиль проезжает на одном галлоне топлива). Данные, использованные для построения диаграммы, были опубликованы в американском журнале *Motor Trend* в 1974 г. и входят в стандартный набор данных R (доступны по команде **data**(mtcars)).

Для начала загрузим соответствующую таблицу с данными (mtcars) в рабочую среду R и исследуем ее содержимое:

```
data(mtcars)
mtcars
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
...
```

В таблице имеются данные по 11 параметрам, характеризующим каждую модель. Подробнее о том, какие параметры были учтены, можно ознакомиться в файле помощи, доступном по команде ?mtcars.

Для начала построим столбиковую диаграмму, причем так, чтобы столбики на ней были залиты разными цветами в соответствии с количеством цилиндров cyl в двигателях автомобилей (рис. 33):

```
par(mar = c(6, 4, 1, 1))

x <- mtcars[order(mtcars$mpg), ]
```

```
x$color[x$cyl==4] <- 1
x$color[x$cyl==6] <- 2
x$color[x$cyl==8] <- 3

with(x, barplot(mpg, names.arg = rownames(x), las = 2,
     cex.axis = 0.6, cex.lab = 0.8, cex = 0.6,
     ylab = "Миль/галлон", col = color))

legend(1, 30, legend = c(8, 6, 4), fill = c(3, 2, 1), cex = 0.6)
```

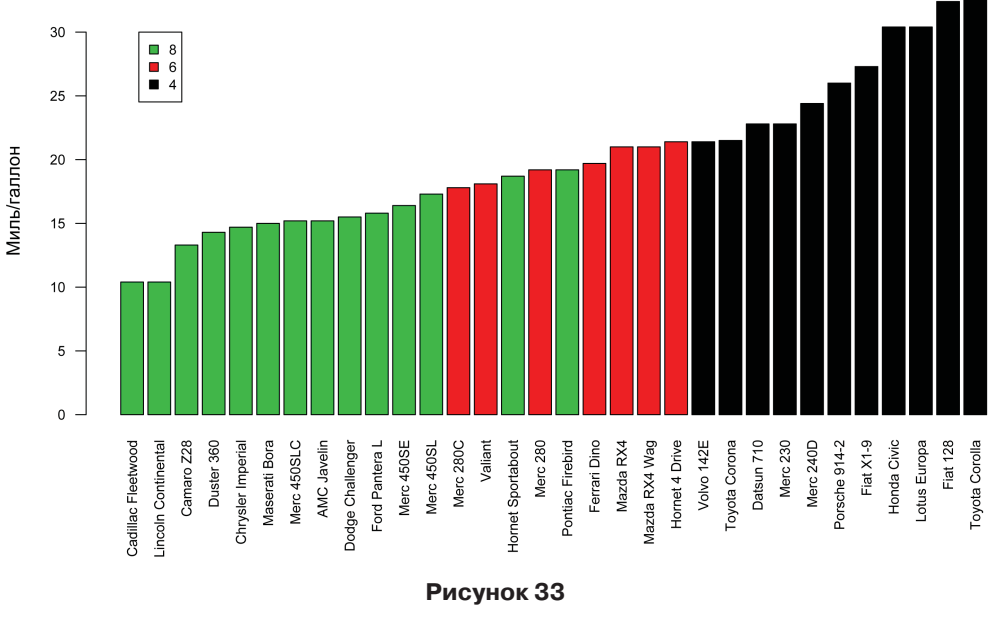


Рисунок 33

На графике вполне неплохо видно, что автомобили с меньшим количеством цилиндров способны проехать большее расстояние на одном галлоне топлива. Кроме того, можно проследить распределение моделей по экономичности в пределах каждой из групп, выделенных по количеству цилиндров.

Проблема со столбиковыми диаграммами состоит, однако, в том, что они в значительной мере избыточны, поскольку площади, ограниченные столбиками, не несут никакой смысловой нагрузки (то есть, выбрав более узкие или более широкие столбики, мы бы все равно пришли к тем же заключениям о характере распределения автомобилей). Согласно У. Кливленду, более подходящим типом статистических графиков для такой ситуации была бы точечная диаграмма. В системе R построение точечных диаграмм осуществляется с помощью функции **dotchart**(). Рассмотрим ее использование на том же примере с автомобилями.

Для нас сейчас интересен столбец mgp (от «*miles per gallon*» – миль на галлон), в котором содержатся данные по пробегу каждой модели в расчете на галлон топлива. Точечную диаграмму по этим данным можно быстро построить следующим образом:

```
dotchart(mtcars$mpg, labels = row.names(mtcars),
         main = "Экономия топлива у 32 моделей автомобилей",
         xlab = "Миль/галлон", cex = 0.8)
```

В приведенном коде указаны следующие параметры функции **dotchart**(): *а*) переменная, для которой строится график (mtcars$mpg); *б*) текстовый вектор, содержащий названия моделей автомобилей (в данном случае они являются названиями строк таблицы – **row.names**(mtcars)); *в*) заголовок графика (аргумент main) и название оси X (аргумент xlab), и, наконец, *г*) размер точек на графике и одновременно размер шрифта для названий моделей (cex = 0.8).

Такой график пока еще достаточно сырой, хотя и позволяет исследовать разброс имеющихся значений mpg у разных моделей. Картина станет гораздо более ясной, если мы отсортируем данные по возрастанию пробега, сгруппируем данные по количеству цилиндров в двигателе и раскрасим соответствующие группы разными цветами.

Сначала отсортируем исходную таблицу по возрастанию mpg с использованием функции **order**() и сохраним результат в виде новой таблицы данных с именем x:

```
x <- mtcars[order(mtcars$mpg), ]
```

Преобразуем количественную переменную cyl (от «*cylinder*» – цилинр) в новой таблице x в фактор – это нужно сделать, поскольку мы собираемся сгруппировать значения mpg именно по количеству цилиндров:

```
x$cyl <- factor(x$cyl)
```

Создадим новый столбец color в таблице x, который будет содержать числовые коды цветов для каждой из трех групп автомобилей:

```
x$color[x$cyl==4] <- 1
x$color[x$cyl==6] <- 2
x$color[x$cyl==8] <- 3
```

Теперь у нас есть все необходимое для построения желаемого графика (рис. 34):

```
dotchart(x$mpg, labels = row.names(x),
   groups = x$cyl, gcolor = "blue", pch = 16,
   main = "Экономичность двигателя у 32 моделей автомобилей",
   xlab = "Миль/галлон", cex = 0.8, color = x$color)
```

В приведенном коде аргумент groups используется для указания группирующей переменной (в нашем случае – преобразованная в фактор переменная cyl). При помощи аргумента gcolor задается цвет названий групп (здесь – голубой). Аргумент color служит для указания цветов, специфичных для каждой группы (в нашем примере – 1 (черный) для машин с четырьмя цилиндрами, 2 (красный) для

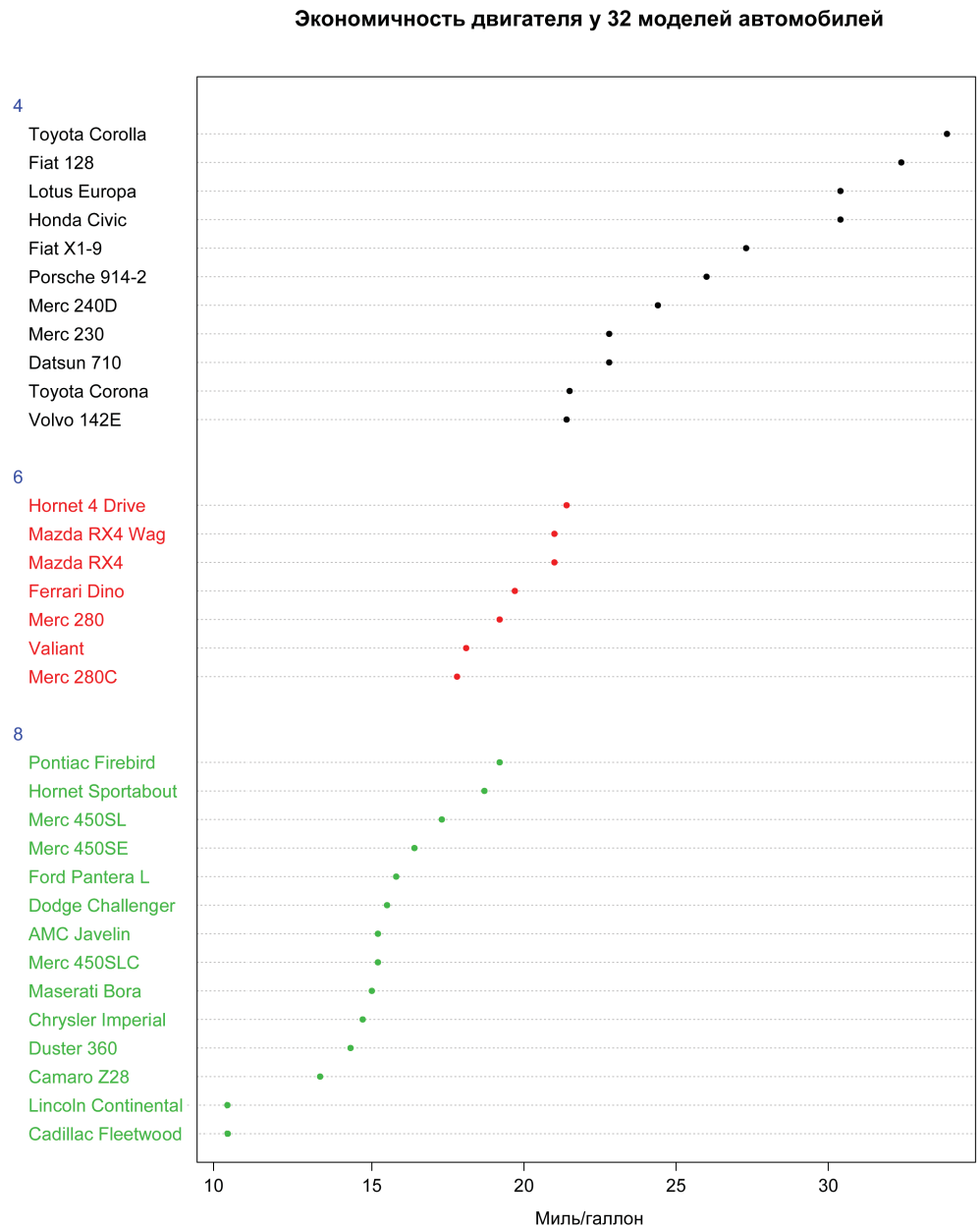

Экономичность двигателя у 32 моделей автомобилей
4
Toyota Corolla
Fiat 128
Lotus Europa
Honda Civic
Fiat X1-9
Porsche 914-2
Merc 240D
Merc 230
Datsun 710
Toyota Corona
Volvo 142E
6
Hornet 4 Drive
Mazda RX4 Wag
Mazda RX4
Ferrari Dino
Merc 280
Valiant
Merc 280C
8
Pontiac Firebird
Hornet Sportabout
Merc 450SL
Merc 450SE
Ford Pantera L
Dodge Challenger
AMC Javelin
Merc 450SLC
Maserati Bora
Chrysler Imperial
Duster 360
Camaro Z28
Lincoln Continental
Cadillac Fleetwood
10
15
20
25
30
Миль/галлон

Рисунок 34

машин с шестью цилиндрами и 3 (зеленый) для машин с восемью двигателями; все эти числовые коды цветов хранятся в столбце `color` таблицы `x`).

Благодаря сортировке и группировке данных, а также использованию точек разного цвета вместо столбиков полученная точечная диаграмма воспринимается гораздо легче, чем приведенная выше «тяжелая» столбиковая диаграмма.

Одномерные диаграммы рассеяния (*1-D scatter plots*) представляют собой один из вариантов графического представления распределений количественных переменных. Точки, соответствующие значениям исследуемой переменной, изображаются на таких графиках вдоль единственной числовой оси. При необходимости визуализировать свойства небольших выборок одномерные диаграммы рассеяния будут отличной альтернативой диаграммам размахов. В англоязычной литературе одномерные диаграммы рассеяния называют также «*strip charts*» или «*strip plots*», что можно перевести как «ленточные диаграммы». Это название происходит от характера расположения точек на графике – они как бы выстраиваются в «ленты» (см. ниже). Реже такие графики называют еще «*точечными диаграммами Уилкинсона*» (Wilkinson, 1999, http://bit.ly/1B8lwGh).

В системе R для построения одномерных диаграмм рассеяния служит функция **`stripchart`**`()`. В качестве примера используем рассмотренные ранее данные по эффективности шести инсектицидных средств. Загрузим таблицу с данными в рабочую среду R:

```
data(InsectSprays)
names(InsectSprays)
[1] "count" "spray"
```

Для начала используем все имеющиеся значения из столбца `count` (число насекомых, выживших после обработки растений инсектицидами) и посмотрим, как они распределяются:

```
stripchart(InsectSprays$count, method = "stack")
```

В приведенной команде помимо переменной, для которой строится график (`InsectSprays$count`), был использован также аргумент `method` (метод) со значением `"stack"`. Этот аргумент контролирует характер расположения точек на графике: при значении `"stack"` («стопка») наблюдения с одинаковыми значениями укладываются друг над другом, образуя фигуры, напоминающие ленты (отсюда название «*strip chart*» – см. выше).

В таблице `InsectSprays` хранятся данные по эффективности шести видов инсектицидов. Мы можем использовать это обстоятельство и изобразить на одном графике распределения значений `count` для каждого из них. Для этого нужно будет сообщить функции **`stripchart`**`()`, что значения переменной `count` сгруппированы по уровням фактора `spray`:

```
stripchart(InsectSprays$count ~ InsectSprays$spray,
           xlab = "Количество выживших насекомых",
           ylab = "Инсектицид",
           method = "stack")
```

Используя аргумент vertical со значением TRUE, мы можем перевернуть приведенный график так, что по оси X будут отображены названия инсектицидов, а по оси Y – количество выживших насекомых (рис. 35):

```
stripchart(InsectSprays$count ~ InsectSprays$spray,
           ylab = "Количество выживших насекомых",
           xlab = "Инсектицид",
           vertical = TRUE,
           method = "stack")
```

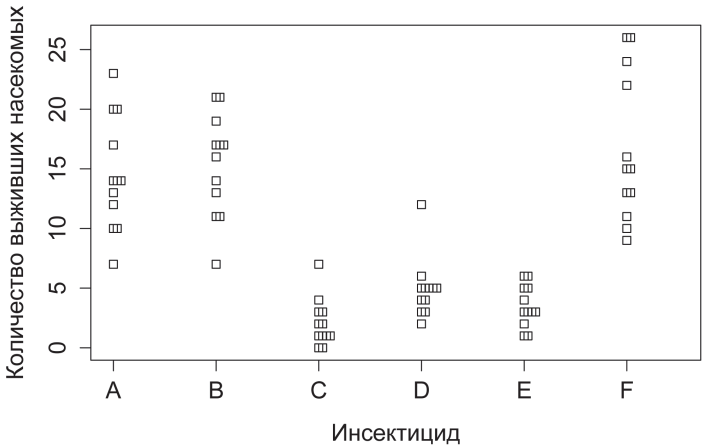


Рисунок 35

Следует отметить, что по умолчанию аргумент method принимает значение "overplot" (от «*over*» – поверх и «*plot*» – рисовать), что приводит к простому наслоению точек с одинаковыми значениями друг на друга:

```
stripchart(InsectSprays$count, method = "overplot")
```

Конечно, такой рисунок имеет мало смысла для рассматриваемых данных, но в некоторых ситуациях метод "overplot" может оказаться полезным.

Иногда очень полезным методом функции **stripchart**() будет "jitter" (дословно переводится как «дрожь»), который позволяет вносить случайный шум в расположение точек на графике. Количество шума задается при помощи одноименного аргумента, например:

```
stripchart(InsectSprays$count ~ InsectSprays$spray,
           ylab = "Количество выживших насекомых",
           xlab = "Инсектицид",
           vertical = TRUE,
           method = "jitter",
           jitter = 0.1,
           pch = 1, col = "blue")
```

Обратите внимание на то, что в приведенном выше коде были указаны также такие параметры, как форма символов (аргумент pch) и их цвет (аргумент col) (подробнее о настройке внешнего вида символов см. раздел 3.1).

Полученный график очень наглядно демонстрирует разброс наблюдений в каждой экспериментальной группе и позволяет сделать определенные предварительные заключения по поводу эффективности исследованных инсектицидов (в частности, похоже, что препараты C, D и E обладают наиболее высокой эффективностью, так как вызывают гибель большего количества насекомых, чем при обработке растений другими препаратами). Однако мы можем пойти еще дальше и изобразить на этом же графике средние значения количества выживших насекомых в каждой группе и их соответствующие стандартные отклонения (рис. 36). Для этого придется выполнить некоторые дополнительные вычисления и применить дополнительные графические функции.

Сначала рассчитаем средние значения для каждой группы и сохраним их в виде вектора с именем means. Это можно сделать при помощи уже известной нам функции **tapply**():

```
means <- tapply(InsectSprays$count, InsectSprays$spray,  FUN = mean)
means
   A    B    C    D    E    F
14.5 15.3  2.1  4.9  3.5 16.7
```

Аналогичным образом рассчитаем стандартные отклонения для каждой группы:

```
(SDs <- tapply(InsectSprays$count, InsectSprays$spray,  FUN = sd))
  A   B   C   D   E   F
4.7 4.3 2.0 2.5 1.7 6.2
```

Теперь добавим рассчитанные средние значения на график в виде небольших горизонтальных линий. Добавить произвольную линию на график позволяет функция **segments**() (сегменты; подробнее см. ?segments). Поскольку у нас имеется шесть групп, функцию **segments**() необходимо будет применить шесть раз. Для ускорения процесса создадим цикл при помощи оператора for:

```
for(i in 0:6){
segments(x0 = 0.8+i, y0 = means[1+i],
   x1 = 1+i, y1 = means[1+i], lwd = 3, lend = "square")
  }
```

Подобно ранее рассмотренному примеру, «усы» к линиям можно добавить при помощи функции **arrows**():

```
arrows(c(0.9, 1.9, 2.9, 3.9, 4.9, 5.9), means + SDs,
       c(0.9, 1.9, 2.9, 3.9, 4.9, 5.9), means - SDs,
       angle = 90, code = 3, length = 0.05, lwd = 1,
       lend = "square")
```

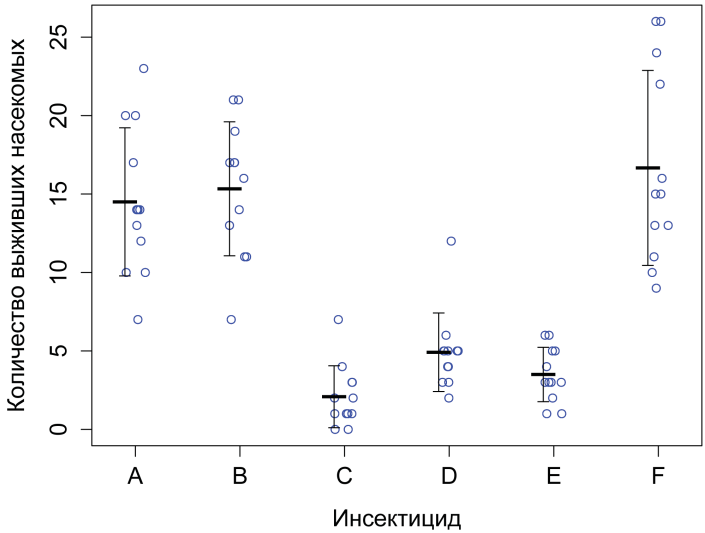


Рисунок 36

Новый график очень информативен – на нем изображены как все исходные наблюдения для каждой группы, так и сводная статистическая информация в виде соответствующих средних значений и стандартных отклонений.

Еще более полную картину мы можем получить, совместив на одном графике одномерные диаграммы рассеяния и диаграммы размахов (рис. 37). Сделать это можно всего несколькими строками R-кода, используя функции **boxplot**() и **stripchart**().

В качестве примера используем все те же данные по эффективности шести инсектицидных средств. Построим диаграмму размахов:

```
boxplot(count ~ spray,
# outline=FALSE отключает изображение точек-выбросов
          outline = FALSE, xlab = "Инсектициды",
          ylab = "Количество выживших насекомых",
          main = "Эффективность инсектицидов",
          data = InsectSprays)
```

При помощи функции **stripchart**() поверх получившегося графика можно наложить диаграмму рассеяния, которая будет отображать все исходные данные в виде точек. Трюк заключается в использовании аргумента add (добавить) этой функции. При add = TRUE точки будут добавлены к уже существующему графику:

```
stripchart(count ~ spray, method = "stack",
           data = InsectSprays, add = TRUE,
           pch = 1, col = "gray60", vertical = TRUE)
```

При необходимости можно легко повернуть весь график на 90 градусов (аргумент `horizontal = TRUE` функции **`boxplot`**`()`; в вызове функции **`stripchart`**`()` аргумент `vertical` можно пропустить – точки автоматически будут изображены в горизонтальной ориентации). Наконец, можно добавить немного «шума» в расположение точек (`method = "jitter"`; рис. 37):

```
boxplot(count ~ spray, data = InsectSprays, jitter = 0.2,
        outline = FALSE, horizontal = TRUE,
        ylab = "Инсектициды",
        xlab = "Количество выживших насекомых",
        main = "Эффективность инсектицидов")
stripchart(count ~ spray, method = "stack",
        data = InsectSprays, add = TRUE,
        pch = 1, col = "gray60")
```

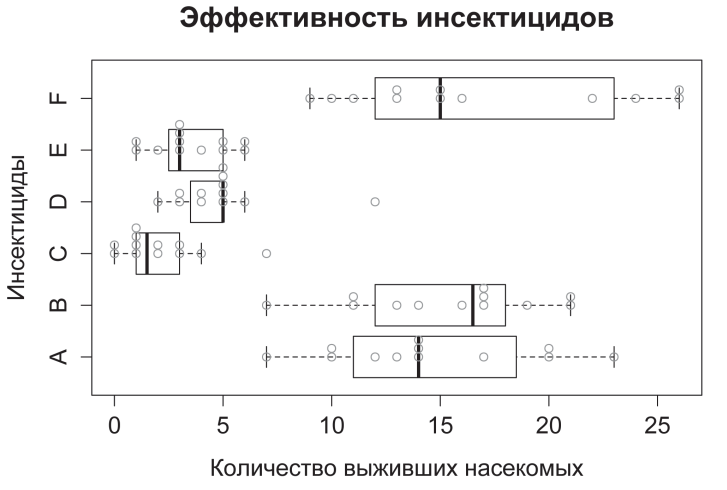


Рисунок 37

3.6. Категоризованные графики

Функция **`coplot`**`()`, входящая в базовую комплектацию R, предназначена для построения так называемых «*conditioning plots*», название которых можно перевести на русский язык как «*категоризованные графики*». Суть здесь сводится к тому, что анализируемые данные разбиваются на отдельные категории (например, в соответствии с уровнями какого-то фактора) и для каждой из них строится свой график (*панель*) определенного типа. Все эти графики затем объединяются на одном рисунке, что существенно облегчает выявление статистических закономерностей и структур в данных – подробности можно найти в книге Cleveland (1993, http://bit.ly/1BMak2K).

Функция **coplot**() обладает большим количеством управляющих аргументов (см. ?coplot), основными из которых являются следующие:

- formula – формула, описывающая взаимодействие между анализируемыми переменными;
- data – таблица со значениями переменных, указанных в formula;
- panel – функция, позволяющая задать тип и настроить внешний вид отдельных панелей категоризованного графика; по умолчанию эти панели представляют собой *диаграммы рассеяния* (см. примеры ниже);
- rows и columns – панели категоризованного графика размещаются в виде матрицы с количеством строк и столбцов, соответствующим значениям аргументов rows и columns; изменяя значения этих аргументов, можно упорядочить панели на графике необходимым образом;
- show.given – логическое значение (или вектор из двух логических значений в случае двух категориальных переменных, по которым разбиваются данные), позволяющее включать (TRUE) или отключать (FALSE) отображение «вывески» графика (см. ниже);
- number – количество интервалов, на которые разбиваются переменные a и b в случаях, если эти переменные не являются факторами;
- overlap – число (< 1), определяющее область перекрытия между данными, которые группируются в соответствии с уровнями количественных переменных a и b; если overlap < 0, соответствующая доля наблюдений на «стыках» групп *не будет* изображаться.

Формула вида y ~ x | a означает, что графики зависимости y от x должны быть построены для каждого уровня переменной a. В свою очередь, формула вида y ~ x | a * b показывает, что графики зависимости y от x должны быть построены для каждого сочетания уровней переменной a и b. Все переменные могут быть как количественными, так и качественными (факторами). Если x или y являются факторами, то их уровни будут автоматически преобразованы в численные значения (при помощи функции **as.numeric**()).

Кроме перечисленных аргументов, функция **coplot**() принимает также такие стандартные графические параметры, как col, pch, xlim, ylim и др. (см. раздел 3.1).

Применение функции **coplot**() мы продемонстрируем на примере данных о плотности популяции (экз./м^2) моллюска *Dreissena polymorpha* в озере Нарочь, Беларусь (Mastitsky & Veres, 2010, http://bit.ly/1H1JSlV). Учеты плотности популяции дрейссены были выполнены в мае и октябре 2005 г. на пяти глубинах озера вдоль восьми трансект. В приведенном ниже примере для простоты использована лишь часть данных (для трех из восьми трансект). Загрузить текстовый файл с данными можно из онлайн-сервиса Dropbox при помощи простой команды:

```
density <- read.delim(file = "http://bit.ly/1GozfvU",
          header = TRUE)

str(density)
'data.frame': 79 obs. of  4 variables:
```

```
$ Transect: Factor w/ 3 levels "A","B","C": 1 1 1 1 1 1 1 1 1 1 ...
$ Month : Factor w/ 2 levels "May","October": 1 1 1 1 1 1 1 1 1 1 ...
$ Depth   : num  0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 2 2 ...
$ Density : int  36 340 40 40 24 56 28 0 1560 40 ...
```

Созданная указанным образом таблица `density` содержит такие переменные, как `Transect` (трансекта), `Month` (месяц), `Depth` (глубина) и собственно `Density` (плотность популяции). Для начала изобразим зависимость плотности популяции дрейссены от глубины для каждого месяца:

```
coplot(Density ~ Depth | Month, data = density,
       xlab = c("Depth", "Month"), ylab = "Density")
```

Воспользовавшись аргументами `pch` и `col`, мы можем изменить внешний вид и цвет точек в каждой панели графика:

```
coplot(Density ~ Depth | Month, data = density,
       pch = 19, col = "blue",
       xlab = c("Depth", "Month"), ylab = "Density")
```

Прямоугольную область графика с названиями уровней фактора, по которым разбиваются данные, по-английски иногда называют «*shingle*» (вывеска). При необходимости можно изменить цвет заливки прямоугольников, в которые вписаны названия уровней фактора (см. аргумент `bar.bg`):

```
coplot(Density ~ Depth | Month, data = density,
     bar.bg = c(fac = "coral"), # fac - сокращение от "factor"
     pch = 19, col = "blue",
     xlab = c("Depth", "Month"), ylab = "Density")
```

Как это часто бывает с данными, характеризующими численность того или иного организма, данные по плотности популяции дрейссены в оз. Нарочь распределены резко асимметрично – большинство наблюдений сосредоточено в области малых значений плотности, и лишь некоторые наблюдения имеют очень высокие значения. Как результат плотное расположение точек в области низких значений затрудняет вывод о характере анализируемой связи. Для лучшей визуализации этой зависимости мы можем добавить сглаживающие кривые в каждую панель графика (рис. 38). Для этого воспользуемся аргументом `panel`, который, как отмечалось выше, позволяет выполнять тонкую настройку внешнего вида панелей путем прописывания соответствующей графической функции, например **`panel.smooth`**`()`:

```
coplot(Density ~ Depth | Month, data = density,
    panel = function(x, y, ... ) {
     panel.smooth(x, y, lty = 2, pch = 19, col = "blue",
     span = 0.6)}, # span регулирует кривизну сглаживающей кривой
       bar.bg = c(fac = "coral"), # fac - сокращение от "factor"
       pch = 19, col = "blue",
       xlab = c("Depth", "Month"), ylab = "Density")
```

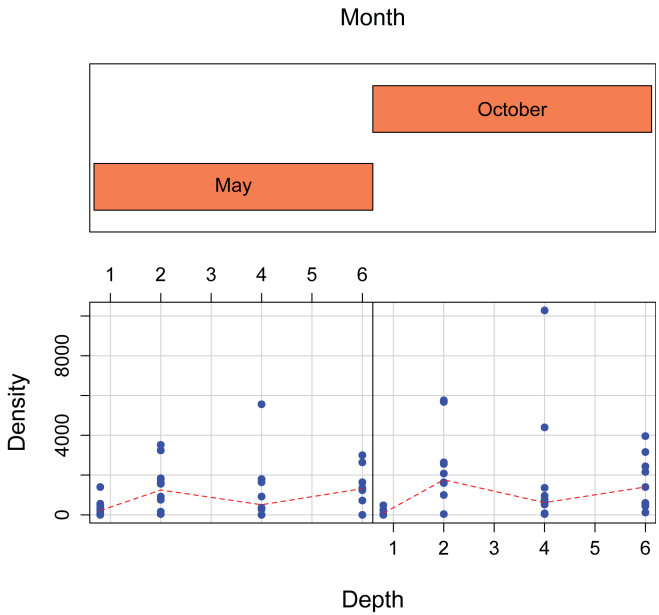


Рисунок 38

Пока закономерность вырисовывается нечетко – и все из-за уже упомянутой асимметричности распределения значений плотности популяции дрейссены. Мы можем несколько снизить эту асимметричность, преобразовав данные путем логарифмирования. Поскольку некоторые значения плотности равны нулю, перед логарифмированием добавим по единице к каждому наблюдению:

```
coplot(log(Density + 1) ~ Depth | Month, data = density,
    panel = function(x, y, ...) {
     panel.smooth(x, y, lty = 2, pch = 19, col = "blue",
     span = 0.6)}, # span регулирует кривизну сглаживающей кривой
       bar.bg = c(fac = "coral"), # fac - сокращение от "factor"
       pch = 19, col = "blue",
       xlab = c("Depth", "Month"), ylab = "Density")
```

На полученном графике гораздо лучше прослеживается тенденция к возрастанию плотности популяции дрейссены от глубины 0.8 к 2 м и к последующей стабилизации на более глубоких участках озера. При этом данная тенденция имела место как в мае, так и в октябре.

Вспомним, что учеты плотности популяции дрейссены проводились вдоль нескольких трансект. При помощи функции `coplot()` мы можем легко изобразить полученные данные в отношении не только времени отбора проб, но и трансект, вдоль которых эти пробы отбирались (обратите внимание на формулу `log(Density + 1) ~ Depth | Month * Transect`; рис. 39):

```
coplot(log(Density + 1) ~ Depth | Month * Transect,
       panel = function(x, y, ...) {
           panel.smooth(x, y, lty = 2, pch = 19,
           col = "blue", span = 0.6)},
       bar.bg = c(fac = "coral"), xlab = c("Depth", "Month"),
       ylab = c("Density", "Transect"), data = density)
```

Полученный категоризованный график дает достаточно полную пространственно-временную характеристику изучаемой зависимости.

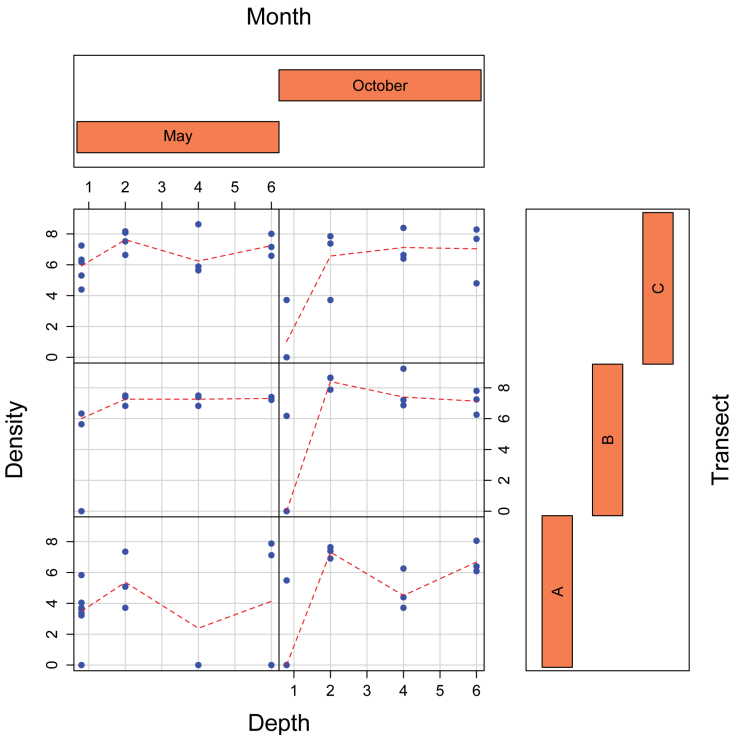


Рисунок 39

Как отмечалось выше, в формуле вида `y ~ x | a * b` любая из переменных может быть количественной, в том числе и переменные, стоящие по правую сторону от «|». Например, можно разбить наши данные не по месяцам, а по интервалам глубин (рис. 40):

```
coplot(log(Density+1) ~ Month | Depth,
       bar.bg = c(num = "coral"), # num - сокращение от numeric
       pch = 19, col = "blue", xlab = c("Month", "Depth"),
       ylab = c("Density", "Transect"), data = density)
```

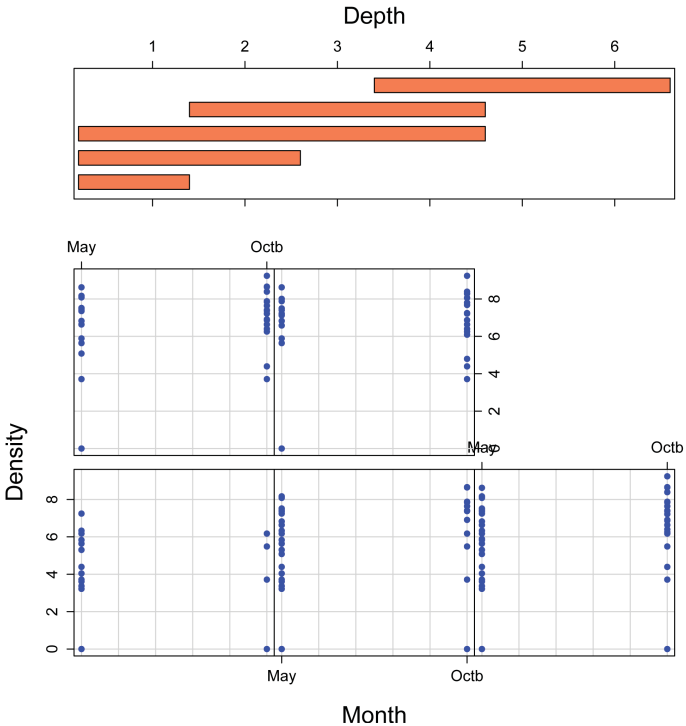


Рисунок 40

Важно понимать, как на полученном графике располагаются отдельные панели: они выстроены в порядке возрастания значений плотности популяции дрейссены. Первой является левая нижняя панель, за которой следуют средняя нижняя и правая нижняя панели. В верхнем ряду порядок аналогичный (то есть «читать» график нужно слева направо). «Вывеска» в верхней части графика является своего рода легендой, подсказывающей, какие именно данные изображены в каждой панели. Так, в левом нижнем углу представлены данные для диапазона глубин от 0 до примерно 1.5 м; средняя нижняя панель содержит данные для диапазона глубин от 0 до примерно 2.5 м и т. д. Практически все *автоматически* выделенные программой диапазоны глубин перекрываются.

Число диапазонов и уровень их перекрытия регулируются при помощи аргументов `number` и `overlap` соответственно (рис. 41):

```
coplot(log(Density + 1) ~ Month | Depth,
       number = 2, overlap = 0.1,
       bar.bg = c(num = "coral"), pch = 19, col = "blue",
       xlab = c("Month", "Depth"),
       ylab = c("Density", "Transect"), data = density)
```

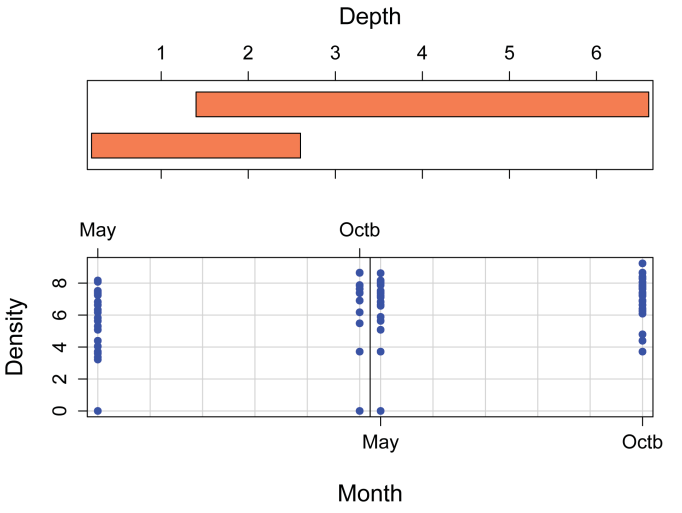


Рисунок 41

Глава 4

Описательная статистика, подгонка распределений и смежные задачи

4.1. Базовые функции для расчета параметров описательной статистики

Благодаря наличию набора специально созданных для этого функций расчет параметров описательной статистики в R не составляет никакого труда. Продемонстрируем их использование на примере рассмотренных в разделе 3.5 данных по характеристикам 32 моделей автомобилей (таблица `mtcars`, входящая в стандартный набор данных R):

```
data(mtcars)
head(mtcars)
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
```

Каждая модель автомобиля описана по 11 признакам, два из которых (`vs` и `am`) являются номинальными переменными (факторами) с уровнями, закодированными в виде 0 и 1 (подробнее см. `?mtcars`).

Для расчета *среднего арифметического*, *медианы*, *дисперсии*, *стандартного отклонения*, а также *минимального и максимального значений* в R служат функции **`mean`**`()`, **`median`**`()`, **`var`**`()`, **`sd`**`()`, **`min`**`()` и **`max`**`()` соответственно. Используем эти функции в отношении, например, параметра `mpg` – пробег автомобиля (в милях) в расчете на один галлон топлива:

```
# Среднее арифметическое:
mean(mtcars$mpg)
 [1] 20.1
```

```
# Медиана:
median(mtcars$mpg)
[1] 19.2
# Дисперсия:
var(mtcars$mpg)
[1] 36.3
# Стандартное отклонение:
sd(mtcars$mpg)
[1] 6.0
# Минимальное значение:
min(mtcars$mpg)
[1] 10.4
# Максимальное значение:
max(mtcars$mpg)
[1] 33.9
```

Специальной функции для расчета *стандартной ошибки среднего* в R нет, однако для этого вполне подойдут уже имеющиеся функции. Как известно, стандартная ошибка среднего рассчитывается как отношение стандартного отклонения к квадратному корню из объема выборки: $S_m = S/\sqrt{n}$. На языке R мы можем записать это следующим образом:

```
SEmpg = sd(mtcars$mpg)/sqrt(length(mtcars$mpg))
# функция length() возвращает число элементов в векторе mpg
SEmpg
[1] 1.065
```

Квантили рассчитываются в R при помощи функции **quantile**():

```
quantile(mtcars$mpg)
    0%    25%    50%    75%   100%
10.400 15.425 19.200 22.800 33.900
```

При настройках, заданных по умолчанию, выполнение указанной команды приведет к расчету минимального (10.4) и максимального (33.9) значений, а также трех *квартилей*, то есть значений, которые делят совокупность на четыре равные части – 15.4, 19.2 и 22.8.

Разница между первым и третьим квартилями носит название «*интерквартильный размах*» (*ИКР;* англ. «*interquartile range*»). ИКР является робастным аналогом дисперсии и может быть рассчитан в R при помощи функции **IQR**():

```
IQR(mtcars$mpg)
[1] 7.375
```

Функция **quantile**() позволяет рассчитать и другие квантили. Например, *децили* (значения, делящие совокупность на десять частей) можно получить следующим образом:

```
quantile(mtcars$mpg, p = seq(0, 1, 0.1))
   0%   10%   20%   30%   40%   50%   60%   70%   80%   90%  100%
10.40 14.34 15.20 15.98 17.92 19.20 21.00 21.47 24.08 30.09 33.90
```

В приведенной команде важен аргумент p (от «*probability*» – вероятность), при помощи которого был задан вектор чисел от 0 до 1 с шагом 0.1. Обратите внимание на то, что существует несколько способов оценки квантилей по выборочным данным. Подробнее об этом можно узнать в справочном файле по функции **quantile**() (доступен по команде ?quantile).

Отсутствующие значения в данных могут несколько усложнить вычисления. В целях демонстрации заменим 3-е значение переменной mpg на NA, то есть на обозначение, используемое в R для отсутствующих наблюдений, а затем попытаемся вычислить среднее значение:

```
mtcars$mpg[3] <- NA
# Просмотрим результат:
head(mtcars$mpg)
[1] 21.0 21.0   NA 21.4 18.7 18.1
# Попробуем рассчитать среднее значение для mpg:
mean(mtcars$mpg)
[1] NA
```

Ничего не вышло – вместо среднего значения программа выдала NA, что вполне логично. R не будет пропускать отсутствующие значения автоматически, если мы не включим соответствующую опцию – na.rm (от «*not available*» – недоступно и «*remove*» – удалить):

```
mean(mtcars$mpg, na.rm = TRUE)
[1] 20.0
```

Рассмотренный прием (то есть использование аргумента na.rm = TRUE) срабатывает в большинстве случаев. Одним из немногих исключений является функция **length**(), используемая для определения размера вектора. Аргумент na.rm у этой функции отсутствует, поэтому будут подсчитаны как имеющиеся, так и отсутствующие значения:

```
length(mtcars$mpg)
[1] 32
```

Если все же стоит задача подсчитать число неотсутствующих значений, то можно воспользоваться следующим приемом:

```
sum(!is.na(mtcars$mpg))
[1] 31
```

Трюк здесь заключается в использовании команды **is.na**(mtcars$mpg), которая проверяет каждое значение mpg и возвращает FALSE, если это значение *не* равно NA, и TRUE иначе. В сочетании с логическим оператором ! («не») функция sum используется для подсчета только тех значений mpg, которые не равны NA (логические TRUE конвертируются программой в 1, которые можно суммировать).

Существуют еще две функции, которые могут оказаться полезными при анализе свойств совокупностей, – **which.min**() и **which.max**(). Как следует из названий, эти функции позволяют выяснить порядковые номера элементов, обладающих мини-

мальным и максимальным значениями соответственно. Если минимальное/максимальное значение принимают несколько элементов в векторе, то будет возвращен порядковый номер первого элемента с этим значением. В случае с mpg имеем:

```
which.min(mtcars$mpg)
[1] 15
which.max(mtcars$mpg)
[1] 20
```

Видим, что минимальный и максимальный пробеги в расчете на галлон топлива имеют модели под номерами 15 и 20 соответственно. Выяснить названия этих моделей мы можем, совместив команды **which.min**() и **which.max**() с командой **rownames**() (от «*row*» – строка и «*names*» – имена):

```
rownames(mtcars)[which.min(mtcars$mpg)]
[1] "Cadillac Fleetwood"
rownames(mtcars)[which.max(mtcars$mpg)]
[1] "Toyota Corolla"
```

Подробнее приемы индексирования векторов в R были рассмотрены в разделе 2.2.

Как было отмечено в разделе 2.8, функции «apply-семейства», к которым относится и **tapply**(), позволяют выполнять векторизованные вычисления над определенными элементами таблиц данных, матриц или массивов (например, быстро вычислять среднее значение для каждого столбца или строки таблицы и т. п.).

Предположим, мы хотим выяснить средний объем двигателя (переменная disp, выраженная в кубических дюймах) у моделей с автоматической и ручной коробкой передач (переменная am; 1 – ручная коробка, 0 – автоматическая коробка). Функция **tapply**() позволяет сделать это следующим образом:

```
tapply(X = mtcars$disp, INDEX = mtcars$am, FUN = mean)
     0          1
290.38     143.53
```

Поскольку аргумент INDEX способен принимать список из нескольких факторов, для уровней которых рассчитываются значения функции FUN, мы можем усложнить приведенную выше команду:

```
tapply(X = mtcars$disp, INDEX = list(mtcars$am, mtcars$vs), FUN = mean)
       0      1
0 357.62 175.11
1 206.22  89.80
```

Аргумент FUN, как уже было отмечено, может принимать любые, в том числе и пользовательские, функции. Рассмотрим, например, расчет стандартных ошибок для средних значений объема двигателя у автомобилей с автоматической и ручной коробкой передач. Для начала создадим функцию SE для расчета стандартных ошибок (см. также пример выше):

```
SE <- function(x) {sd(x)/sqrt(length(x))}
```

Теперь совместим эту новую функцию с **tapply**():

```
tapply(X = mtcars$disp, INDEX = mtcars$am, FUN = SE)
    0     1
25.3  24.2
```

Таким образом, объем двигателя у моделей с автоматической коробкой передач составляет в среднем 290.4 ± 25.3, а у автомобилей с механической коробкой – 143.5 ± 24.2 кубического дюйма.

4.2. summary() и функции из дополнительных пакетов

В системе R имеется возможность и более быстрого расчета основных параметров описательной статистики. Для этого, в частности, служит функция общего назначения **summary**():

```
summary(mtcars$mpg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  10.40   15.35   19.20   20.00   22.15   33.90    1.00
```

Всего одной строки кода оказалось достаточно для получения минимального (Min) и максимального (Max) значений переменной mpg, медианы (Median), арифметического среднего (Mean), первого (1st Qu.) и третьего (3rd Qu.) квартилей, а также для нахождения числа отсутствующих значений (NA's). Более того, подобную сводку мы можем получить сразу для всей таблицы данных:

```
summary(mtcars)
      mpg             cyl             disp             hp
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
 1st Qu.:15.35   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
 Median :19.20   Median :6.000   Median :196.3   Median :123.0
 Mean   :20.00   Mean   :6.188   Mean   :230.7   Mean   :146.7
 3rd Qu.:22.15   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
 NA's   : 1.00
      drat             wt             qsec             vs
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
       am              gear            carb
 Min.   :0.0000   Min.   :3.000   Min.   :1.000
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
 Median :0.0000   Median :4.000   Median :2.000
 Mean   :0.4062   Mean   :3.688   Mean   :2.812
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

Результат выглядел бы замечательно, если бы не одно «но». Переменные vs и am являются факторами, но уровни их закодированы при помощи *чисел* 0 и 1. Система R не распознала эти две переменные как факторы и рассчитала соответствующие параметры описательной статистики как для обычных числовых переменных. Однако мы можем изменить такое поведение R, самостоятельно преобразовав vs и am в факторы при помощи функции **as.factor**():

```
mtcars$vs <- as.factor(mtcars$vs)
mtcars$am <- as.factor(mtcars$am)
# Проверим, удалась ли конвертация:
is.factor(mtcars$vs)
[1] TRUE
is.factor(mtcars$am)
[1] TRUE
```

Если применить функцию **summary**() к таблице mtcars теперь, то переменные vs и am будет распознаны программой как факторы: единственный способ описать их – это подсчитать число наблюдений для каждого уровня.

Рассмотренные выше функции позволяют получить достаточно полное представление об анализируемых выборках и таблицах данных. Однако для расчета некоторых параметров описательной статистики необходимы специальные функции, которые не входят в базовую версию R. С одним из таких параметров мы уже столкнулись – стандартная ошибка арифметического среднего. Другие примеры включают коэффициенты *эксцесса* («*kurtosis*») и *асимметрии* («*skewness*») – параметры, характеризующие форму распределения. Конечно, мы можем рассчитать эти величины по соответствующим формулам или даже написать собственные функции для этих целей. Однако это уже было сделано до нас – достаточно воспользоваться имеющимися дополнительными пакетами для R, например moments. Рассчитать коэффициенты эксцесса и асимметрии при помощи функций из этого пакета очень просто:

```
library(moments)
kurtosis(mtcars$mpg, na.rm = TRUE)
[1] 2.79
skewness(mtcars$mpg, na.rm = TRUE)
[1] 0.68
```

Многие пакеты для R имеют собственные функции, аналогичные стандартной **summary**(), предназначенные для вывода компактных описательных сводок по таблицам данных. Ниже приведено несколько примеров таких пакетов и функций (здесь и далее предполагается, что соответствующий пакет уже установлен на вашем компьютере). Подробности вывода результатов анализа здесь не обсуждаются – читателю предлагается самостоятельно ознакомиться со справочными материалами по соответствующим командам.

```
# Пакет Hmisc, функция describe():
describe(mtcars)
mtcars
11  Variables      32  Observations
```

```
------------------------------------------------
mpg
      n missing  unique  Mean    .05    .10    .25    .50    .75    .90    .95
     31       1      25    20  11.85  14.30  15.35  19.20  22.15  30.40  31.40
     lowest : 10.4 13.3 14.3 14.7 15.0, highest: 26.0 27.3 30.4 32.4 33.9
------------------------------------------------
#  И так далее по всем остальным переменным

# Пакет pastecs, функция stat.desc(); показаны семь переменных
stat.desc(mtcars)
                  mpg    cyl    disp      hp    drat     wt   qsec
nbr.val         31.00  32.00 3.2e+01   32.00  32.000  32.00  32.00
nbr.null         0.00   0.00 0.0e+00    0.00   0.000   0.00   0.00
nbr.na           1.00   0.00 0.0e+00    0.00   0.000   0.00   0.00
min             10.40   4.00 7.1e+01   52.00   2.760   1.51  14.50
max             33.90   8.00 4.7e+02  335.00   4.930   5.42  22.90
range           23.50   4.00 4.0e+02  283.00   2.170   3.91   8.40
sum            620.10 198.00 7.4e+03 4694.00 115.090 102.95 571.16
median          19.20   6.00 2.0e+02  123.00   3.695   3.33  17.71
mean            20.00   6.19 2.3e+02  146.69   3.597   3.22  17.85
SE.mean          1.10   0.32 2.2e+01   12.12   0.095   0.17   0.32
CI.mean.0.95     2.24   0.64 4.5e+01   24.72   0.193   0.35   0.64
var             37.28   3.19 1.5e+04 4700.87   0.286   0.96   3.19
std.dev          6.11   1.79 1.2e+02   68.56   0.535   0.98   1.79
coef.var         0.31   0.29 5.4e-01    0.47   0.149   0.30   0.10

# Пакет psych, функция describe.by() - расчет параметров
# описательной статистики для каждого уровня некоторого фактора.
# Показаны первые три переменные:
   describe.by(mtcars, mtcars$am)
group: 0
     var  n  mean     sd median trimmed    mad   min   max range  skew kurtosis    se
mpg    1 19  17.1   3.83   17.3    17.1   3.11  10.4  24.4  14.0  0.01    -0.33  0.88
cyl    2 19   7.0   1.54    8.0     7.1   0.00   4.0   8.0   4.0 -0.95    -0.24  0.35
disp   3 19 290.4 110.17  275.8   289.7 124.83 120.1 472.0 351.9  0.05    -1.01 25.28
------------------------------------------------
group: 1
     var  n  mean    sd median trimmed   mad  min   max range  skew kurtosis    se
mpg    1 12  24.5  6.42   23.7    24.5  7.93 15.0  33.9  18.9 -0.01    -1.35  1.85
cyl    2 13   5.1  1.55    4.0     4.9  0.00  4.0   8.0   4.0  0.87    -0.15  0.43
disp   3 13 143.5 87.20  120.3   131.2 58.86 71.1 351.0 279.9  1.33     2.17 24.19

# Пакет doBy, функция summaryBy(). Обладает мощным функционалом.
summaryBy(mpg + wt ~ cyl + vs, data = mtcars,
          FUN = function(x){ c(m = mean(x), s = sd(x)) } )
  cyl vs mpg.m mpg.s wt.m wt.s
1   4  0    26    NA  2.1    NA
2   4  1    NA    NA  2.3 0.60
3   6  0    21  0.75  2.8 0.13
4   6  1    19  1.63  3.4 0.12
5   8  0    15  2.56  4.0 0.76
```

4.3. Анализ выбросов

Дадим рабочее определение: под *выбросом* мы будем понимать наблюдение, которое «слишком» велико или «слишком» мало, по сравнению с большинством других имеющихся наблюдений.

Для визуального выявления выбросов обычно используют диаграммы размахов или точечные диаграммы Кливленда (см. разделы 3.3 и 3.5). Например, на рис. 42 представлена диаграмма Кливленда с данными о длине крыла у 1295 воробьев (Zuur et al., 2010, http://bit.ly/1E0Ub8J). Здесь таблица была предварительно упорядочена в соответствии с весом птиц, и поэтому облако точек имеет примерно *S*-образную форму:

```
Sparrows = read.table("http://bit.ly/1AOZbZs", sep = "\t", header = T)
dotchart(Sparrows$wingcrd, xlab = "Длина крыла (мм)",
         ylab = "Порядковый номер", lcolor = NA)
```

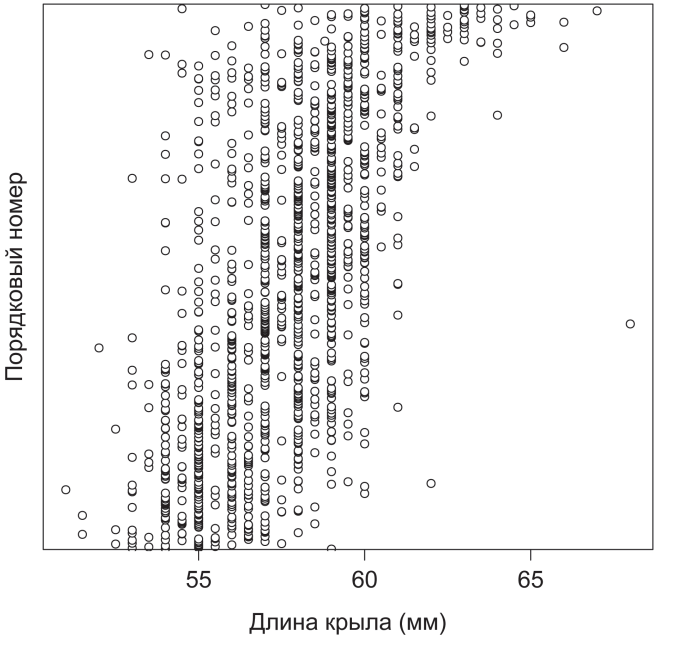


Рисунок 42

На рисунке хорошо выделяется точка, соответствующая длине крыла 68 мм. Однако это значение не следует рассматривать в качестве выброса, поскольку оно лишь незначительно отличается от других значений длины. Эта точка выделяется на общем фоне лишь потому, что исходные значения длины крыла были упорядочены по весу птиц. Соответственно, выброс скорее стоит искать среди значений

веса (то есть очень высокое значение длины крыла (68 мм) было отмечено у воробья, необычно мало весящего для этого).

Более строгий подход к определению выбросов состоит в оценке того, какое влияние эти необычные наблюдения оказывают на результаты анализа. При этом следует делать различие между необычными наблюдениями для зависимых и независимых переменных (предикторов). Например, при изучении зависимости численности какого-либо биологического вида от температуры большинство значений температуры может лежать в пределах от 15 до 20 °C, и лишь одно значение может оказаться равным 25 °C. Такой план эксперимента, мягко говоря, неидеален, поскольку диапазон температур от 20 до 25 °C будет исследован неравномерно. Однако при проведении реальных полевых исследований возможность выполнить измерения для высокой температуры может представиться только однажды. Что же тогда делать с этим необычным измерением, выполненным при 25 °C? При большом объеме наблюдений подобные редкие наблюдения можно исключить из анализа. Однако при относительно небольшом объеме данных еще большее его уменьшение может быть нежелательным с точки зрения статистической значимости получаемых результатов.

Альтернативой удалению необычных значений предиктора является нормализующее преобразование (чаще всего логарифмирование). В общем случае найти оптимальное решение позволяет так называемое преобразование Бокса-Кокса (Box & Cox, 1964, http://bit.ly/1Bg5EQK).

Преобразование Бокса-Кокса (БК) для случайной величины x в действительности представляет собой целое семейство степенных преобразований вида $x' = \frac{x^{\lambda} - 1}{\lambda}$, где показатель степени λ может принимать произвольные положительные или отрицательные значения. Поскольку деление на ноль приводит к неопределенности, то при $\lambda = 0$ используется логарифмическое преобразование $x'(\lambda) = \ln(x)$. Оптимальное значение λ обычно находят путем максимизации логарифма функции максимального правдоподобия.

Рассмотрим пример из раздела 1.3 с моллюсками *Dreissena polymorpha*, в которых подсчитывалась численность симбиотических инфузорий (CAnumber):

```
molluscs <- read.table("http://bit.ly/1wMGDOQ", header = TRUE, sep = "\t")

library(car)
# Поиск максимума функции правдоподобия и построение графика
# изменения параметра БК-трансформации для заданной модели
m.null <- lm(molluscs$CAnumber+1~1)
bc.null <- boxCox(m.null)
bc.null.opt <- bc.null$x[which.max(bc.null$y)]

paste("Оптимальная лямбда БК-преобразования:", bc.null.opt)
[1] "Оптимальная лямбда БК-преобразования: 0.181818182"

CAnumber_bc <- bcPower(molluscs$CAnumber+1, bc.null.opt)
```

```
par(mfrow = c(2,1))
boxplot(molluscs$CAnumber,horizontal = TRUE,col = "steelblue",
            xlab = "Численность инфузорий (экз)")
boxplot(CAnumber_bc, horizontal = TRUE, col = "coral",
            xlab = "Численность инфузорий (БК-преобразование)")
```

В нашем случае оптимальное значение $\lambda = 0.182$ (то есть наилучшее преобразование достаточно близко к логарифмическому; рис. 43).

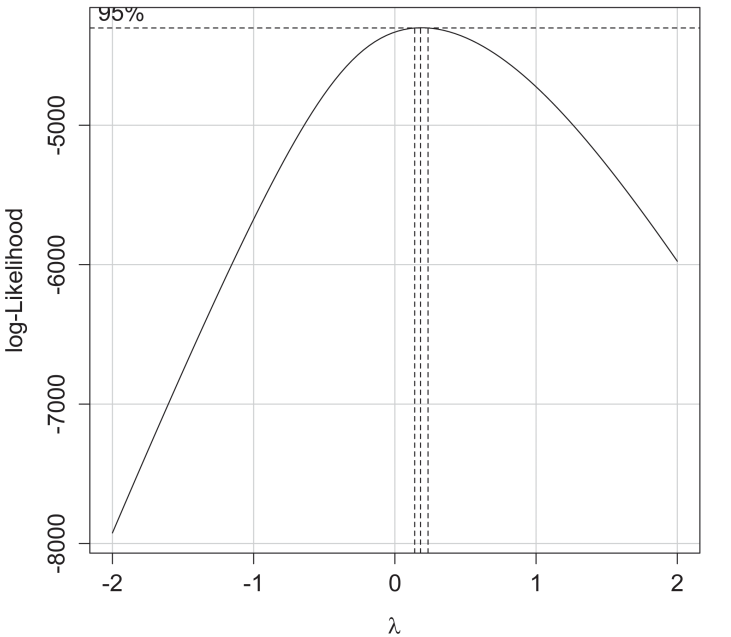


Рисунок 43

Эффект преобразования исходной переменной `CAnumber` по методу Бокса-Кокса проиллюстрирован на приведенном ниже рис. 44. Как видно из диаграммы, где приведены исходные данные, значительное число точек лежит за пределами верхней границы «усов», и, соответственно, эти точки можно считать потенциальными выбросами. Однако после БК-преобразования распределение значений `CAnumber` по форме приблизилось к нормальному, в связи с чем за пределами указанного интервала оказалось только одно наблюдение.

Имеются и формальные критерии для идентификации выбросов – например, критерии Шовене, Граббса, Пирса, Q-тест Диксона и др. (соответствующие функции для их реализации собраны в пакете `outliers`). Например, с использованием теста Граббса можно проверить нулевую гипотезу о том, что отмеченное в выборке максимальное значение не является выбросом:

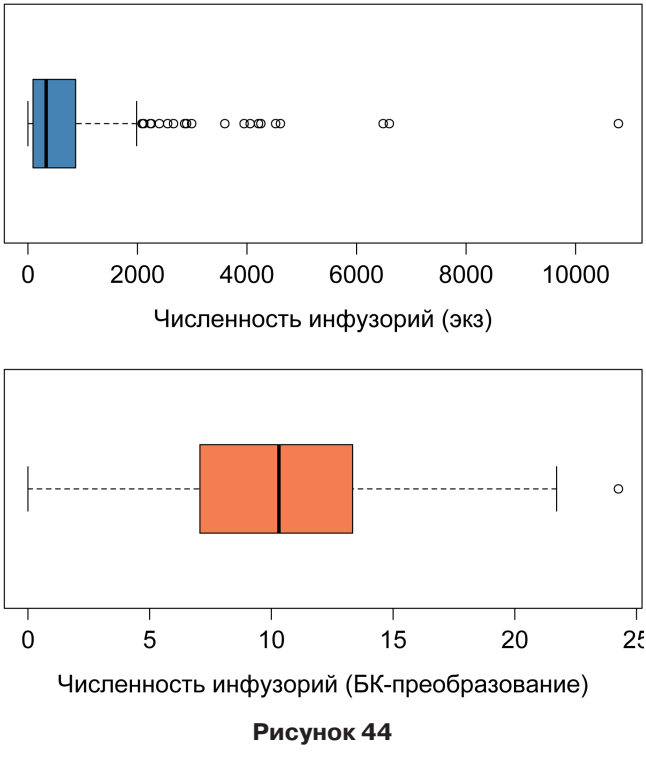


Рисунок 44

```
library(outliers)
grubbs.test(Моллюски$CAnumber, type = 10)
        Grubbs test for one outlier
data:  Моллюски$CAnumber
G = 10.8336, U = 0.7524, p-value < 2.2e-16
alternative hypothesis: highest value 10782 is an outlier

grubbs.test(CAnumber_bc, type = 10)
        Grubbs test for one outlier
data:  CAnumber_bc
G = 3.3267, U = 0.9767, p-value = 0.1961
alternative hypothesis: highest value 24.2569 is an outlier
```

Следует помнить, однако, что исходные значения зависимой переменной обычно представляют особый интерес при построении регрессионных моделей, и их преобразование может нарушить содержательный смысл проверяемых гипотез. Например, выполнив логарифмическое преобразование, можно легко продемонстрировать, что ежемесячный доход российского олигарха мало отличается от дохода пенсионера. Поэтому лучше попытаться подобрать метод анализа, который основан на распределении вероятностей, допускающем асимметрию разброса зна-

чений (например, гамма-распределение для непрерывных переменных или распределение Пуассона для дискретных количественных переменных).

В конечном счете решение об удалении необычных значений из анализа принимает сам исследователь. При этом он должен помнить о том, что причины для возникновения таких наблюдений могут быть разными. Так, удаление выбросов, возникших из-за неудачного планирования эксперимента (см. выше пример с температурой), может быть иногда вполне оправданным. Оправданным будет также удаление выбросов, возникших из-за явных ошибок при выполнении измерений.

В то же время необычные наблюдения среди значений зависимой переменной могут потребовать более тонкого подхода, особенно если они отражают естественную вариабельность этой переменной. В этой связи важно вести подробное документирование условий, в которых происходила экспериментальная часть исследования, – это может помочь интерпретировать «выбросы» в ходе анализа данных. Независимо от причин возникновения необычных наблюдений, в итоговом научном отчете (например, в статье) важно сообщить читателю как о самом факте выявления таких наблюдений, так и о принятых в их отношении мерах.

4.4. Заполнение пропущенных значений в таблицах данных

К сожалению, большинство статистических методов предполагает, что в ходе наблюдений были получены полностью укомплектованные матрицы, векторы и другие структуры с данными. Однако на практике пропуски в данных являются повсеместным явлением, и поэтому, прежде чем начать аналитические изыскания, необходимо привести обрабатываемые таблицы к «каноническому» виду, то есть либо удалить фрагменты объектов с недостающими элементами, либо заменить имеющиеся пропуски на некоторые разумные значения.

Несмотря на то что книги по статистике обычно скупо разбирают проблему исследования недостающих данных, в этой области существует впечатляющее множество подходов, методологий и их критических анализов (см., например, классическую монографию Little & Rubin (2002, http://bit.ly/1zWQhZV)). На практике процедура «борьбы с пропусками» обычно включает следующие шаги:

1) идентификация отсутствующих значений;
2) исследование закономерностей появления этих значений в данных;
3) формирование наборов данных, не содержащих пропусков (в результате удаления или замены соответствующих фрагментов).

Необходимо признать, что само обнаружение недостающих данных является здесь единственным однозначным шагом. Анализ причин наличия пропусков зависит от понимания исследователем процессов, приведших к формированию имеющихся данных. Решение о способе устранения пропущенных значений также будет зависеть от оценки того, какие процедуры приведут к самым надежным и точным результатам. Мы лишь бегло остановимся на некоторых функциях паке-

та mice, который является хорошим средством реализации всех шагов описанной выше процедуры.

Для конкретизации наших дальнейших рассуждений рассмотрим набор данных sleep из пакета VIM, составленный по результатам наблюдений за процессом сна у 62 млекопитающих разных видов (Allison & Chichetti, 1976, http://bit.ly/1EnjC7d). Авторы заинтересовались тем, почему потребность во сне у животных меняется от вида к виду и от каких экологических и таксономических переменных она зависит. Зависимые переменные включали продолжительность сна со сновидениями (Dream), сна без сновидений (NonD) и их сумму (sleep). Таксономические переменные включали массу тела (BodyWgt), вес мозга (BrainWgt), продолжительность жизни (Span) и время беременности (Gest). Экологические переменные состояли из 5-бальных оценок степени хищничества животных (Pred), меры защищенности их места для сна (Exp), изменяющегося от глубокой норы до полностью открытого пространства, и показателя риска (Danger), который основывался на логической комбинации предыдущих двух переменных (Pred и Exp). Отметим, что данный пример заимствован из книги Kabacoff (2011) и приводится здесь в сокращенном виде. С дополнительными деталями можно ознакомиться как в оригинальной работе (Kabacoff, 2011), так и в русскоязычном переводе этой отличной книги (Кабаков, 2014).

Идентификация пропущенных значений легко выполняется с использованием функций **is.na**() и **complete.cases**():

```
data(sleep, package = "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

# список строк, в которых нет пропущенных значений:
sleep[complete.cases(sleep),]
# список строк, где есть хотя бы одно пропущенное значение:
sleep[!complete.cases(sleep),]

sum(is.na(sleep$Dream))
[1] 12
```

Таким образом, мы имеем 42 полностью укомплектованные строки и 20 строк, имеющих хотя бы одно пропущенное значение. Из них 12 недостающих значений принадлежат переменной Dream.

Дополнительную статистическую информацию о пропусках предоставляет функция **md.pattern**(), а функции **aggr**(), **matrixplot**() и **scattMiss**() позволяют графически изобразить закономерности во встречаемости отсутствующих наблюдений:

```
library(mice)
md.pattern(sleep)
   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
```

Значения 0 в теле приведенной таблицы соответствуют недостающим значениям, причем в первой строке пропусков нет, а последующие строки упорядочены по числу их появления. Первый столбец указывает число случаев в каждой строке исходных данных, а последний столбец – число переменных с отсутствующими значениями в каждой строке. Например, в четвертой строке в исходных данных содержится 9 случаев с одновременно отсутствующими NonD и Dream. Набор данных содержит в общей сложности (42×0) + (2×1) + ... + (1×3) = 38 отсутствующих значений, а в последней строке приведена частота пропусков для каждой переменной.

На диаграмме, полученной при помощи функции **matrixplot**(), числовые данные масштабируются к интервалу [0, 1] и представлены уровнями шкалы яркости, причем более темными цветами показаны большие значения (рис. 45). По умолчанию недостающие значения представлены красным цветом. График является интерактивным: нажатие на столбец отсортирует матрицу по этой переменной (на рисунке – в порядке убывания BodyWgt).

```
matrixplot(sleep)
```

Такая визуализация позволяет увидеть, связаны ли пропуски одной или более переменных с имеющимися значениями других признаков. Например: при низких значениях массы тела или мозга (BodyWgt, BrainWgt) пропуски среди значений Dream и NonD отсутствуют. Эти зависимости можно выразить количественно при помощи коэффициентов корреляции следующим образом:

```
# Формируем матрицу со значениями 1 в местах пропусков
x <- as.data.frame(abs(is.na(sleep)))
y <- x[, which(colSums(x) > 0)]

print(cor(y),4)
         NonD    Dream    Sleep      Span     Gest
NonD   1.0000  0.90711  0.48626  0.01520 -0.14183
Dream  0.9071  1.00000  0.20370  0.03752 -0.12865
Sleep  0.4863  0.20370  1.00000 -0.06897 -0.06897
Span   0.0152  0.03752 -0.06897  1.00000  0.19828
Gest  -0.1418 -0.12865 -0.06897  0.19828  1.00000
```

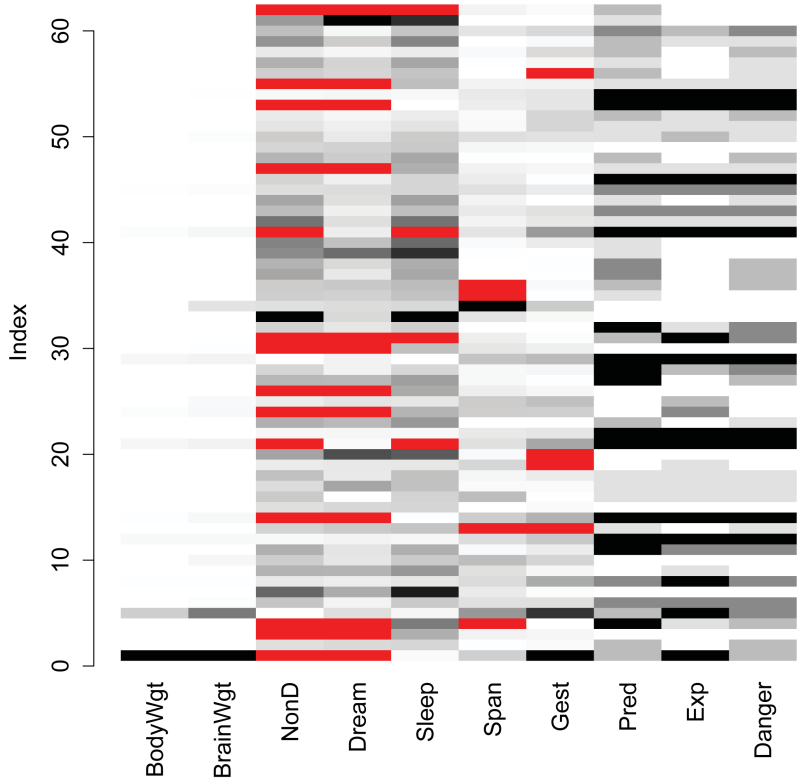


Рисунок 45

Полученная корреляционная матрица показывает, что появление недостающих значений для `NonD` и `Dream` взаимосвязано (r = 0.9071). Это естественно: если мы имеем фактическое значение одной из этих переменных, мы можем рассчитать другое с использованием алгебраического соотношения `NonD + Dream = Sleep`.

```
cor(sleep, y, use = "pairwise.complete.obs")
              NonD    Dream     Sleep     Span     Gest
BodyWgt    0.22683  0.22259  0.001685 -0.05832 -0.05397
BrainWgt   0.17946  0.16321  0.007859 -0.07921 -0.07333
NonD           NA       NA        NA -0.04315 -0.04553
Dream     -0.18895       NA -0.188952  0.11699  0.22775
Sleep     -0.08023 -0.08023        NA  0.09638  0.03976
Span       0.08336  0.05981  0.005239       NA -0.06527
Gest       0.20239  0.05140  0.159702 -0.17495       NA
Pred       0.04758 -0.06834  0.202463  0.02314 -0.20102
Exp        0.24547  0.12741  0.260773 -0.19292 -0.19292
Danger     0.06528 -0.06725  0.208884 -0.06666 -0.20444
```

В первом столбце последней корреляционной матрицы можно заметить, что частота пропусков регистрации сна без сновидений с большей вероятностью связана с высоким весом тела BodyWgt ($r = 0.227$), продолжительностью беременности Gest ($r = 0.202$) и защищенностью лежбища Exp ($r = 0.245$).

На использовании корреляционных отношений основан популярный метод заполнения пропусков с использованием регрессионных моделей. Другой подход, называемый «множественным присвоением» («*multiple imputation*»), основан на многократно повторяющемся моделировании данных с использованием методов Монте-Карло.

Функция **mice**() из одноименного пакета реализует метод Гиббса («*Gibbs sampler*»), который представляет собой способ формирования выборок (x_1; ...; x_n) из заданных распределений $p(x)$ m-мерных переменных путем многократного формирования выборок имитируемых значений. По умолчанию каждая переменная, содержащая недостающие величины, моделируется в зависимости от всех других переменных в наборе данных. Полученные уравнения используются для предсказания наиболее вероятных значений для пропущенных данных. Процесс повторяется, пока не будет достигнут определенный критерий сходимости. Для каждой переменной пользователь может выбрать форму модели предсказания, называемую «элементарным методом присвоения».

Процесс заполнения пропусков реализуется в несколько этапов:

- функция **mice**() обрабатывает mydata – матрицу или таблицу данных с пропущенными значениями и возвращает объект imp, содержащий несколько полных наборов данных (например, $m = 5$), незначительно отличающихся между собой значениями пропусков, заполненных в ходе процедуры Гиббса:

```
imp <- mice(mydata, m)
```

- функция **with**() используется для применения статистической модели analysis к каждому набору данных из imp; это может быть линейная **lm**() или обобщенная линейная **glm**() модель, аддитивная модель **gam**() или любая иная, задаваемая пользователем модель:

```
fit <- with(imp, analysis)
```

- функция **pool**() комбинирует результаты и объединяет их в заключительную модель, в которой стандартные ошибки и p-значения сбалансированы относительно неопределенности, порожденной пропусками:

```
pooled <- pool(fit)
```

- наконец, функция **complete**() создает набор данных с заполненными пропусками: **complete**(imp, action=#).

Применим процедуру множественного присвоения к набору данных sleep:

```
imp <- mice(sleep, seed = 1234)
fit <- with(imp, lm(Dream ~ Span + Gest))
pooled <- pool(fit)
summary(pooled)
```

По результатам анализа моделей из объекта pooled можно установить, что показатель Dream статистически значимо зависит от переменной Gest, тогда как его зависимость от Span незначима. Исследуя объекты, созданные в ходе этого анализа, можно получить дополнительную информацию о процессе восстановления отсутствующих наблюдений. Например, рассмотрев данные объекта imp, можно установить, что восстановлены отсутствующие значения для следующих показателей:

```
NonD Dream Sleep  Span  Gest
   3     4     5     6     7
"pmm" "pmm" "pmm" "pmm" "pmm"
```

При этом был использован метод "pmm" («*predictive mean matching*»). В нашем примере метод заполнения пропусков был выбран по умолчанию, но пользователь может самостоятельно задать любой иной алгоритм из 19 возможных с помощью параметра method функции **mice**().

Для конкретного восстанавливаемого показателя в каждом из пяти имитируемых наборов данных мы можем посмотреть и проанализировать результаты расчетов:

```
imp$imp$Dream
    1   2   3   4   5
1  0.5 0.5 0.5 0.5 0.0
3  2.3 2.4 1.9 1.5 2.4
4  1.2 1.3 5.6 2.3 1.3
14 0.6 1.0 0.0 0.3 0.5
24 1.2 1.0 5.6 1.0 6.6
26 1.9 6.6 0.9 2.2 2.0
30 1.0 1.2 2.6 2.3 1.4
31 5.6 0.5 1.2 0.5 1.4
47 0.7 0.6 1.4 1.8 3.6
53 0.7 0.5 0.7 0.5 0.5
55 0.5 2.4 0.7 2.6 2.6
62 1.9 1.4 3.6 5.6 6.6
```

Наконец, мы можем сохранить любой из 5 полных наборов данных для дальнейшего использования (например, нам понравился набор № 3):

```
sleep_imp3 <- complete(imp, action = 3)
head(sleep_imp3)
   BodyWgt BrainWgt NonD Dream Sleep Span Gest Pred Exp Danger
1 6654.000   5712.0  2.1   0.5   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 10.6   1.9  12.5 14.0   60    1   1      1
4    0.920      5.7 11.0   5.6  16.5  4.7   25    5   2      3
5 2547.000   4603.0  2.1   1.8   3.9 69.0  624    3   5      4

sleep[!complete.cases(sleep_imp3),]
save(sleep_imp3, file = "sleep_imp.Rdata")
```

Как видим, все пропуски в нашей таблице оказались заполненными.

Среда R поддерживает также несколько других подходов для восстановления недостающих данных. В иных обстоятельствах могут быть весьма полезными функции **mitools**(), **pan**(), **mix**() из одноименных пакетов, а также **aregImpute**() и **transcan**() из пакета Hmisc.

4.5. Воспроизводимость результатов при использовании генератора случайных чисел

В практике статистического анализа данных часто приходится иметь дело с необходимостью генерации случайных чисел, подчиняющихся тому или иному закону распределения вероятностей (например, при необходимости случайным образом отобрать небольшую выборку из массивной таблицы данных, при использовании бутстреп-метода, метода Монте-Карло, метода Гиббса (см. предыдущий раздел) и т. п.).

В системе R реализован целый ряд алгоритмов, позволяющих генерировать случайные числа (см. ?RNGkind). По умолчанию используется один из наиболее быстрых генераторов – так называемый «вихрь Мерсенна» («*Mersenne twister*»). Важно понимать, что практически все эти алгоритмы детерминированы, и поэтому генерируемые с их помощью числа правильнее называть *псевдослучайными*.

Генератор псевдослучайных чисел (ГПСЧ) начинает свою работу с определенной точки в пространстве возможных чисел. Эта точка называется *начальным числом* («*seed*»). В R имеется возможность зафиксировать это число так, что при повторном использовании ГПСЧ будет генерироваться точно та же последовательность чисел, что и в первый раз. Это может оказаться полезным в случаях, когда исследователь по некоторым причинам желает иметь точно воспроизводимые результаты, получаемые с использованием ГПСЧ.

В качестве примера создадим таблицу example с двумя столбцами. В первом столбце будут храниться коды уровней некоторого гипотетического фактора Factor (три уровня: A, B, и C). Для каждого из этих уровней сгенерируем (псевдо) случайным образом по 300 нормально распределенных значений с разными средними и стандартными отклонениями:

```
example = data.frame(
    Factor = rep(c("A", "B", "C"), each = 300),
    Variable = c(rnorm(300, 5, 2), rnorm(300, 4, 3), rnorm(300, 2, 1)))
tapply(example$Variable, example$Factor, summary)
$A
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.9135  3.5950  5.0190  4.8520  6.2250 10.4900
$B
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -4.407   2.194   4.226   4.028   5.962  13.750
$C
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -0.517   1.365   1.997   1.975   2.582   4.458
```

Для построения диаграмм с распределениями полученных значений в каждой из трех групп A, B, и C используем ggplot2 – один из наиболее популярных графических пакетов для R:

```
library(ggplot2)
p <- ggplot(example, aes(x = Variable))
(p <- p + geom_density(aes(fill = Factor), alpha = 1/2))
```

Если бы мы повторили создание таблицы example описанным выше способом еще несколько раз, мы получили бы очень похожие, но все-таки несколько различающиеся распределения для групп A, B и C (рис. 46). На рисунке показан пример кривых распределения для четырех повторностей таблицы example, созданных при помощи ГПСЧ.

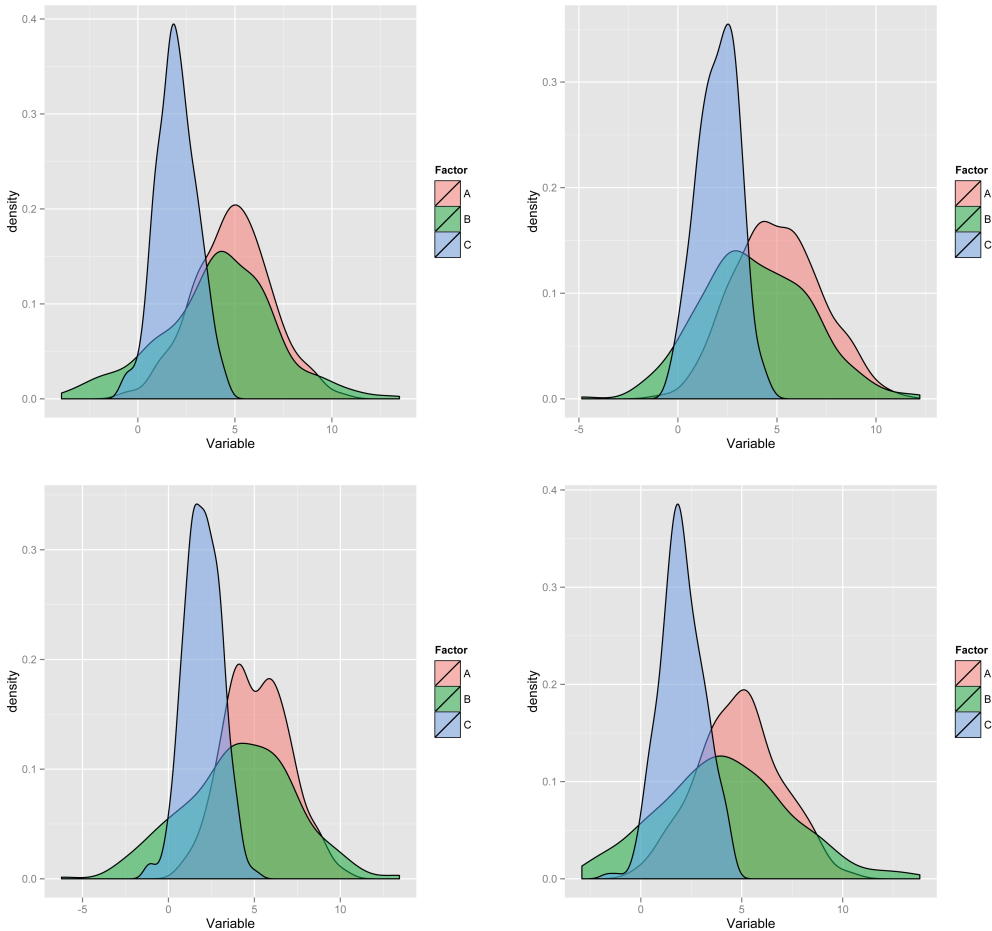


Рисунок 46

Для получения идентичных последовательностей чисел при использовании ГПСЧ в R имеется функция **set.seed**() (set – задать, установить, seed – начальное число). Как следует из названия, эта функция фиксирует начальную точку для запуска алгоритма генерации (псевдо)случайных чисел. В качестве аргумента функции указывают произвольное целое число. При повторном выполнении кода, которому предшествует команда **set.seed**(), мы будем всегда получать одинаковые результаты.

```
set.seed(1020)
example = data.frame(
      Factor = rep(c("A", "B", "C"), each = 300),
      Variable = c(rnorm(300, 5, 2), rnorm(300, 4, 3), rnorm(300, 2, 1)))
```

Повторив приведенный код четыре раза, мы получили бы четыре идентичные таблицы и, соответственно, четыре идентичных графика, изображающих распределения плотностей вероятности (рис. 47).

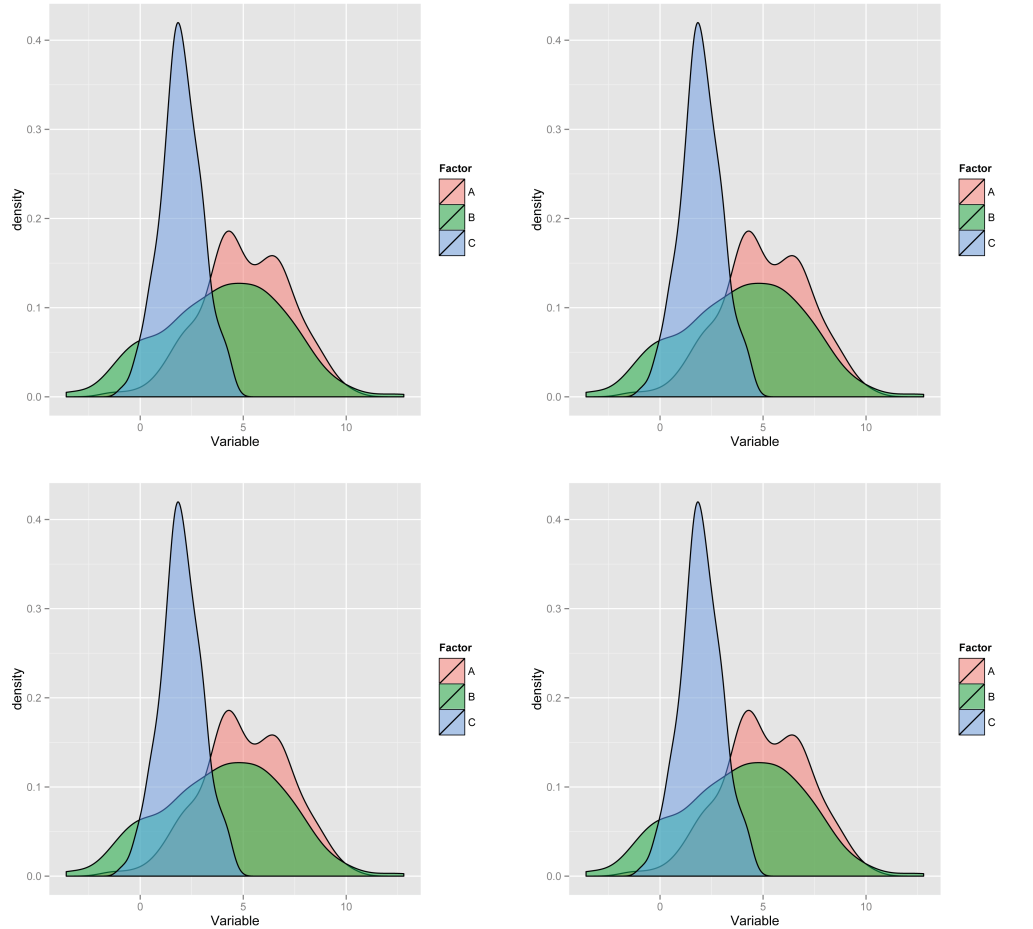


Рисунок 47

4.6. Законы распределения вероятностей, реализованные в R

В базовой версии R имеются функции для работы с целым рядом распространенных законов распределения вероятностей. В зависимости от назначения имена этих функций начинаются с одной из следующих четырех букв:

- `d` («*density*» – плотность): функции плотности вероятности (или «функции распределения масс» для дискретных количественных переменных);
- `p` («*probability*» – *вероятность*): функции распределения вероятностей;
- `q` («*quantile*» – квантиль): функции для нахождения квантилей того или иного распределения;
- `r` («*random*» – случайный): функции для генерации случайных чисел в соответствии с параметрами того или иного закона распределения вероятностей.

В частности, в базовой версии R реализованы следующие законы распределения вероятностей:

- бета-распределение (см. `dbeta`);
- биномиальное распределение (включая распределение Бернулли) (`dbinom`);
- распределение Коши (`dcauchy`);
- распределение хи-квадрат (`dchisq`);
- экспоненциальное распределение (`dexp`);
- распределение Фишера (`df`);
- гамма-распределение (`dgamma`);
- геометрическое распределение (как частный случай отрицательного биномиального распределения) (`dgeom`);
- гипергеометрическое распределение (`dhyper`);
- логнормальное распределение (`dlnorm`);
- полиномиальное (или мультиномиальное) распределение (`dmultinom`);
- отрицательное биномиальное распределение (`dnbinom`);
- нормальное распределение (`dnorm`);
- распределение Пуассона (`dpois`);
- распределение Стьюдента (`dt`);
- равномерное распределение (`dunif`);
- распределение Вейбулла (`dweibull`).

В качестве примера рассмотрим `d`-, `p`-, `q`- и `r`-функции, предназначенные для работы с нормальным распределением. Предположим, что мы имеем дело с непрерывной количественной величиной X, значения которой распределены в соответствии со стандартным нормальным распределением (среднее значение = 0, стандартное отклонение = 1). Функция плотности вероятности представляет собой такую функцию $f(x)$, что для любых двух значений a и b (при $a \le b$)

$$P(a \le x \le b) = \int_a^b f(x)dx.$$

Иными словами, вероятность того, что некоторая случайная величина X принимает значение, лежащее в интервале $[a, b]$, равна площади под кривой плотности вероятности, ограниченной этим интервалом. Заметим, что, по определению, площадь под всей кривой плотности вероятности равна 1. Функция **drnorm**() позволяет рассчитать значение функции плотности вероятности для заданного значения (или сразу для вектора из нескольких значений) X. Так, для $x = -1$ в случае со стандартным нормальным распределением получаем

```
dnorm(-1)
[1] 0.2419707
```

Кумулятивная функция распределения (или просто «функция распределения») описывает вероятность того, что вещественнозначная случайная переменная X примет *какое-либо* значение, не превышающее либо равное x. Для нашего примера получаем (рис. 48):

```
pnorm(-1)
[1] 0.1586553
```

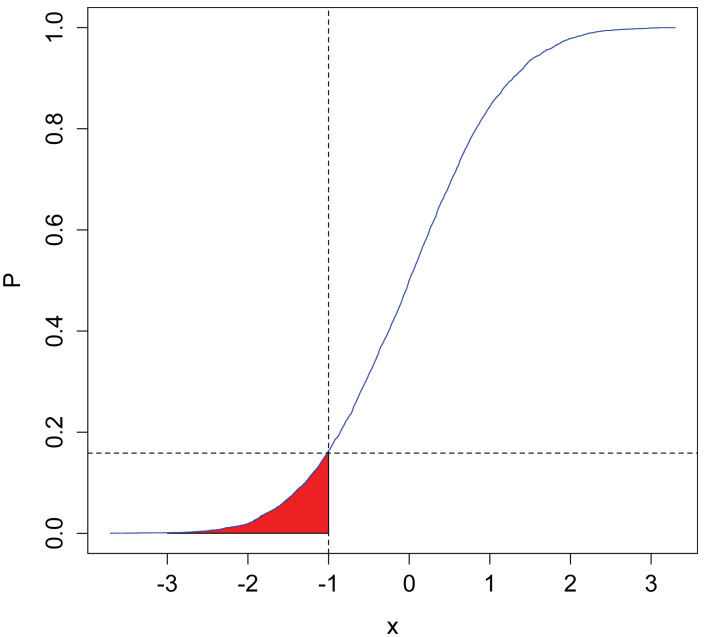


Рисунок 48

Для вычисления квантилей нормального распределения служит функция **qnorm**(). Например, следующим образом можно вычислить нижний и верхний квартили стандартного нормального распределения:

```
qnorm(p = c(0.25, 0.75))
[1] -0.6744898  0.6744898
```

Наконец, функция **rnorm**() служит для случайной генерации совокупностей нормально распределенных чисел. Сгенерируем совокупность из 10 значений, представляющих стандартное нормальное распределение:

```
rnorm(10, mean = 0, sd = 1)
 [1] -2.69409418  0.09407048 -0.19622289 -1.06719782 -0.55212590
 [6] -0.94019419 -0.14916494  0.17760798 -0.29358930 -0.77420692
```

d-, p-, q- и r-функции для других распределений работают сходным образом. В заключение стоит отметить, что в дополнительных пакетах можно найти функции для работы со многими другими законами распределения вероятностей. Особенно полезными, в частности, могут оказаться пакеты VGAM, actuar, gamlss и ActuDistns.

4.7. Подбор закона и параметров распределения в R

Подбор закона распределения заключается в нахождении математической функции, которая бы наилучшим образом описывала характер распределения экспериментальных данных. Исследователь часто сталкивается с такой задачей: у него есть некоторые наблюдения количественного показателя x_1, x_2, ... x_n, и он желает проверить, принадлежит ли полученная им выборка некой теоретической генеральной совокупности с функцией плотности вероятности $f(x, \theta)$, где θ является вектором параметров, оцениваемых по имеющимся данным.

Можно выделить 4 шага при подборе распределений (Ricci, 2005):

1) выбор модели: выдвигается гипотеза о принадлежности выборки некоторому семейству распределений;
2) оценка параметров теоретического распределения;
3) оценка качества приближения;
4) проверка согласия между наблюдаемыми и ожидаемыми значениями с использованием статистических тестов.

Имея одну конкретную выборку и задав форму некоторого теоретического распределения, можно рассчитать точечные оценки его параметров (например, среднее и стандартное отклонения нормального распределения). Для этого используются следующие процедуры оценивания: *а*) по аналогии, *б*) метод моментов и *в*) метод максимального правдоподобия.

Метод «по аналогии» мы использовали, рассчитав в разделе 4.1 арифметическое среднее для пробега автомобиля (mtcars$mpg), допустив для этой выборки нормальное распределение. Метод моментов – это техника конструирования оценок неизвестных параметров, основанная на предполагаемом соотношении выборочных моментов для заданных распределений. В общем случае для использования метода нужно составить нелинейное уравнение (или систему уравнений, в зависимости от числа неизвестных параметров), где выборочные моменты при-

равниваются к соответствующим теоретическим. Нахождение корней уравнений выполняется аналитическими методами или с использованием таких функций, как, например, **multiroot**().

Сформируем случайную выборку из гамма-распределения с использованием функции **rgamma**() и параметрами масштаба $\lambda = 0.5$ (аргумент rate) и формы $\gamma = 3.5$ (аргумент shape). Аналитическое решение для оценки этих параметров зависит от первых двух моментов нормального распределения (среднего и дисперсии): $\hat{\lambda} = \bar{x}/s^2$ и $\hat{\gamma} = \bar{x}^2/s^2$. В R получаем:

```
library(MASS)
set.seed(0)
x.gam <- rgamma(200, rate = 0.5, shape = 3.5)

# Аналитический путь
med.gam <- mean(x.gam)      # выборочное среднее
var.gam <- var(x.gam)       # выборочная дисперсия

(l.est <- med.gam/var.gam)  ## оценка лямбда
[1] 0.5257432

(g.est <- ((med.gam)^2)/var.gam)  ## оценка гамма
[1] 3.533591

# Общий подход
library(rootSolve)
f1 <- function(x){ c(F1 = x[1]/x[2] - med.gam,
                     F2 = x[1]/x[2]^2 - var.gam) }
multiroot(f1, c(3, 0.6))
$root
[1] 3.5335907 0.5257432
```

При оценке параметров известного семейства вероятностных распределений гораздо более распространенным является метод максимального правдоподобия (MLE, от «*Maximum Likelihood Estimation*»), так как максимально правдоподобная оценка имеет большую вероятность оказаться ближе к истинному значению оцениваемой величины.

Принцип этого метода состоит в том, что в качестве «наиболее правдоподобного» значения параметра берут такое значение Θ, которое максимизирует вероятность получить при n опытах имеющуюся выборку $X = (x_1, \ldots x_n)$. Предположим, что нам известен закон распределения случайной величины X, определяемый параметром Θ, но неизвестно численное значение этого параметра, и пусть $p(x_i, \Theta)$ – вероятность того, что в результате испытания величина X примет значение x_i. *Функцией правдоподобия* случайной величины X называют функцию аргумента Θ, определяемую по формуле:

$$L(x_1, x_2, ..., x_n; \Theta) = p(x_1, \Theta)p(x_2, \Theta) \ldots p(x_n, \Theta).$$

В качестве искомой точечной оценки параметра Θ принимают такое его значение, при котором функция правдоподобия достигает максимума: $\theta_{MLE} = \text{argmax} L(\mathbf{x}|\theta)$, $\theta \in \Theta$.

Для поиска экстремума необходимо найти частную производную функции правдоподобия, приравняв ее к нулю: $\partial L(\mathbf{x}, \theta_0)/\partial\theta = 0$. Если вторая производная в критической точке отрицательна, то это – точка максимума, соответствующая искомому значению параметра. Если нам необходимо найти k параметров, то формируется и решается система из k уравнений правдоподобия.

Поскольку функции L и $\ln(L)$ достигают максимума при одном и том же значении Θ, с математической точки зрения удобнее искать максимум логарифмической функции правдоподобия. Например, в случае гамма-распределения выражение для функции правдоподобия будет иметь вид:

$$\ln[L(x_1, ..., x_n, \gamma, \lambda)] = -n\gamma\ln(\lambda) - n\ln(\Gamma(\gamma)) + (\gamma - 1)\sum_n \ln x_i - \lambda\sum_n x_i.$$

В большинстве функций R, специально предназначенных для оценки параметров, по выбору пользователя реализуется как метод моментов, так и метод максимального правдоподобия (более того, последний используется по умолчанию). Для подробного рассмотрения практических коллизий, возникающих при подгонке распределений, создадим выборку значений из смеси распределений и постараемся выбрать наиболее подходящую теоретическую модель. При оценке параметров будем использовать функции **fitdistr**() из пакета MASS и **fitdist**() из пакета fitdistrplus.

Рассмотрим имитацию случайной выборки из распределения Вейбулла с двумя параметрами: параметр формы shape = λ и параметр масштаба scale = $1/\lambda$. Оба аргумента – положительные конечные числа, но значение scale сделаем переменным, изменяющимся по биномиальному закону. Выполним генерацию 100 значений такой выборки:

```
set.seed(1946)
x <- sort(rweibull(100, 2, (1 + 1.21*rbinom(100, 1, 0.05))))

x  # Отсортированные значения
 [1] 0.03073383 0.11966667 0.15825498 0.16137953 0.17215545 0.17739292 0.23748408
...
[92] 1.59118211 1.84739051 1.85831686 1.90531232 1.98264573 2.02601002 2.11484324
[99] 2.73019016 3.29075676

summary(x)  # Выборочные характеристики
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.03073 0.57190 0.79260 0.89500 1.06700 3.29100
```

Выведем гистограмму полученной выборки, совмещенную с кривой ядерной функции плотности распределения (рис. 49):

```
hist(x, freq = FALSE, breaks = 15, col = "grey88",
     main = "Гистограмма и ядерная плотность")
lines(density(x), lwd = 2)
```

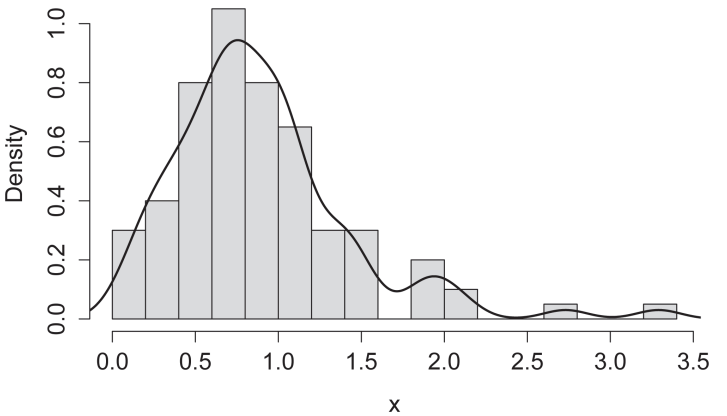


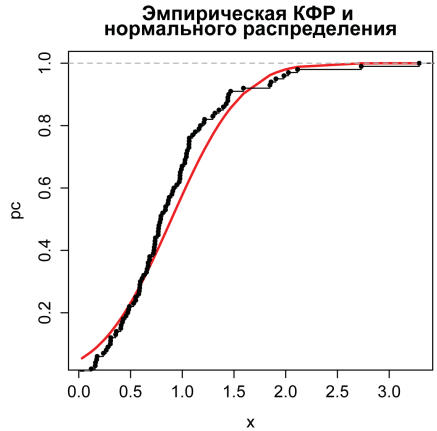
Рисунок 49

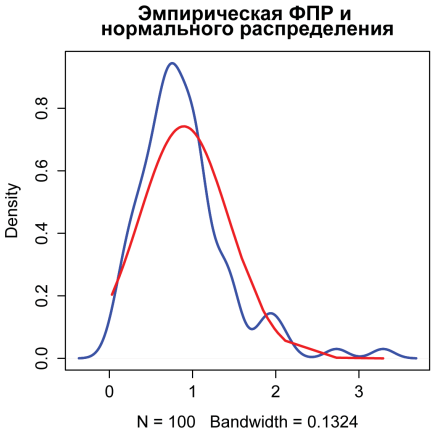
Перед началом процесса подбора теоретического распределения определим функцию вывода двух графиков, на которых мы будем сравнивать функции эмпирического и теоретического распределений – кумулятивную (КФР) и плотности (ФПР):

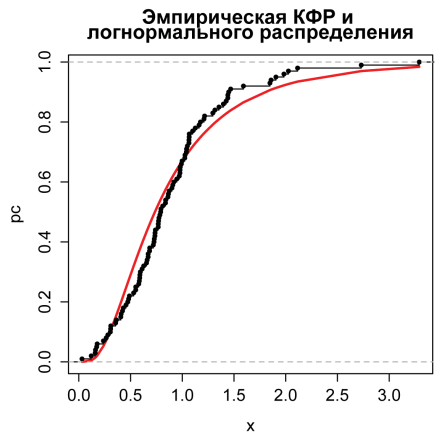
```
graph_distr <- function (x, pc, pd, main_name = "")
{
  op <- par(mfrow = c(1, 1), pty = "s")
  par(mfrow = c(1, 2))
  mn <- paste(c("Эмпирическая КФР и ", main_name))
  plot(x, pc, type = "l", col = "red", lwd = 2, main = mn)
  plot(ecdf(x), add = TRUE)
  mn <- paste(c("Эмпирическая ФПР и ", main_name))
  plot(density(x), lwd = 2, col = "blue", main = mn)
  lines(x, pd, col = "red", lwd = 2)
  par(op)
}
```

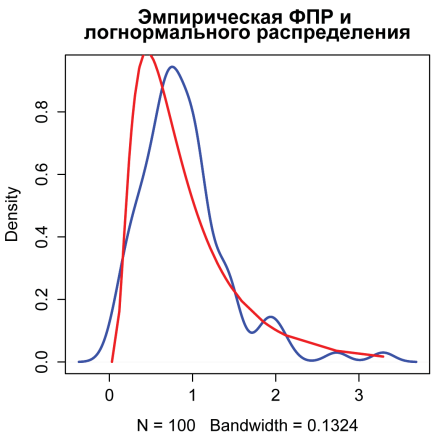
Рассмотрим в качестве моделей-претендентов три закона распределения: нормальное, логнормальное и распределение Вейбулла. Процедура подгонки для нашего эмпирического распределения будет включать три шага:

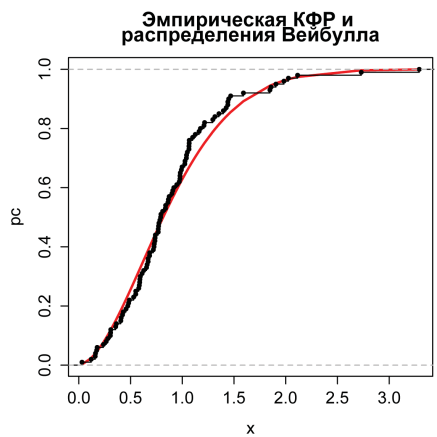
- оценка параметров распределения на основе метода максимального правдоподобия;
- проверка гипотезы о согласии эмпирического и теоретического распределений с использованием критерия Колмогорова-Смирнова;
- вывод графика с использованием определенной выше функции **graph_distr**() (рис. 50).


Эмпирическая КФР и
нормального распределения
pc
x


Эмпирическая ФПР и
нормального распределения
Density
N = 100 Bandwidth = 0.1324


Эмпирическая КФР и
логнормального распределения
pc
x


Эмпирическая ФПР и
логнормального распределения
Density
N = 100 Bandwidth = 0.1324


Эмпирическая КФР и
распределения Вейбулла
pc
x

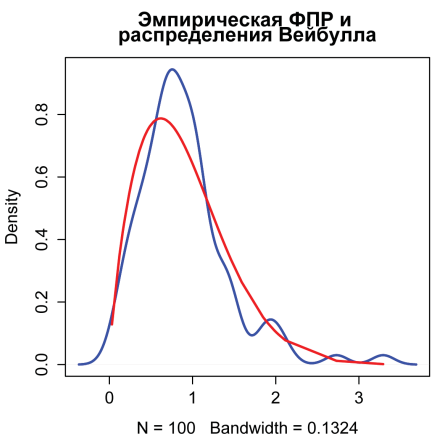

Эмпирическая ФПР и
распределения Вейбулла
Density
N = 100 Bandwidth = 0.1324

Рисунок 50

```
library(MASS)

##  оценка параметров нормального распределения
(dof = fitdistr(x, "normal"))
      mean          sd
  0.89502201   0.53760487
 (0.05376049) (0.03801440)

ep1 = dof$estimate[1]; ep2 = dof$estimate[2]

ks.test(x, pnorm, mean = ep1, sd = ep2)
        One-sample Kolmogorov-Smirnov test
data:  x
D = 0.1342, p-value = 0.05463
alternative hypothesis: two-sided

graph_distr(x, pnorm(x, mean = ep1, sd = ep2),
            dnorm(x, mean = ep1, sd = ep2),
            "нормального распределения")

##  оценка параметров логнормального распределения
(dof = fitdistr(x, "log-normal"))
     meanlog        sdlog
  -0.30750108    0.69803371
 ( 0.06980337) ( 0.04935844)

ep1 = dof$estimate[1]; ep2 = dof$estimate[2]

ks.test(x, plnorm, meanlog = ep1, sdlog = ep2)
        One-sample Kolmogorov-Smirnov test
data:  x
D = 0.1161, p-value = 0.1346
alternative hypothesis: two-sided

graph_distr(x, plnorm(x, meanlog = ep1, sdlog = ep2),
            dlnorm(x, meanlog = ep1, sdlog = ep2),
            "логнормального распределения")

##  оценка параметров распределения Вейбулла
(dof = fitdistr(x, densfun = dweibull,
                start = list(scale = 2, shape = 2)))
ep1 = dof$estimate[1]; ep2 = dof$estimate[2]
1: In dweibull(x, shape, scale, log): созданы NaN
```

Ввиду того, что в ходе оптимизации для распределения Вейбулла при помощи **fitdistr**() были получены неопределенные значения (NaN), для большей уверенности в правильности оценок повторим расчет при помощи аналогичной, но более устойчивой функции из пакета fitdistrplus:

```
library(fitdistrplus)
(dof = fitdist(x, "weibull"))
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters:
      estimate Std. Error
shape 1.745618 0.12932631
scale 1.005806 0.06078022

ep1 = dof$estimate[1]; ep2 = dof$estimate[2]

ks.test(x, pweibull, shape = ep1, scale = ep2)
        One-sample Kolmogorov-Smirnov test
data:  x
D = 0.0897, p-value = 0.3968
alternative hypothesis: two-sided

graph_distr(x, pweibull(x, scale = ep2, shape = ep1),
            dweibull(x, scale = ep2, shape = ep1),
            "распределения Вейбулла")
```

Из приведенных расчетов с использованием теста Колгогорова-Смирнова можно сделать вывод, что все три теоретических распределения статистически значимо согласуются с эмпирическим. В этих условиях и при отсутствии серьезных априорных предположений о законе распределения возникает традиционная (но так однозначно и не решенная) задача статистики: какую модель из некоторого множества претендентов следует предпочесть?

Для выбора оптимального закона распределения из трех возможных можно воспользоваться целым набором мер, таких как средняя абсолютная разность между фактическими и прогнозируемыми значениями, сумма квадратов этих разностей, относительные средние разности, критерий χ^2 или та же *D*-статистика Колмогорова-Смирнова. Ориентируясь, например, на значения последнего критерия, можно полагать нашу выборку распределенной по закону Вейбулла (однако возможно, что другие меры привели бы к какому-то иному заключению).

В общем случае вопрос о том, являются ли полученные оценки параметров наилучшими, обычно остается открытым. Во-первых, соображения, из которых задается вид теоретического распределения, часто оказываются субъективными. Это нередко приводит к тому, что, выполнив вычисления, мы полагаем, что найдены оценки истинных характеристик генерального распределения, тогда как на самом деле получены оценки параметров некого теоретического распределения, которое может быть всего лишь «похожим» на распределение генеральной совокупности, из которого взята выборка. Во-вторых, обычно мы ничего не можем сказать об устойчивости модели и ошибке ее воспроизводимости: нет никакой гарантии, что при взятии повторной выборки расхождение искомых значений параметров не окажется слишком большим.

Проблема подбора наилучшего распределения еще более усложняется при обработке счетных данных («*count data*»), выборочные элементы которых можно рассматривать как подмножество натуральных чисел. К таким выборкам относят-

ся, например, количество отловленных мышей, результаты стрельбы по мишеням, число избирателей, проголосовавших за каждого кандидата, и т. д.

Традиционно считается, что счетные данные следует аппроксимировать дискретными распределениями типа Пуассона, биномиального или геометрического. Однако известно также, что при среднем значении $\lambda > 9$ и достаточно большом объеме выборки распределение Пуассона хорошо аппроксимируется нормальным распределением. Поэтому «спектр» возможных распределений-претендентов для выборок, состоящих из счетных данных, может включать как дискретные, так и непрерывные теоретические распределения.

Рассмотрим следующий пример: из небольшой реки Байтуган было отобрано 60 дночерпательных проб, в каждой из которых подсчитали число обнаруженных видов донных организмов (моллюсков, рачков, червей, личинок). Это число варьировало от 2 до 30 при среднем $\bar{x} = 11.2$. Какое распределение является лучшим с формально-статистической точки зрения для описания этих данных: Пуассона с $\lambda = 11.2$ или нормальное (см. рис. 51)?

```
x <- c(12, 20, 19, 19, 18, 10, 19, 30, 16, 10, 8, 11, 10, 11,
       16, 3, 7, 6, 5, 11, 8, 14, 9, 8, 10, 11, 14, 17, 2, 7,
       17, 19, 9, 15, 9, 8, 4, 8, 11, 8, 5, 3, 10, 14, 22, 11,
       8, 7, 3, 5, 8, 11, 14, 2, 13, 9, 12, 6, 19, 21)

# Оценка параметров распределений нормального и Пуассона
n = length(x); p1 = mean(x); p2 = sqrt(var(x)*(n-1)/n)

# Создание векторов эмпирических и теоретических частот
pr_obs <- as.vector(table(x)/n); nr <- length(pr_obs)
pr_norm <- dnorm(1:nr, p1, p2) # Частоты нормального распр.
pr_pois <- dpois(1:nr, p1)     # Частоты распр. Пуассона

# Графическое изображение подогнанных распределений
plot(pr_obs, type = "b", ylab = "Частоты")
lines(1:nr, pr_pois, col = "red", lwd = 2)
lines(1:nr, pr_norm, col = "blue", lwd = 2)
legend("topright", legend = c("Нормальное", "Пуассона"),
       lwd = 2, col = c("red", "blue"))

# Сравнение качества подгонки распределений:
# Среднее абсолютное отклонение
c(sum(abs(pr_obs-pr_norm))/nr, sum(abs(pr_obs-pr_pois))/nr)
[1] 0.02314994 0.03176255

# Средняя квадратичная ошибка
c(sum((pr_obs-pr_norm)^2)/nr, sum((pr_obs - pr_pois)^2)/nr)
[1] 0.0009595203 0.0017446052

# Критерий согласия Колмогорова-Смирнова
c(ks.test(pr_obs, pr_norm)$statistic,
      ks.test(pr_obs, pr_pois)$statistic)
[1] 0.2272727 0.4090909
```

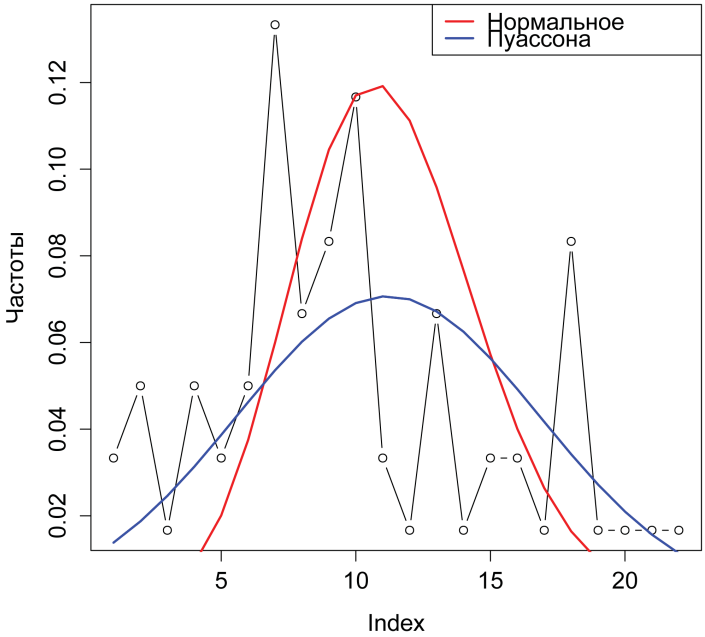


Рисунок 51

На основании всей совокупности выполненных тестов можно заключить, что нормальное распределение обеспечивает наименьшую ошибку аппроксимации данных.

Подгонку счетных данных дискретными распределениями можно выполнить с помощью функции **goodfit**() из пакета vcd. Аргумент type = c("poisson", "binomial", "nbinomial") задает здесь тип предполагаемого распределения (см. также рис. 52):

```
library(vcd)
gf <- goodfit(table(x), type = "poisson", method = "ML")
summary(gf)
         Goodness-of-fit test for poisson distribution
                      X^2 df     P(> X^2)
Likelihood Ratio 75.41796 20 2.320178e-08

#  Визуализация подогнанных данных в виде гистограммы
plot(gf, ylab = "Частота", xlab = "Число классов")
```

4.8. Проверка на нормальность распределения

Далее в главе 6 будет показано, что проверка исследуемых переменных на нормальность распределения является важной составной частью разведочного анализа данных. Существует несколько способов такой проверки, и их можно разделить на две рассмотренные ниже группы.

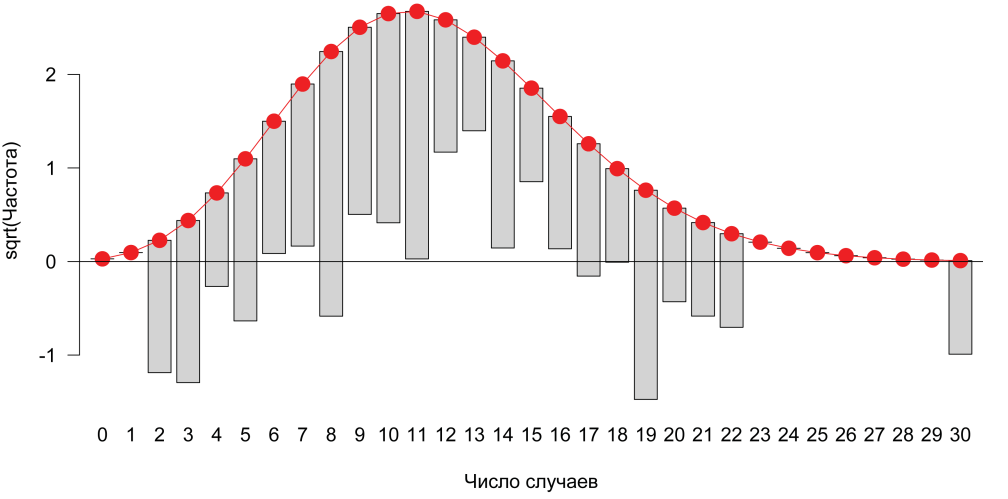


Рисунок 52

Графические способы

Самый простой графический способ проверки характера распределения данных – это построение гистограммы. Если гистограмма имеет колоколообразный симметричный вид, можно сделать заключение о том, что анализируемая переменная имеет примерно нормальное распределение. Однако при интерпретации гистограмм следует соблюдать осторожность, поскольку их внешний вид может сильно зависеть как от числа наблюдений, так и от шага, выбранного для разбиения данных на классы (подробнее см. раздел 3.2). Кроме того, достаточно часто при анализе выборок, извлеченных из *смеси* нормально распределенных совокупностей, гистограммы приобретают асимметричный вид, вводя исследователя в заблуждение.

Например, на гистограмме на рис. 53 (слева) приведены данные по весу 1193 воробьев (Zuur et al., 2010, http://bit.ly/1E0Ub8J), объединенные для июня, июля и августа. Поскольку вес птиц зависит от времени года, гистограмма приобретает асимметричный вид. На графике справа показаны те же данные, но отдельно по каждому месяцу. На основании этого графика А. Цуур и соавторы заключают (возможно, без достаточно серьезных оснований), что на самом деле вес воробьев имеет примерно нормальное распределение.

Другим очень часто используемым графическим способом проверки характера распределения данных является построение так называемых «*графиков квантилей*» («*q-q plots*», «*quantile-quantile plots*»). На таких графиках изображаются квантили двух распределений – эмпирического (то есть построенного по анализируемым данным) и теоретически ожидаемого стандартного нормального распределения. При нормальном распределении проверяемой переменной точки на графике квантилей должны выстраиваться в прямую линию, исходящую под углом 45 градусов из левого нижнего угла графика. Графики квантилей особенно полез-

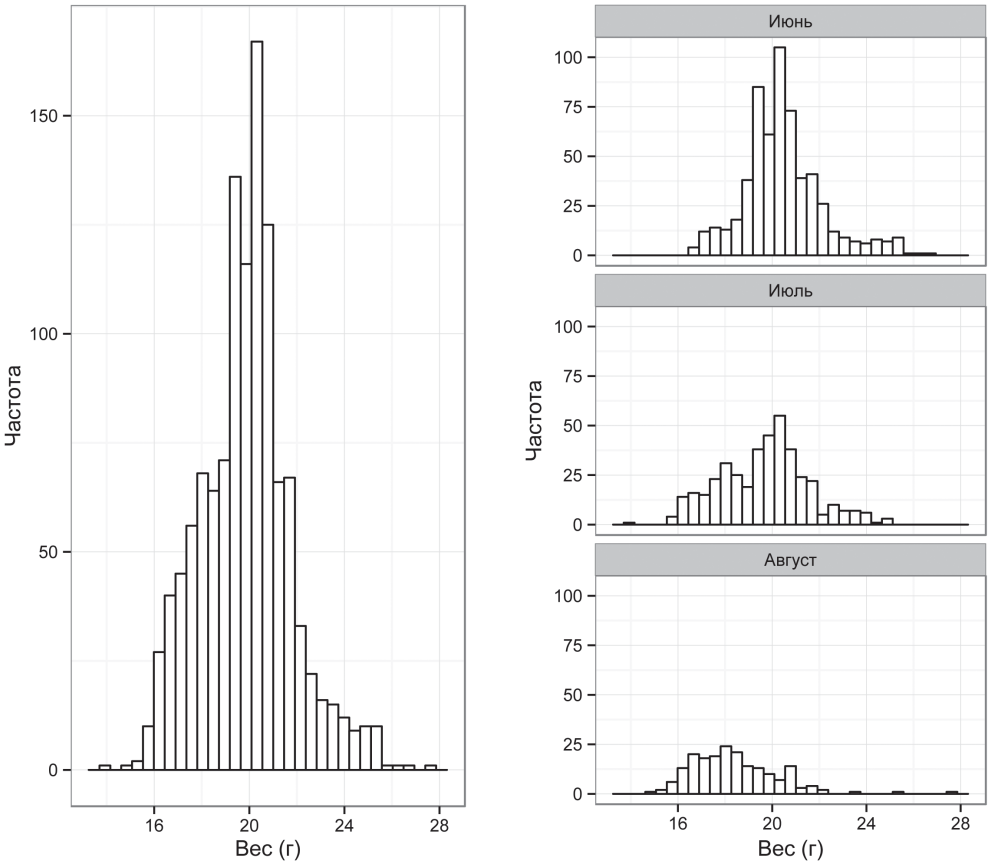


Рисунок 53

ны при работе с небольшими по размеру совокупностями, для которых невозможно построить гистограммы, принимающие какую-либо выраженную форму.

Интерпретация квантильных графиков, получаемых с использованием распространенных функций `qqnorm()` и `qqplot()`, носит в значительной мере субъективный характер. Так, одни исследователи могут полагать, что экспериментальные точки в значительной степени отклоняются от диагонали теоретических квантилей, а другие на том же графике сочтут эти отклонения достаточно приемлемыми для принятия гипотезы о нормальности.

Проблема объективной количественной оценки меры расхождения эмпирической и теоретических кривых является одной из важнейших задач прикладной статистики. Б. Рипли (Ripley, 1978, http://bit.ly/1BZQ7Ih) ввел удобный графический метод для анализа возможных причин расхождения модели и данных – *метод огибающих* («*simulation envelopes*»). В основе алгоритма имитаций лежит, в частности, параметрический бутстреп.

Используем метод огибающих для оценки расхождений между квантилями нормального распределения и стьюдентизированными значениями случайной выборки из смеси распределений Вейбулла (см. раздел 4.7). Для этого извлечем из нормального распределения с заданными параметрами (то есть со средним $m = 0.895$ и стандартным отклонением $s = 0.538$, что соответствует полученным нами ранее выборочным оценкам) 999 случайных выборок по 100 наблюдений каждая. Далее построим на графике пучок из 999 линий, соответствующих имитируемым выборкам. Поскольку каждая из наших нормальных выборок является ограниченной ($n = 100$), соответствующие кривые не будут полностью совпадать с теоретической линией квантилей (к которой они будут приближаться при $n \to \infty$), но их совокупный статистический разброс даст нам объективную оценку области, в которой может быть принята гипотеза о нормальности. Если задаться доверительной вероятностью $\gamma = (1 - \alpha/2)$, то можно найти геометрическое место точек гипотетических огибающих, внутри которых будет находиться γ% имитаций теоретической модели (рис. 54):

```
set.seed(1946)
x = sort(rweibull(100, 2, (1 + 1.21*rbinom(100, 1, 0.05))))
c(mean(x), sd(x))
[1]  0.89502201   0.53760487

# Квантиль-квантильный график без доверительных огибающих:
qqnorm(x); qqline(x)

# Определяем и рисуем огибающие:
z <- (x - mean(x))/sqrt(var(x))  # стандартизация выборки
x.qq <- qqnorm(z, plot.it=FALSE); x.qq <- lapply(x.qq, sort)
plot(x.qq, ylim = c(-2, 5), ylab = "Z-статистики выборки", xlab = "Квантили НР")

# Генерация 999 бутстреп-выборок (то есть случайных выборок из
#  нормального распределения с заданными параметрами)
library(boot)
x.gen <- function(dat, mle) rnorm(length(dat))
x.qqboot <- boot(z, sort, R = 999, sim = "parametric", ran.gen = x.gen)
sapply(1:999, function(i) lines(x.qq$x, x.qqboot$t[i,], type = "l", col = "grey"))
points(x.qq, pch = 20)
lines(c(-3,3), c(-3,3), col = "red", lwd = 2)
```

Если далее использовать функцию **envelope**(), то можно построить доверительные огибающие («*confidence envelopes*»), соответствующие заданной доверительной вероятности. Однако приведенный алгоритм уже заложен в функции **qqPlot**() пакета car, что избавляет нас от необходимости громоздких вычислений (рис. 55):

```
library(car)
qqPlot(x, dist = "norm", col = palette()[1], pch = 19,
     xlab = "Квантили нормального распределения",
     ylab = "Наблюдаемые квантили",
     main = "Сравнение квантилей ЭР и НР")
```

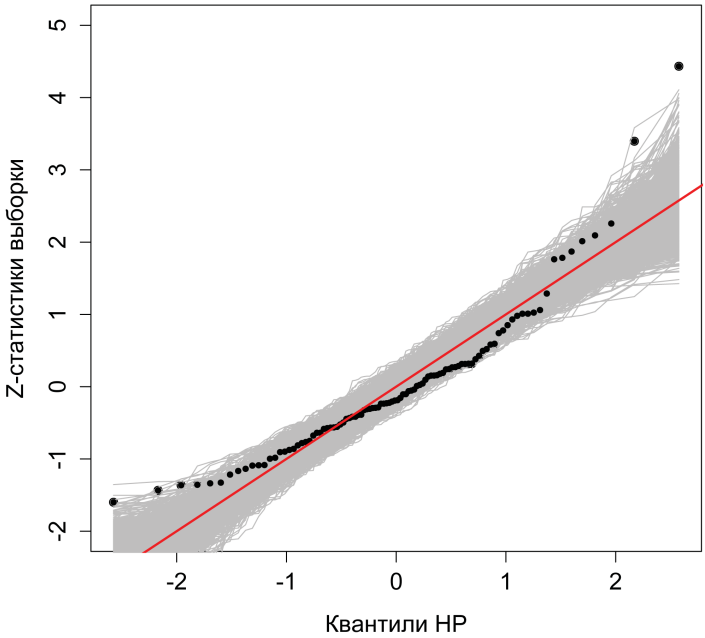


Рисунок 54

На графиках для нашего примера можно обнаружить группы точек, которые выходят за пределы доверительной полосы огибающих с $\gamma = 0.95$, что заставляет сомневаться в согласии эмпирического и нормального распределений.

Прекрасной альтернативой квантильным графикам, построенным при помощи **qqPlot**(), является использование функций **sm.density**() и **sm.density.compare**() из пакета sm. Если, например, указать для функции **sm.density**() параметр model = "Normal", то получим на графике такую же доверительную полосу, но относительно теоретической функции плотности распределения при выборочных значениях параметров (рис. 56). На этом же графике показывается кривая ядерной плотности для эмпирических данных, и можно оценить, насколько правдоподобно предположение о нормальности и в каких диапазонах анализируемой переменной наблюдаются отклонения:

```
library(sm)
sm.density(x, model = "Normal",
           xlab = "Имитированная выборка",
           ylab = "Функция плотности распределения")
```

Формальные тесты

Существует целый ряд статистических тестов, специально разработанных для проверки нормальности распределения данных. В общем виде проверяемую при помощи этих тестов нулевую гипотезу можно сформулировать так: «анализируе-

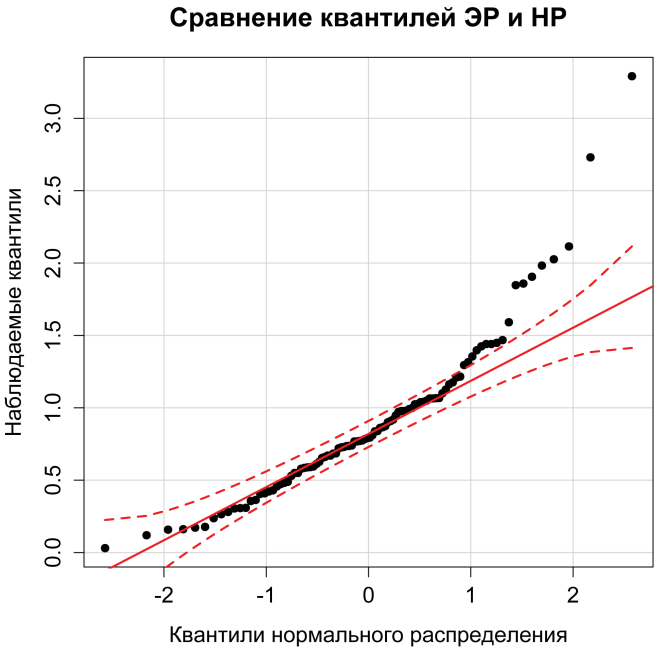


Рисунок 55

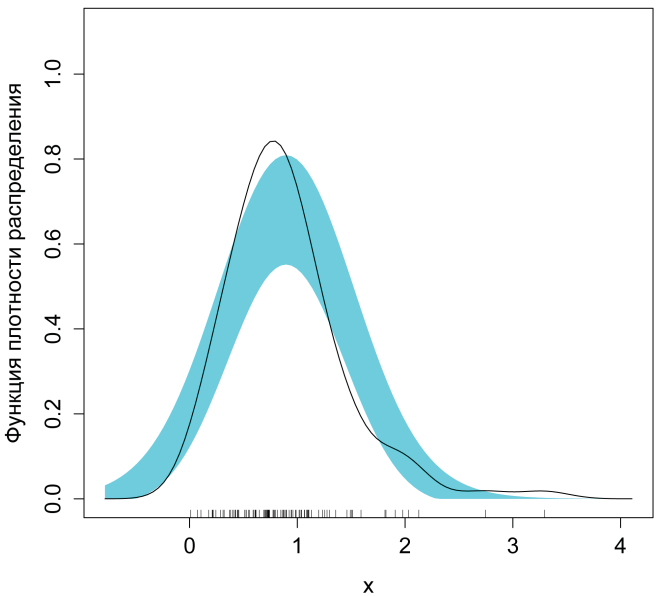


Рисунок 56

мая выборка происходит из генеральной совокупности, имеющей нормальное распределение». Если получаемая при помощи того или иного теста вероятность ошибки p оказывается меньше некоторого заранее принятого уровня значимости (например, 0.05), то нулевая гипотеза отклоняется.

В R реализованы практически все имеющиеся тесты на нормальность – либо в виде стандартных функций, либо в виде функций, входящих в состав подгружаемых пакетов. Примером базовой функции является **shapiro.test**(), при помощи которой можно выполнить широко используемый тест Шапиро-Уилка. Стоит также перечислить функции из пакета nortest, реализующие другие распространенные тесты на нормальность:

- **ad.test**() – тест Андерсона-Дарлинга;
- **cvm.test**() – тест Крамера фон Мизеса;
- **lillie.test**() – тест Колмогорова-Смирнова в модификации Лиллиефорса;
- **sf.test**() – тест Шапиро-Франсия (см. Thode, 2002, http://bit.ly/1ASUr53).

Выполним серию тестов на проверку нормальности все той же случайной выборки из смеси распределений Вейбулла (раздел 4.7) с использованием всех перечисленных критериев:

```
# Тесты на нормальность
shapiro.test(x)
        Shapiro-Wilk normality test
data:  x
W = 0.8986, p-value = 1.219e-06

library(nortest)
ad.test(x)
        Anderson-Darling normality test
data:  x
A = 2.0895, p-value = 2.382e-05

cvm.test(x)
        Cramer-von Mises normality test
data:  x
W = 0.3369, p-value = 0.0001219

lillie.test(x)
        Lilliefors (Kolmogorov-Smirnov) normality test
data:  x
D = 0.1348, p-value = 0.0001225

sf.test(x)
        Shapiro-Francia normality test
data:  x
W = 0.8936, p-value = 3.617e-06
```

Обратим внимание на то, что тест Колмогорова-Смирнова, выполненный в разделе 4.7, не нашел оснований отклонить нулевую гипотезу о нормальном характере распределения анализируемых данных, что свидетельствует о недостаточной мощности этого критерия согласия при малых и средних объемах выборок.

Глава 5

Классические методы статистики

5.1. Гипотеза о равенстве средних двух генеральных совокупностей

Критерий Стьюдента t относится к одним из наиболее давно разработанных и широко используемых статистических критериев. Чаще всего он применяется для проверки нулевой гипотезы о равенстве средних значений двух совокупностей, хотя существует также и одновыборочная модификация этого метода. В этом разделе мы продемонстрируем, как можно реализовать в R статистические тесты, основанные на этом критерии.

Начать, пожалуй, следует с математических допущений, на которых основан критерий Стьюдента. Основных таких допущений, как известно, два:

- сравниваемые выборки должны происходить из нормально распределенных совокупностей;
- дисперсии сравниваемых генеральных совокупностей должны быть равны.

Кроме того, в своей исходной форме *t*-критерий предполагает независимость сравниваемых выборок.

Проверка указанных требований применительно к анализируемым данным должна всегда предшествовать формальному статистическому анализу, в котором задействован критерий Стьюдента (к сожалению, многие исследователи забывают об этом). Методы проверки одного из этих предположений мы рассмотрели в разделе 4.8. Отметим, однако, что условие нормальности распределения данных становится не таким жестким при больших объемах выборок, а для выборок с разными дисперсиями существует особая модификация *t*-критерия (критерий Уэлча; см. ниже). Кроме того, если сам по себе *t*-критерий корректно отражает степень различия выборок, но неверно оценивается его статистическая значимость с использованием таблиц стандартного распределения Стьюдента, то для решения этой проблемы и расчета *p*-значения, не зависящего от законов распределения, с успехом могут использоваться методы рандомизации и бутстрепа (Шитиков, Розенберг, 2014).

Одновыборочный *t*-критерий

Этот вариант критерия Стьюдента служит для проверки нулевой гипотезы о равенстве среднего значения (μ_1) генеральной совокупности, из которой была взята выборка, некоторому априори известному значению (μ_0): $H_0 : \mu_1 = \mu_0$.

В общем виде проверка (или, иными словами, «тестирование») этой гипотезы выполняется при помощи *t*-критерия, который рассчитывается как отношение разницы между выборочным средним и известным значением к стандартной ошибке выборочного среднего:

$$t = \frac{\bar{x} - \mu_0}{S_{\bar{x}}}.$$

Рассчитанное значение критерия мы можем далее интерпретировать следующим образом, исходя из свойств *t*-распределения: если это значение попадает в так называемую *область отклонения* нулевой гипотезы, то мы вправе отвергнуть проверяемую нулевую гипотезу. Область отклонения нулевой гипотезы для критерия Стьюдента определяется заранее принятым уровнем значимости (например, $\alpha = 0.05$) и числом степеней свободы.

Эквивалентным подходом к интерпретации результатов теста будет следующий: если допустить, что нулевая гипотеза верна, то мы можем рассчитать, насколько велика вероятность получить *t*-критерий, равный или превышающий то реальное значение, которое мы нашли по имеющимся выборочным данным. Если эта вероятность оказывается меньше, чем заранее принятый уровень значимости (например, $p < 0.05$), мы вправе отклонить проверяемую нулевую гипотезу. Именно такой подход сегодня используется чаще всего: исследователи приводят в своих работах *p*-значение (то есть достигнутый уровень значимости), которое легко рассчитывается при помощи статистических программ. Рассмотрим, как это можно сделать в системе R.

Предположим, у нас имеются данные по суточному потреблению энергии, поступающей с пищей (кДж/сутки), для 11 женщин (Altman, 1981, http://bit.ly/1MhGYKF):

```
d.intake <- c(5260, 5470, 5640, 6180, 6390, 6515,
              6805, 7515, 7515, 8230, 8770)
```

Среднее значение для этих 11 наблюдений составляет:

```
mean(d.intake)
[1] 6753.6
```

Зададимся вопросом: отличается ли это выборочное среднее значение от установленной нормы в 7725 кДж/сутки? Разница между нашим выборочным значением и этим нормативом довольно прилична: 7725 – 6753.6 = 971.4. Но насколько эта разница статистически значима с учетом уровня вариации приведенных выше 11 значений? Ответить на этот вопрос поможет одновыборочный *t*-тест. Как и другие варианты *t*-теста, одновыборочный тест Стьюдента выполняется в R при помощи функции **t.test**():

```
t.test(d.intake, mu = 7725)
        One Sample t-test
data:  d.intake
```

```
t = -2.8208, df = 10, p-value = 0.01814
alternative hypothesis: true mean is not equal to 7725
95 percent confidence interval:
 5986.348 7520.925
sample estimates:
mean of x
 6753.636
```

Видим, что для имеющихся выборочных данных *t*-критерий составляет –2.821 при 10 степенях свободы (`df`). Вероятность получить такое (либо большее) значение *t* при условии, что проверяемая нулевая гипотеза верна, оказалась мала: `p-value = 0.01814` (во всяком случае, это меньше 5%). Следовательно, по правилу, представленному выше, мы можем отклонить проверяемую нулевую гипотезу о равенстве выборочного среднего значения нормативу и принять альтернативную гипотезу (`alternative hypothesis: true mean is not equal to 7725`). Принимая это предположение, мы рискуем ошибиться с вероятностью менее 5%.

Помимо *t*-критерия, числа степеней свободы, *p*-значения и выборочного среднего (`sample estimates: mean of x`), программа рассчитала также 95%-ный доверительный интервал (`95 percent confidence interval`) для истинной разницы между выборочным средним значением суточного потребления энергии и нормативом. Если бы мы повторили аналогичный тест много раз для разных групп из 11 женщин, то в 95% случаев эта разница оказалась бы в диапазоне от 5986.3 до 7520.9 кДж/сутки.

Сравнение двух независимых выборок

При сравнении двух выборок проверяемая нулевая гипотеза состоит в том, что обе эти выборки происходят из нормально распределенных генеральных совокупностей с одинаковыми средними значениями, то есть H_0: $\mu_1 = \mu_2$. Поскольку эти генеральные средние мы оцениваем при помощи выборочных средних значений, формула *t*-критерия приобретает вид:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{S_{\bar{x}_1 - \bar{x}_2}},$$

где $\bar{x}_1$ и $\bar{x}_2$ – средние значения сравниваемых выборок, а $S_{\bar{x}_1 - \bar{x}_2}$ – стандартная ошибка разности средних, формула для расчета которой определяется тем, одинаковы ли размеры сравниваемых выборок (продробнее см. http://bit.ly/1ATaQX8).

Поскольку на практике зачастую нет оснований предполагать, что выборки происходят из генеральных совокупностей с одинаковыми дисперсиями, обычно используют критерий Стьюдента в *модификации Уэлча* («*Welch's t-test*»):

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}.$$

Если H_0 верна, то значение критерия является случайным числом из t-распределения с $(n_1 + n_2 - 2)$ степенями свободы, и ее интерпретация выполняется точно так же, как и в случае с одной выборкой (см. выше). При использовании t-критерия в модификации Уэлча число степеней свободы рассчитывается по более сложной формуле (см. http://bit.ly/1ATaQX8).

Рассмотрим пример о суточном расходе энергии (`expend`) у худощавых женщин (`lean`) и женщин с избыточным весом (`obese`), приведенный в книге П. Дальгаарда (Dalgaard, 2008). Данные из этого примера (подробнее см. `?energy`) входят в состав пакета `ISwR`, сопровождающего указанную книгу:

```
library(ISwR)
data(energy)
attach(energy)
head(energy)
   expend stature
1    9.21   obese
2    7.53    lean
3    7.48    lean
4    8.08    lean
5    8.09    lean
6   10.15    lean
```

Соответствующие средние значения потребления энергии в рассматриваемых группах пациенток можно найти с использованием знакомой нам функции **`tapply`**`()`:

```
tapply(expend, stature, mean)
lean obese
8.07 10.30
```

Вопрос заключается в том, различаются ли эти средние значения статистически. Проверим гипотезу об отсутствии разницы при помощи t-теста:

```
t.test(expend ~ stature)
        Welch Two Sample t-test
data:  expend by stature
t = -3.8555, df = 15.919, p-value = 0.001411
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.459167 -1.004081
sample estimates:
 mean in group lean mean in group obese
           8.066154           10.297778
```

Обратите внимание на использование знака `~` в вызове функции **`t.test`**`()`. Это стандартный для R способ записи формул, описывающих связь между переменными. В нашем случае выражение `expend ~ stature` можно расшифровать как «зависимость суточного потребления энергии (`expend`) от статуса пациентки (`stature`)».

Согласно величине полученного *p*-значения (`p-value = 0.001411`), средний уровень потребления энергии у женщин из рассматриваемых весовых групп статистически значимо различается. Отвергая нулевую гипотезу о равенстве этих средних значений, мы рискуем ошибиться с вероятностью лишь около 0.15%. При этом истинная разница между средними значениями с вероятностью 95% находится в диапазоне от –3.5 до –1.0 (см. `95 percent confidence interval`).

Следует подчеркнуть, что при выполнении двухвыборочного *t*-теста функция R по умолчанию принимает, что дисперсии сравниваемых совокупностей *не равны*, и, как следствие, выполняет *t*-тест в модификации Уэлча. Мы можем изменить такое поведение программы, воспользовавшись аргументом `var.equal = TRUE`: (от «*var*iance» – дисперсия и «*equal*» – равный):

```
t.test(expend ~ stature, var.equal = TRUE)
        Two Sample t-test
data:  expend by stature
t = -3.9456, df = 20, p-value = 0.000799
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.411451 -1.051796
sample estimates:
 mean in group lean mean in group obese
          8.066154           10.297778
```

P-значение стало еще меньше, и мы так же, как и после теста в модификации Уэлча, можем сделать вывод о наличии существенной разницы групповых средних. Однако такое совпадение выводов будет иметь место не всегда, и, следовательно, на разницу между групповыми дисперсиями (или ее отсутствие) следует обращать серьезное внимание при выборе и интерпретации того или иного варианта *t*-теста.

Сравнение двух зависимых выборок

Зависимыми, или *парными*, являются две выборки, содержащие результаты измерений какого-либо количественного признака, выполненных на одних и тех же объектах. Во многих исследованиях определенный отклик измеряется у одних и тех же объектов *до и после* экспериментального воздействия. При такой схеме эксперимента исследователь более точно оценивает эффект воздействия именно потому, что прослеживает его фактически у каждого уникального объекта.

Нас интересуют свойства выборки, составленной из разностей значений признака у одних и тех же объектов, а точнее – «истинная средняя разность» как результат экспериментального воздействия (обозначим его δ). Если верна нулевая гипотеза H_0: δ = 0, утверждающая, что средняя разность δ между парами реализаций случайных величин статистически значимо не отличается от нуля, то нет оснований предполагать, что эффект воздействия имеет место. Тогда формула для *парного критерия Стьюдента* примет вид:

$$t = \frac{\overline{d}}{s\sqrt{n}};\quad s = \sqrt{\frac{1}{n-1}\sum_{1}^{n}(d_i - \overline{d})^2};\quad d_i = x_{2i} - x_{1i}.$$

В книге П. Дальгаарда (Dalgaard, 2008) имеется пример о суточном потреблении энергии, измеренном уже у *одних и тех же* 11 женщин до и после периода менструаций:

```
data(intake) # из пакета ISwR
attach(intake)
head(intake, n = 5)
    pre post
1  5260 3910
2  5470 4220
3  5640 3885
4  6180 5160
5  6390 5645
```

Индивидуальные разности потребления энергии у этих женщин составляют:

```
post - pre
[1] -1350 -1250 -1755 -1020  -745 -1835 -1540 -1540
[9]  -725 -1330 -1435
```

Усреднив эти индивидуальные разницы, получим:

```
mean(post - pre)
[1] -1320.5
```

Задача заключается в том, чтобы оценить, насколько статистически значимо эта средняя разность отличается от нуля. Применим парный критерий Стьюдента (обратите внимание на использование аргумента `paired = TRUE`):

```
t.test(pre, post, paired = TRUE)
        Paired t-test
data:  pre and post
t = 11.9414, df = 10, p-value = 3.059e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1074.072 1566.838
sample estimates:
mean of the differences
               1320.455
```

Как видим, рассчитанное программой *p*-значение оказалось намного меньше 0.05, что позволяет нам сделать заключение о наличии существенной разницы в потреблении энергии у исследованных женщин до и после менструации. 95%-ный доверительный интервал для истинной величины эффекта (в абсолютном выражении) ограничивается значениями 1074.1 и 1566.8 кДж/сутки.

Приведенные выше примеры охватывают наиболее типичные случаи применения критерия Стьюдента. За рамками этого раздела остаются так называемые *односторонние* варианты *t*-теста, когда проверяемая нулевая гипотеза заключается в том, что одно из сравниваемых средних значений больше (или меньше) другого. Однако можно отметить, что односторонний вариант *t*-теста легко реа-

лизуется при помощи функции **t.test**() в сочетании с аргументом alternative, который может принимать одно из трех значений – «two.sided» («двухсторонний», выбирается программой по умолчанию), «greater» («больше») или «less» («меньше»).

5.2. Ранговый критерий Уилкоксона-Манна-Уитни

Как описано выше, одно из важных условий корректного применения критерия Стьюдента состоит в том, что анализируемые выборки происходят из нормально распределенных генеральных совокупностей. В случаях, когда это условие не выполняется, вместо критерия Стьюдента следует использовать его непараметрический аналог – *критерий Уилкоксона* (*Wilcoxon rank test*). Здесь необходимо пояснить, что создатели системы R под названием «критерий Уилкоксона» (или «тест Уилкоксона») объединяют как метод, предложенный собственно Ф. Уилкоксоном (Frank Wilcoxon) в 1945 г., так и опубликованный несколько позднее (1947 г.) метод Г. Манна (Herald Mann) и Д. Уитни (Donald Whitney). Первый из этих методов обычно используется для сравнения двух парных выборок, тогда как второй предназначен для сравнения двух независимых выборок. Ниже мы не будем разграничивать эти методы, придерживаясь подхода, принятого в системе R.

Одновыборочный критерий Уилкоксона

Этот вариант критерия («*Wilcoxon signed rank test*») служит для проверки нулевой гипотезы о том, что анализируемая выборка происходит из симметрично распределенной генеральной совокупности с центром в точке μ_0. Алгоритм метода заключается в следующем:

- μ_0 вычитают от каждого выборочного значения;
- получившиеся величины ранжируют по возрастанию, игнорируя знак;
- ранговые номера со знаком + суммируют, получая величину V;
- критерий V сравнивают с критическим значением для заданного уровня значимости и числа степеней свободы.

Альтернативный вариант интерпретации результатов теста заключается в нахождении условной вероятности случайным образом получить значение критерия, равное или превышающее эмпирическую величину V, при условии истинности нулевой гипотезы.

Ниже при описании теста Уилкоксона мы будем использовать примеры, рассмотренные в предыдущем разделе для иллюстрации *t*-теста Стьюдента. Это позволит нам сравнить результаты, получаемые при помощи обоих методов. В частности, обратимся к данным о суточном потреблении энергии у 11 женщин и выясним, имеются ли отличия от нормативного значения 7725 кДж/сутки:

```
d.intake <- c(5260, 5470, 5640, 6180, 6390, 6515,
              6805, 7515, 7515, 8230, 8770)
```

Для выполнения теста Уилкоксона в системе R используется функция **wilcox.test**():

```
wilcox.test(d.intake, mu = 7725)
        Wilcoxon signed rank test with continuity correction
data:  d.intake
V = 8, p-value = 0.0293
alternative hypothesis: true location is not equal to 7725
Warning message:
In wilcox.test.default(d.intake, mu = 7725) :
  cannot compute exact p-value with ties
```

Видим, что рассчитанное значение критерия V = 8. Вероятность случайно получить такое (или превышающее его) значение при условии, что нулевая гипотеза верна, не превышает 0.05 (p-value = 0.0293). Это позволяет нам отклонить нулевую гипотезу о том, что суточное потребление энергии у обследованных 11 женщин не отличается от принятой нормы. Обратите внимание на выданное программой предупреждение о том, что полученное значение вероятности *p* не является точным из-за наличия в данных значений с одинаковыми рангами (Warning message... cannot compute exact p-value with ties). Проблема расчета точных *p*-значений при наличии повторяющихся значений в данных характерна для статистических методов, основанных на рангах, и критерий Уилкоксона не является исключением. При наличии повторяющихся наблюдений *p*-значение рассчитывается путем аппроксимации распределения критерия Уилкоксона нормальным распределением (см. справочный файл по функции ?wilcox.test).

Сравнение двух независимых выборок

Если сравниваемые выборки являются независимыми (аргумент paired = FALSE), то мы имеем дело с критерием Уилкоксона, который в англоязычной литературе называют «*Wilcoxon rank sum test*» (как было отмечено выше, этот метод называют также методом Манна-Уитни). Проверяемая с его помощью нулевая гипотеза состоит в том, что центры распределений, из которых происходят сравниваемые выборки, смещены относительно друг друга на величину μ (например, μ = 0). Алгоритм метода состоит в следующем:

- все имеющиеся значения ранжируют, игнорируя их знак;
- ранги значений, принадлежащих к первой группе, суммируют, получая величину W;
- W сравнивают со значением, которое можно было бы ожидать при верной нулевой гипотезе и имеющемся числе степеней свободы.

Альтернативный подход – расчет вероятности случайным образом получить значение W, равное или превышающее наблюдаемое значение (при условии истинности нулевой гипотезы).

Используем рассмотренный ранее пример о суточном расходе энергии (expend) у худощавых женщин (lean) и женщин с избыточным весом (obese):

```
data(energy) # из пакета ISwR
attach(energy)
str(energy)
'data.frame': 22 obs. of  2 variables:
 $ expend : num  9.21 7.53 7.48 8.08 8.09 ...
 $ stature: Factor w/ 2 levels "lean","obese": 2 1 1 1 1 1 1 1 1 1 ...
```

Проверим гипотезу об отсутствии разницы в потреблении энергии у женщин из этих двух групп при помощи критерия Уилкоксона для независимых выборок:

```
wilcox.test(expend ~ stature, paired = FALSE)
Wilcoxon rank sum test with continuity correction
data:  expend by stature
W = 12, p-value = 0.002122
alternative hypothesis: true location shift is not equal to 0
Warning message:
In wilcox.test.default(x = c(7.53, 7.48, 8.08, 8.09, 10.15, 8.4,  :
  cannot compute exact p-value with ties
```

Согласно полученному *p*-значению (`p-value = 0.002122`), потребление энергии у женщин из рассматриваемых весовых групп статистически значимо различается. Отвергая нулевую гипотезу о равенстве потребляемой энергии, мы рискуем ошибиться с вероятностью лишь около 0.2%.

Сравнение двух зависимых выборок

Понятие «зависимые», или «связные», выборки обсуждалось ранее в разделе, посвященном критерию Стьюдента. Сейчас для нас более важен тот факт, что обе сравниваемые выборки происходят из ненормально распределенных генеральных совокупностей. Это дает нам весомые основания выполнить сравнение при помощи парного рангового критерия Уилкоксона.

Как и в парном тесте Стьюдента, находят разницу между всеми имеющимися парными выборочными наблюдениями с целью проверить нулевую гипотезу о том, что медиана полученных разностей равна нулю (либо какому-то другому, отличному от нуля значению). Здесь (псевдо)медианой распределения F называют медиану распределения $(u + v)/2$, где u и v являются независимыми переменными, каждая из которых имеет распределение F. Если распределение F симметрично, псевдомедиана и медиана совпадают (подробнее см. `?wilcox.test`).

Используем рассмотренный ранее пример о суточном потреблении энергии, измеренном у *одних и тех же* 11 женщин до и после периода менструаций:

```
data(intake) # из пакета ISwR
attach(intake)
```

Сравнить два периода по потреблению энергии при помощи критерия Уилкоксона можно следующим образом (обратите внимание на использование аргумента `paired = TRUE`):

```
wilcox.test(pre, post, paired = TRUE)
Wilcoxon signed rank test with continuity correction
data:  pre and post
V = 66, p-value = 0.00384
alternative hypothesis: true location shift is not equal to 0
Warning message:
In wilcox.test.default(pre, post, paired = T) :
  cannot compute exact p-value with ties
```

Рассчитанное программой *p*-значение оказалось меньше 0.05, что позволяет нам сделать заключение о наличии статистически значимой разницы в потреблении энергии у исследованных женщин до и после менструации. (Для сравнения: *p*-значение, полученное при помощи критерия Стьюдента, было << 0.001). Мы можем оценить доверительный интервал истинной величины эффекта, воспользовавшись аргументом `conf.int` (доверительная вероятность задается при помощи аргумента `conf.level`; по умолчанию рассчитывается 95%-ный доверительный интервал):

```
wilcox.test(pre, post, paired = TRUE, conf.int = TRUE)
Wilcoxon signed rank test with continuity correction
data:  pre and post
V = 66, p-value = 0.00384
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 1037.5 1582.5
sample estimates:
(pseudo)median
      1341.332
Warning messages:
1: In wilcox.test.default(pre, post, paired = TRUE, conf.int = TRUE):
  cannot compute exact p-value with ties
2: In wilcox.test.default(pre, post, paired = TRUE, conf.int = TRUE):
  cannot compute exact confidence interval with ties
```

Видим, что при многократном повторении этого эксперимента разность уровней потребленной энергии в 95% случаев находилась бы в интервале от 1037.5 до 1581.5 кДж/сутки. Опять-таки, из-за наличия повторяющихся наблюдений расчет точных доверительных пределов оказался невозможным (см. пункт 2 в списке предупреждений `Warning messages`). Псевдомедиана (`(pseudo)median`) индивидуальных разностей между парными значениями потребления энергии была оценена в 1341.3 кДж/сутки.

Важно отметить одно из ограничений критерия Уилкоксона для двух выборок (зависимых или независимых): если общее количество наблюдений не превышает 6, то обнаружить разницу между выборками с уровнем ошибки в 5% просто невозможно (Dalgaard, 2008).

5.3. Рандомизация, бутстреп и оценка статистической мощности (на примере двухвыборочного *t*-критерия)

Как было показано в предыдущих разделах этой главы, обычно процесс проверки статистической гипотезы включает следующие шаги:

1. Формулировка проверяемой нулевой гипотезы. Например, в случае двухвыборочного критерия Стьюдента она состоит в том, что обе выборки происходят из нормально распределенной генеральной совокупности с некоторым средним значением.
2. Выбор подходящего статистического критерия для проверки нулевой гипотезы и вычисление значения этого критерия по имеющимся выборочным данным.
3. Определение критического значения критерия, исходя из желаемого уровня статистической значимости α и свойств теоретического распределения этого критерия.
4. Проверка того, превышает ли рассчитанный по выборочным данным критерий некоторое заданное критическое значение. Если такое превышение *не наблюдается*, делают вывод о том, что нулевая гипотеза *верна*.

Современные компьютерно-интенсивные технологии позволяют обобщить и углубить смысл этой процедуры. В законченном варианте проблема заключается в том, чтобы получить *три* эмпирические функции распределения статистического критерия: *а*) при условии справедливости нулевой гипотезы H_0; *б*) для имеющихся эмпирических данных, и, наконец, *в*) при условии справедливости альтернативной гипотезы H_1.

Получить эмпирическое распределение тестового критерия в условиях справедливости нулевой гипотезы – наиболее простая задача, которая решается с использованием рандомизации. Для этого мы объединяем обе сравниваемые выборки, случайно перемешиваем наблюдения между собой (другие термины – «перестановка» или «пермутация», от англ. «*permutation*»), снова создаем две группы и вычисляем значение нашего критерия. Эту процедуру можно повторить многократно и получить частотное распределение критерия $p_0(t^*)$ – например, распределение разности групповых средних $t = \bar{x}_1 - \bar{x}_2$.

Это распределение $p_0(t^*)$ будет иметь место *при справедливости нулевой гипотезы* H_0: $\mu_1 = \mu_2$, поскольку все возможные комбинации элементов в сравниваемых группах будут равновероятны. Тогда если величина t_{obs} для экспериментальных данных является типичным значением из распределения $p_0(t^*)$, то нет оснований отклонять H_0. Если же это не так и t_{obs} оказывается в критической области критерия, то нулевая гипотеза отвергается на некотором уровне значимости и (косвенно) альтернативная гипотеза считается более вероятной.

Здесь уровень значимости для t_{obs} – процент от значений из распределения $p_0(t^*)$, которые равны или превышают t_{obs}. Этот процент можно интерпретировать таким

же образом, как и для обычных критериев значимости: если число превышений t_{obs} менее 5%, то это является достаточно весомым свидетельством в пользу того, что нулевая гипотеза не верна, а если процент превышений оказывается меньше 1%, то значимость отклонения H_0 становится еще более убедительной.

Получить эмпирическое распределение тестового критерия T_{obs} *для имеющихся эмпирических данных* – также довольно простая задача, которую можно решить с использованием описанного ранее бутстреп-метода (см. раздел 2.7). Более того, мы легко можем оценить доверительный интервал для t_{obs} при заданном уровне надежности, и если он не включает 0, то нулевая гипотеза отклоняется.

Покажем, как это можно выполнить в R на том же примере (`expend`) о суточном расходе энергии у худощавых и упитанных женщин (результат представлен на рис. 57):

```
# Функция, выполняющая пермутацию t-критерия заданное число раз
simP <- function(x, group, permutations = 5000) {
     n <- length(x)
     replicate(permutations, {
          xr <- sample(x, n)
          t.test(xr ~ group)$statistic
  })
}
#  Функция, выполняющая бутстреп t-критерия заданное число раз
bootP <- function(x, group, boots=5000) {
     n <- length(x)
     replicate(boots, {
      ind <- sample.int(n, n, replace=T)
      t.test(x[ind] ~ group[ind])$statistic
  })
}

#  Выполнение расчетов:
data(energy, package = "ISwR")
attach(energy)
t.emp <- t.test(expend ~ stature)$statistic
t.perm <- simP(expend, stature, 1000)
t.boot <- bootP(expend, stature, 1000)

mm <- range(c(t.emp, t.perm, t.boot))
mm # размах значений t:
[1] -17.691141   4.351739

# псевдо-P:
p <- (sum(abs(t.perm)- abs(t.emp) >= 0)+1) / (1000 + 1)
 [1] 0.002997003

# Доверительный интервал:
CI <- quantile(t.boot, probs = c(0.025, 0.975))
```

```
     2.5%      97.5%
-7.102815 -2.126967

plot(density(t.boot), col = "blue", lwd = 2, xlim = c(-10,5),
     ylim = c(0,0.4), main = "",
     xlab = "", ylab = "Плотность вероятности")
#  Заливаем полигон доверительной области:
   x1 <- min(which(dens$x >= CI[2]))
   x2 <- max(which(dens$x <  CI[1]))
   with(dens, polygon(x = c(x[c(x1, x1:x2, x2)]),
          y = c(0, y[x1:x2], 0), col = "gray"))
abline(v = t.emp, col = "green", lwd = 2 )
lines(density(t.perm), col = "red", lwd = 2)
lines(density(t.boot-5), col = "black", lwd = 2)
text(-8, 0.36, "Область H1")
text(-3.1, 0.36, "t(obs)")
text(2, 0.36, "Область H0")
```

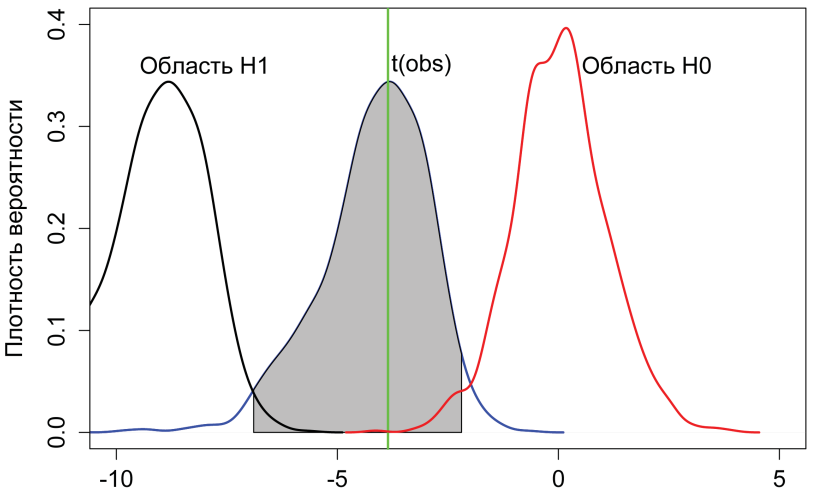


Рисунок 57

Функция распределения $p_0(t^*)$, полученная при выполнении рандомизации (на рис. 57 показана красным цветом), как и положено, имеет среднее значение, близкое к 0. Из 1000 итераций только в двух случаях t-критерий превысил эмпирическое значение t_{obs} (сравнение проводилось по абсолютной величине, поскольку проверялась двухсторонняя гипотеза). Следовательно, риск α совершить ошибку первого рода, то есть *отклонить верную* нулевую гипотезу, равен достигнутому уровню значимости: $P = (2 + 1) / (1000 + 1) = 0.003$, где 1 добавлена для учета исходной выборки. Эта оценка несколько выше значения, рассчитанного ранее с использованием рангового критерия ($p = 0.0021$), но меньше значения, рассчитанного параметрическим методом ($p = 0.014$).

Несколько шире, по сравнению с параметрическими оценками, оказался 95%-ный доверительный интервал для t_{obs}, найденный бутстреп-методом (область, залитая на рис. 57 серым цветом). Обратим также внимание на его несимметричность, связанную с асимметрией кривой эмпирического распределения (показана синим цветом).

Значительно более сложной задачей является нахождение по эмпирическим данным функции $p_1(t^*)$ распределения t-критерия *при справедливой альтернативной гипотезе* H_1. Такая гипотетическая кривая для нашего примера должна была пройти на представленном графике где-то левее остальных кривых (показана черным цветом), а сегмент площади под ней, ограниченный слева от t_{obs}, определит риск β ошибки второго рода, то есть *принятие неверной* нулевой гипотезы.

Число $\chi = (1 - \beta)$, равное вероятности отклонения неверной нулевой гипотезы H_0, называется *статистической мощностью* критерия. Эта вероятность соответствует площади фигуры $\chi = \int_{x=t_{obs}}^{\infty} p_1(t)dt$, образованной графиком функции $p_1(t^*)$ и лежащей справа от точки t_{obs}. Мощность критерия χ («*statistical power*») является одной из количественных оценок статистической *мощности исследования*, отражающей *чувствительность* диагностического теста, то есть вероятность того, что в эксперименте будет найден статистически значимый эффект исследуемого фактора, когда этот эффект действительно существует.

Можно показать, что при фиксированном объеме выборки n любая перестройка статистического критерия в направлении уменьшения уровня значимости α связана с одновременным уменьшением значений функции мощности χ, так как при этом расширяется область отклонения альтернативы. И наоборот: любые попытки увеличения мощности критерия при одних и тех же n неизбежно приводят к одновременному увеличению его уровня значимости (Айвазян и совт., 1983, http://bit.ly/1BHe0kJ).

При увеличении объема выборки n мощность теста становится выше, и при достаточно больших выборках даже небольшие различия окажутся статистически значимыми. Соответственно, при малых выборках даже большие различия выявить становится весьма трудно. Казалось бы, здесь налицо «тупиковая» ситуация: при малых выборках вывод статистически ненадежен, а при слишком больших – однозначно предопределен и неизменно приводит к отклонению проверяемой гипотезы.

Чтобы избежать эффектов «слишком малой» и «слишком большой» выборок, априорный выбор значений обеих характеристик точности критерия (то есть уровня значимости α и ошибки второго рода β) необходимо увязывать с объемом имеющихся данных. При малых n, прежде чем выполнять тест на наличие различий, исследователь должен оценить, будет ли мощность планируемого к применению статистического критерия в данном эксперименте достаточной для их обнаружения. В случае больших выборок полезно смоделировать динамику уменьшения достигнутого уровня значимости α в зависимости от n и найти порог обязательного отклонения основной гипотезы.

Анализ статистической мощности («*statistical power analysis*») позволяет исследовать взаимосвязь между четырьмя компонентами: *а*) величиной ожидаемого эффекта δ; *б*) числом выполняемых измерений *n*; *в*) величиной уровня значимости α; *г*) мощностью критерия. Математически взаимосвязь между уровнем значимости, β-риском и величиной эффекта в достаточно широком классе случаев может быть представлена следующей приближенной формулой:

$$n(\alpha;\beta;\delta) \approx \frac{(u_{1-\alpha} + u_{1-\beta})^2}{\delta(H_0, H_1)},$$

где u_q – квантиль уровня q стандартного нормального распределения; $\delta(H_0, H_1)$ – величина эффекта или «расстояние» между гипотезами H_0 и H_1, которое находится на основе распределений $p_0(t^*)$ и $p_1(t^*)$ (см. рис. 57). Если закон распределения критерия неизвестен, то для нахождения необходимого общего объема двух выборок можно воспользоваться специальными номограммами, встречающимися во многих учебниках по статистике (см., например, Реброва, 2002, http://bit.ly/1ECieh8), или, лучше, использовать соответствующие возможности R.

В нашем случае с двумя выборками *величина эффекта* δ, или *минимальная стандартизованная разница*, определяется как отношение *d*/*s*, где *d* – абсолютное значение средней разности между сравниваемыми группами, а *s* – известное априори стандартное отклонение изучаемого показателя у данной категории экспериментальных единиц. На практике обычно (но не всегда) приемлемой считается мощность теста χ, равная или превышающая 80% (что соответствует β-риску в 20%). Этот уровень является следствием так называемого «*отношения 1 к 4*» («*one-to-four trade-off*») между уровнями α-риска и β-риска: если принять уровень значимости α = 0.05, тогда β = 0.05×4 = 0.20, и мощность критерия составит χ = 1 – 0.20 = 0.80.

В качестве примера предположим, что мы планируем проведение эксперимента для установления эффекта влияния температуры воды на индивидуальный вес особей некоторого вида водных жуков. В эксперименте будут задействованы два температурных режима, и, соответственно, наличие эффекта воздействия температуры мы будем проверять, сравнивая средние значения веса жуков из двух экспериментальных групп при помощи двухстороннего критерия Стьюдента. Двухсторонний вариант выбран потому, что до проведения эксперимента мы не знаем, какой именно эффект вызовет повышение температуры – повышение или понижение веса жуков. Проверяемая в этом случае нулевая гипотеза состоит в том, что температура не оказывает никакого влияния на вес жуков.

Допустим, что минимальная разница в среднем весе жуков, которую мы хотим выявить в ходе эксперимента, составляет 3 мг. Если принять уровень значимости α = 0.05, желаемая мощность теста из соотношения 1:4 будет составлять χ = 80%. Вопрос заключается в том, сколько особей мы должны задействовать в эксперименте для того, чтобы перечисленные условия были выполнены.

Как следует из приведенного выше списка параметров анализа мощности, для определения оптимального размера выборки нам необходимо знать стандартное

отклонение веса изучаемого вида жуков. К сожалению, до проведения эксперимента мы не можем оценить этот параметр. Вариантов решения этой проблемы два: *а*) основываясь на своем экспертном заключении, исследователь может дать *примерную* оценку стандартного отклонения; *б*) можно попытаться найти соответствующие *литературные данные*. Предположим, что мы воспользовались вторым вариантом и выяснили, что стандартное отклонение массы тела для изучаемого вида жуков составляет 1.8 мг.

Теперь у нас есть вся необходимая информация для расчета минимального объема выборки. В среде R соответствующие вычисления можно выполнить при помощи базовой функции **power.t.test**():

```
power.t.test(delta = 3.0,
             sd = 1.8,
             sig.level = 0.05,
             power = 0.8)
     Two-sample t test power calculation
              n = 6.76095
          delta = 3
             sd = 1.8
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
NOTE: n is number in *each* group
```

В приведенной выше команде `delta` – минимальная величина эффекта, которую мы хотим обнаружить в ходе эксперимента; `sd` – стандартное отклонение веса жуков (по литературным данным); `sig.level` – уровень значимости, а `power` – мощность *t*-критерия. В выведенных результатах программа еще раз перечисляет имеющиеся исходные параметры, а также сообщает `n` – рассчитанный минимальный размер *каждой* выборки, необходимый для выявления желаемого эффекта при этих параметрах (округлив, получаем 7 жуков в каждой экспериментальной группе). Кроме того, программа напоминает нам, что вычисления были выполнены для двухстороннего критерия Стьюдента (`alternative = two.sided`) и что параметр `n` соответствует числу наблюдений в каждой группе (`n is number in *each* group`).

Зная число наблюдений, величину эффекта, стандартное отклонение и уровень значимости, мы можем рассчитать мощность теста:

```
power.t.test(n = 15,
             delta = 3.0,
             sd = 1.8,
             sig.level = 0.05)
     Two-sample t test power calculation
              n = 15
          delta = 3
             sd = 1.8
      sig.level = 0.05
          power = 0.9927162
    alternative = two.sided
NOTE: n is number in *each* group
```

Как видим, при n = 15, delta = 3, sd = 1.8 и sig.level = 0.05 мощность критерия составит 99%.

При необходимости выполнить вычисления для парного критерия Стьюдента в вызове функции **power.t.test**() достаточно добавить аргумент type = "paired":

```
power.t.test(delta = 3.0,
             sd = 1.8,
             sig.level = 0.05,
             power = 0.8,
             type  = "paired")
     Paired t test power calculation
              n = 5.04919
          delta = 3
             sd = 1.8
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs
```

Видно, что в случае с зависимыми наблюдениями минимальный размер выборок, необходимый для выявления заданной величины эффекта, несколько меньше, чем в случае с независимыми выборками (в рассматриваемом примере – 5 против 7 жуков в каждой группе).

Наконец, при необходимости выполнить одновыборочный *t*-тест аргументу type следует присвоить значение "one.sample":

```
power.t.test(delta = 3.0,
             sd = 1.8,
             sig.level = 0.05,
             power = 0.8,
             type  = "one.sample")
     One-sample t test power calculation
              n = 5.04919
          delta = 3
             sd = 1.8
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

В заключение следует еще раз подчеркнуть, что если в ходе выполнения *t*-теста (или любого другого теста) проверяемая нулевая гипотеза не отклонена, это не значит, что эффект в действительности отсутствует. Возможно, объемы выборок были просто недостаточно велики для этого. Поэтому рекомендуется приводить в отчетах (статьях, презентациях и т. п.) не только результаты статистического теста как такового, но еще и информацию о его мощности (оцененную *до проведения* исследования). Это позволит читателям отчета сформировать более четкое представление о том, насколько полученные данные поддерживают сделанные по результатам анализа выводы.

5.4. Гипотеза об однородности дисперсий

Проверка однородности дисперсии в двух группах

В общем виде *F-критерий Фишера*, или *F*-тест, используется для сравнения дисперсий двух нормально распределенных генеральных совокупностей, то есть проверятся нулевая гипотеза $H_0 : \sigma_1^2 = \sigma_2^2$. Генеральные дисперсии оцениваются на основе выборок, и критерий рассчитывается как отношение одной выборочной дисперсии к другой: $F = s_1^2/s_2^2$. В числитель приведенной формулы принято помещать бóльшую дисперсию, а в знаменатель – меньшую.

Критерий Фишера можно использовать для сравнения и более чем двух совокупностей (как, например, в дисперсионном анализе). В таких случаях критерий рассчитывается как отношение *межгрупповой дисперсии* к *внутригрупповой дисперсии*. Кроме того, *F*-критерий используется при оценке значимости линейной регрессии (подробнее см. главу 7).

Очевидно, что чем ближе рассчитанное значение *F* к 1, тем больше у нас оснований сделать заключение о справедливости приведенной выше нулевой гипотезы. И наоборот: чем выше это значение, тем больше имеется оснований отклонить нулевую гипотезу о равенстве дисперсий. *Критическое значение F*, начиная с которого нулевую гипотезу отклоняют, определяется уровнем значимости (например, $\alpha = 0.05$) и числом степеней свободы для каждой из сравниваемых дисперсий. Кроме того, справедливость нулевой гипотезы можно выразить при помощи *p*-значения для *F*-критерия, то есть вероятности того, что случайная величина с соответствующим распределением Фишера окажется равной или превысит значение *F*, рассчитанное по выборочным данным.

При выполнении *F*-теста и интерпретации получаемых с его помощью результатов важно помнить о следующих теоретических допущениях (см., например, Zar, 2010):

- обе сравниваемые совокупности нормально распределены;
- наблюдения в этих совокупностях независимы.

Для выполнения теста Фишера в R имеется функция **var.test**() (от «*variance*» – дисперсия и «*test*» – тест). Используем рассмотренный ранее пример о суточном потреблении энергии у худощавых женщин (lean) и женщин с избыточным весом (obese):

```
data(energy, package = "ISwR")
attach(energy)
```

Дисперсии в этих двух весовых группах можно легко сравнить следующим образом:

```
var.test(expend ~ stature)
        F test to compare two variances
data:  expend by stature
F = 0.7844, num df = 12, denom df = 8, p-value = 0.6797
alternative hypothesis: true ratio of variances is not equal to 1
```

```
95 percent confidence interval:
 0.1867876 2.7547991
sample estimates:
ratio of variances
          0.784446
```

Как видим, полученное *p*-значение значительно превышает 5%-ный уровень значимости, на основании чего мы *не можем* отклонить нулевую гипотезу о равенстве дисперсий в исследованных совокупностях. 95%-ный доверительный интервал для истинного отношения сравниваемых дисперсий ограничен значениями от 0.19 до 2.75 (см. `95 percent confidence interval`).

Исходя из данного результата, мы, например, вправе были бы использовать вариант *t*-критерия Стьюдента для совокупностей с одинаковыми дисперсиями при сравнении среднего потребления энергии у женщин из рассматриваемых весовых групп (подробнее см. раздел 5.1).

Проверка однородности дисперсии в нескольких группах

Решение более общей задачи проверки однородности дисперсии в двух или более группах осуществляется с использованием различных классических и непараметрических тестов: Левене, Бартлетта, Кохрана, Хартли, Флигнера, Ансари-Бредли, Сижела-Тьюки, Муда и др. Подробное исследование распределения используемых при этом статистик и сравнительный анализ мощности критериев можно найти, например, в работе Лемешко и соавторов (2013, http://bit.ly/1FkbeV4).

Проверяемая гипотеза о постоянстве дисперсий k выборок имеет вид $H_0 : \sigma_1^2 = \sigma_2^2 = ... = \sigma_k^2$, а конкурирующая с ней гипотеза $H_A : \sigma_i^2 \neq \sigma_j^2$, где неравенство выполняется хотя бы для одной пары индексов i, j.

Критерий Левене («*Levene's test*») имеет вид:

$$L = \frac{(n-k)\sum_{i=1}^{k} n_i (\bar{z}_{i\bullet} - \bar{z}_{\bullet\bullet})}{(k-1)\sum_{i=1}^{k}\sum_{j=1}^{n_i} (z_{ij} - \bar{z}_{i\bullet})},$$

где $z_{ij} = |x_{ij} - c_i|$; c_i – среднее, усеченное среднее или медиана для i-й группы; n, n_i – объемы всех данных и i-й выборки соответственно, $\bullet$ – символ усреднения по индексу. Считается, что критерий Левене малочувствителен к отклонениям анализируемых выборок от нормального закона, хотя он же оказывается и менее мощным.

Критерий Бартлетта («*Bartlett's test*») вычисляется в соответствии с соотношениями:

$$\chi^2 = M\left[1 + \frac{1}{3(k-1)}\left(\sum_{i=1}^{k}\frac{1}{n_i - 1} - \frac{1}{n-k}\right)\right];$$

$$M = (n-k)\ln\left[\frac{1}{n-k}\sum_{i=1}^{k}(n_i - 1)s_i^2\right] - \sum_{i=1}^{k}(n_i - 1)\ln s_i^2,$$

где s_i^2 – оценка дисперсии в i-й группе. Тест Бартлетта не зависит от объема выборок, но чувствителен к отклонениям от нормальности распределения.

Критерий Флигнера-Килина («*median-centering Fligner-Killeen test*») не требует предположений о нормальности сравниваемых выборок и оценивает совокупное распределение *рангов* абсолютных разностей $z_{ij} = |x_{ij} - c_i|$, где c_i – выборочная медиана для i-й группы. Критерий Флигнера-Килина (формулу для его расчета можно найти на странице http://bit.ly/1x6baHi), как и критерий Бартлетта, имеет стандартное χ^2-распределение с $(k - 1)$ степенями свободы.

Рассмотрим реализацию перечисленных тестов в среде R на примере данных, полученных в ходе эксперимента по изучению эффективности шести видов инсектицидных средств (см. раздел 3.3). Тест Левене можно выполнить при помощи функции **leveneTest**() из пакета car:

```
library(car)
leveneTest(count ~ spray, data = InsectSprays)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value   Pr(>F)
group  5  3.8214 0.004223 **
      66
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

leveneTest(count ~ spray, data = InsectSprays, center = mean)
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value    Pr(>F)
group  5  6.4554 6.104e-05 ***
      66
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Тесты Бартлетта и Флигнера-Килина выполняются при помощи функций **bartlett.test**() и **fligner.test**() соответственно:

```
bartlett.test(count ~ spray, data = InsectSprays)
Bartlett test of homogeneity of variances
data:  count by spray
Bartlett's K-squared = 25.9598, df = 5, p-value = 9.085e-05

fligner.test(count ~ spray, data = InsectSprays)
Fligner-Killeen test of homogeneity of variances
data:  count by spray
Fligner-Killeen: med chi-squared = 14.4828, df = 5, p-value = 0.01282
```

Все тесты показывают, что нулевая гипотеза о равенстве дисперсий в группах должна быть отклонена, однако оценки значимости p существенно разнятся в зависимости от того, используется ли для анализа среднее или медиана (что косвенно свидетельствует о возможной невыполнимости допущения о нормальности распределений исследуемых совокупностей).

5.5. Введение в дисперсионный анализ

Рассмотренный ранее *t*-критерий Стьюдента (равно как и его непараметрические аналоги) предназначен для сравнения исключительно двух совокупностей. Однако часто он неверно используется для попарного сравнения большего числа групп, что приводит к так называемому *эффекту множественных сравнений* («*multiple comparisons*»; Гланц, 1999). Об этом эффекте и о том, как с ним бороться, мы поговорим позднее в главе 6. Здесь же мы опишем принципы *однофакторного дисперсионного анализа*, как раз предназначенного для *одновременного* сравнения средних значений двух и более групп. Принципы дисперсионного анализа (англ. «*analysis of variance*», ANOVA) были разработаны в 1920-х гг. сэром Рональдом Эйлмером Фишером (Ronald Aylmer Fisher) – «*гением, едва ли не в одиночку заложившим основы современной статистики*» (Hald, 1998, http://bit.ly/1G7UZsM).

Может возникнуть вопрос: почему метод, используемый для сравнения *средних* значений, называется *дисперсионным* анализом? Все дело в том, что при установлении разницы между средними значениями мы в действительности сравниваем дисперсии анализируемых совокупностей. Однако обо всем по порядку.

Постановка задачи

Рассмотренный ниже пример заимствован из книги Maindonald & Braun (2010). Имеются данные о весе кустов томатов (все растение целиком; `weight`, в кг), которые выращивали в течение 2 месяцев при трех экспериментальных условиях (`trt`, от «*treatment*») – при поливе водой (`Water`), в среде с добавлением удобрения (`Nutrient`), а также в среде с добавлением удобрения и гербицида 24-D (`Nutrient+24D`):

```
# Создадим таблицу с данными:
tomato <-
  data.frame(weight =
               c(1.5, 1.9, 1.3, 1.5, 2.4, 1.5, # water
                 1.5, 1.2, 1.2, 2.1, 2.9, 1.6, # nutrient
                 1.9, 1.6, 0.8, 1.15, 0.9, 1.6), # nutrient+24D
             trt = rep(c("Water", "Nutrient", "Nutrient+24D"), c(6, 6, 6)))
```

Переменная `trt` представляет собой фактор с тремя уровнями. Для более наглядного сравнения экспериментальных условий в последующем сделаем уровень «`Water`» базовым («*reference level*»), то есть уровнем, с которым R будет сравнивать все остальные уровни. Это можно сделать при помощи функции **`relevel`**`()`:

```
tomato$trt <- relevel(tomato$trt, ref = "Water")
```

Подлежащую проверке нулевую гипотезу можно записать как H_0: $\mu_1 = \mu_2 = \mu_3$, то есть все полученные измерения веса растений происходят из одной нормально распределенной генеральной совокупности. Другими словами, нулевая гипотеза утверждает, что в действительности исследованные условия выращивания расте-

ний не оказывают никакого влияния на вес последних, а *наблюдаемые различия между групповыми средними несущественны и вызваны влиянием случайных факторов*.

Подчеркнем еще раз, что рассматриваемый пример соответствует случаю *однофакторного* дисперсионного анализа: изучается действие одного фактора – условий выращивания (с тремя уровнями `Water`, `Nutrient` и `Nutrient+24D`) на интересующую нас переменную-отклик (вес растений).

К сожалению, исследователь почти никогда не имеет возможности изучить всю генеральную совокупность. Как же нам тогда узнать, верна ли приведенная выше нулевая гипотеза, располагая только выборочными данными? Мы можем сформулировать этот вопрос иначе: *какова вероятность получить наблюдаемые различия между групповыми средними, извлекая случайные выборки из одной нормально распределенной генеральной совокупности*? Для ответа на этот вопрос нам потребуется статистический критерий, который количественно характеризовал бы величину различий между сравниваемыми группами.

Перед тем как сконструировать такой критерий, зададимся еще одним вопросом: что заставляет нас думать, что различия между средними неслучайны? Чтобы лучше понять свойства имеющихся данных, изобразим данные в виде одномерной диаграммы рассеяния:

```
attach(tomato)
stripchart(weight ~ trt, xlab = "Вес, кг", ylab = "Условия")
```

Для подтверждения визуальных впечатлений рассчитаем групповые средние

```
(Means <- tapply(weight, trt, mean))
      Water     Nutrient Nutrient+24D
   1.683333     1.750000     1.325000
```

и построим график (рис. 58), на котором к исходным данным добавлена еще одна группа из точек, соответствующих выборочным средним (`Means`) .

Если сравнить на рис. 58 разброс значений *внутри* экспериментальных групп с *разбросом трех групповых средних*, видно, что разброс групповых средних в целом меньше, чем разброс значений в экспериментальных группах. Измеренные значения среднего веса растений достаточно близки для всех трех экспериментальных условий, хотя и есть некоторая тенденция к снижению веса в группе «`Nutrient+24D`».

Теперь (исключительно с целью продемонстрировать принцип!) несколько модифицируем исходные данные – см. рис. 59. Что изменилось, по сравнению с рис. 58? Группы точек, отражающих экспериментальные данные, оказались значительно раздвинутыми вдоль оси абсцисс. Изменился также разброс трех групповых средних, по сравнению с разбросом значений внутри экспериментальных групп, – разброс групповых средних превышает разброс значений внутри групп. Теперь, глядя на рис. 59, почти любой скажет, что условия выращивания оказывают влияние на вес растений.

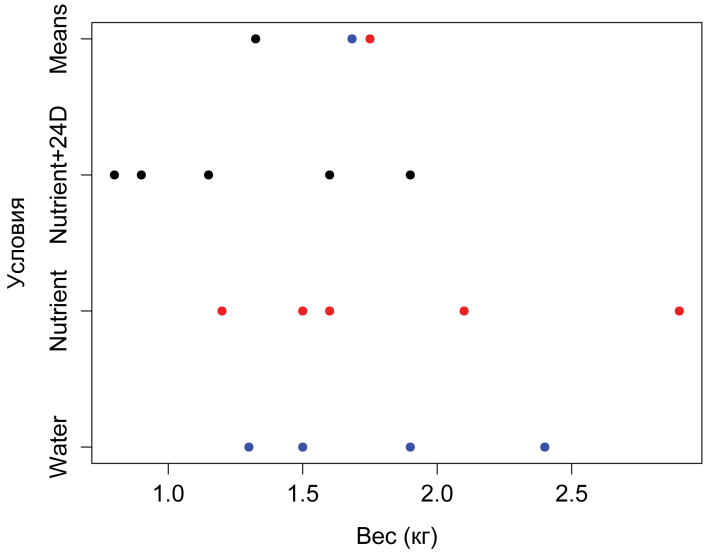


Рисунок 58

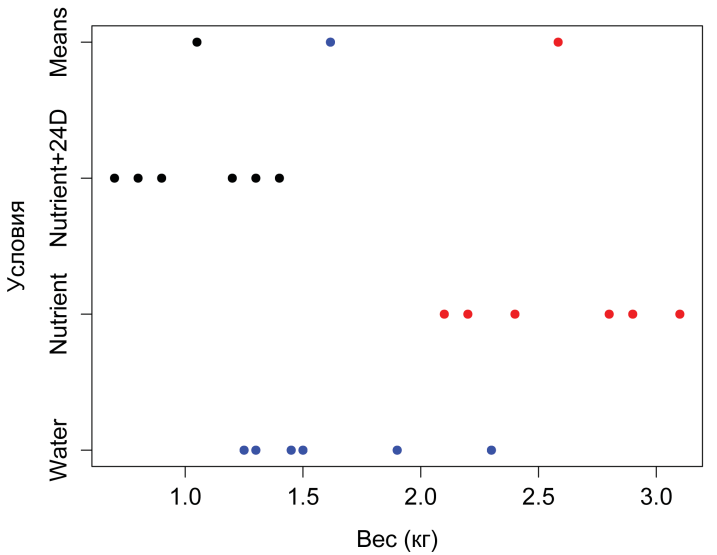


Рисунок 59

Таким образом, для оценки различий между группами следует каким-то образом сравнить разброс групповых средних с разбросом значений внутри групп. Это – ключевая идея дисперсионного анализа.

Две оценки генеральной дисперсии в дисперсионном анализе

Чем больше разброс выборочных средних и чем меньше разброс значений внутри групп, тем меньше вероятность того, что наши группы являются случайными выборками из одной совокупности. Дисперсию генеральной совокупности можно оценить двумя способами. С одной стороны, оценкой дисперсии генеральной совокупности будет дисперсия, вычисленная *для каждой группы*. Такая оценка не будет зависеть от различий групповых средних. С другой стороны, при верной нулевой гипотезе (см. выше) *разброс групповых средних* тоже позволит оценить дисперсию генеральной совокупности. Очевидно, что такая оценка уже будет зависеть от различий между группами.

Если экспериментальные группы – это случайные выборки из одной и той же нормально распределенной генеральной совокупности, то оба способа оценки генеральной дисперсии должны давать примерно одинаковые результаты. Соответственно, если эти оценки действительно оказываются близки, то мы не можем отвергнуть нулевую гипотезу. И наоборот: если разница между этими оценками оказывается существенной, имеет смысл принять альтернативную гипотезу: *маловероятно, что мы получили бы наблюдаемые различия между группами, если бы они были просто случайными выборками из одной нормально распределенной генеральной совокупности*.

Как именно проверить сформулированную гипотезу? Пусть x_{ij} обозначает наблюдение j в группе i (например, x_{13} будет третьим наблюдением из первой группы), $\bar{x}_{i\bullet}$ – среднее значение в группе i, а $\bar{x}_{\bullet\bullet}$ – общее среднее значение, рассчитанное по всем имеющимся наблюдениям. Тогда каждое наблюдение мы можем разложить на следующие составляющие: $x_{ij} = \bar{x}_{\bullet\bullet} + (\bar{x}_{i\bullet} - \bar{x}_{\bullet\bullet}) + (x_{ij} - \bar{x}_{i\bullet})$, где $(\bar{x}_{i\bullet} - \bar{x}_{\bullet\bullet})$ – отклонения групповых средних от общего среднего, а $(x_{ij} - \bar{x}_{i\bullet})$ – отклонения отдельных наблюдений от среднего значения группы, к которой они принадлежат.

Тогда разброс наблюдений *внутри групп* можно рассчитать как

$$SSW = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_{i\bullet})^2,$$

а разброс *между группами* (разброс групповых средних) – как

$$SSB = \sum_{i=1}^{k} (\bar{x}_{i\bullet} - \bar{x}_{\bullet\bullet})^2.$$

В приведенных выражениях буквы W и B соответствуют английским словам «*within*» (внутри) и «*between*» (между).

Нормализовав SSW и SSB по их соответствующим степеням свободы, получим *внутри- и межгрупповую дисперсии*, о которых мы говорили выше:

$$MSW = SSW/(n - k) \text{ и } MSB = SSB/(k - 1).$$

В последних двух выражениях n – общее число наблюдений, а k – число сравниваемых групп. Аббревиатура MS означает «*mean squares*» («средние квадраты»; имеются в виду усредненные суммы квадратов отклонений SSW и SSB) и часто

встречается в результатах дисперсионного анализа, выдаваемых статистическими программами.

Нормализация *SSW* и *SSB* по числу степеней свободы позволяет получить сравнимые величины. Формально внутри- и межгрупповые дисперсии сравниваются при помощи *F-критерия*, или *критерия Фишера* (обратите внимание: в числителе всегда находится межгрупповая дисперсия, *MSB*):

$F = MSB / MSW$.

Очевидно, что чем ближе F к 1, тем меньше у нас оснований утверждать, что внутри- и межгрупповая дисперсии различаются. Иными словами, у нас нет оснований отклонить сформулированную выше нулевую гипотезу. Если же F значительно выше 1, нулевую гипотезу можно отклонить. Возникает вопрос: начиная с какой именно величины F нулевую гипотезу можно отклонять?

Критическое значение F-критерия определяется желаемым уровнем значимости и свойствами F-распределения, форма которого полностью задается меж- и внутригрупповым степенями свободы. Так, для нашего примера межгрупповое число степеней свободы составляет $df_B = k - 1 = 3 - 1 = 2$, а внутригрупповое – $df_W = n - k = 18 - 3 = 15$. Внешний вид F-распределения при этих степенях свободы представлен на рис. 60.

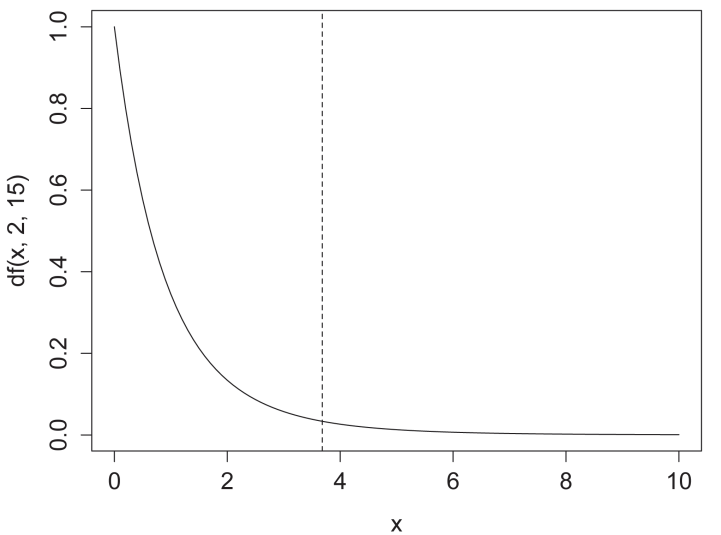


Рисунок 60

Вертикальная линия на рис. 60 соответствует 3.682 – критическому значению F при $\alpha = 0.05$. Если F-значение, рассчитанное по экспериментальным данным, превышает критическое значение, мы можем отклонить нулевую гипотезу об отсутствии эффекта изучаемого фактора.

Выполнение дисперсионного анализа в R

Дисперсионный анализ в R можно выполнить при помощи базовых функций **aov**() или **lm**(). Для нашего примера получаем:

```
summary(aov(weight ~ trt, data = tomato))
            Df Sum Sq Mean Sq F value Pr(>F)
trt          2  0.627  0.3135   1.202  0.328
Residuals   15  3.912  0.2608
```

В приведенных результатах строка, обозначенная как `trt`, соответствует источнику дисперсии в данных – условиям выращивания растений. Строка, обозначенная как `Residuals`, характеризует внутригрупповую дисперсию (ее еще называют *остаточной,* или *флуктуирующей*, дисперсией – в том смысле, что она не может быть объяснена влиянием экспериментального фактора и является следствием случайной флуктуации).

Столбец `Sum Sq` содержит *SSB* и *SSW*, а столбец `Mean Sq` – меж- и внутригрупповую дисперсии *MSB* и *MSW*. В столбце `F value` представлено оцененное по выборочным данным значение *F*-критерия. Наконец, в столбце `Pr(>F)` представлена вероятность получить *F*-значение, равное или превышающее наблюдаемое значение, при условии что нулевая гипотеза верна. Как видим, эта вероятность достаточно высока. Во всяком случае, она превышает 5%-ный уровень значимости, и на этом основании мы заключаем, что нулевая гипотеза верна. Таким образом, с достаточно высокой степенью уверенности мы можем утверждать, что экспериментальные условия не оказали существенного влияния на вес растений.

Двухфакторный дисперсионный анализ

Описанный однофакторный дисперсионный анализ заключается в выяснении влияния какого-то одного фактора на интересующую нас количественную переменную. Однако очень редко тот или иной процесс определяется только одним фактором. Напротив – обычно наблюдается одновременное влияние многих факторов. Задача исследователя – выявить, какие факторы оказывают существенное влияние на изучаемое явление, а какие можно исключить из рассмотрения. Как будет показано ниже, двухфакторный дисперсионный анализ («*two-way analysis of variance*», или «*two-way ANOVA*») позволяет установить одновременное влияние двух факторов, а также взаимодействие между этими факторами. При наличии более двух факторов говорят о *многофакторном дисперсионном анализе* («*multifactor ANOVA*», который не следует путать с многомерным анализом *MANOVA* – «*multivariate ANOVA*»).

В качестве примера рассмотрим результаты реального эксперимента (Hand et al., 1993, http://bit.ly/1GSiBVL), в котором лабораторным крысам примерно одинакового возраста и веса в течение определенного времени давали корм с разным содержанием белка (фактор `type` с двумя уровнями: низкое содержание – `Low` и высокое содержание – `High`). Кроме того, корма различались по происхождению белка (фактор `source` с двумя уровнями: `beef` – говядина и `cereal` – злаки). В кон-

це эксперимента был измерен прирост веса у крыс (weightgain) в каждой группе. Таблица с данными из этого эксперимента входит в состав пакета HSAUR2, который является приложением к одной из лучших, на наш взгляд, вводных книг по R – Everitt & Hothorn (2010). Структура таблицы weightgain выглядит так:

```
library(HSAUR2)
data(weightgain)
str(weightgain)
'data.frame': 40 obs. of  3 variables:
 $ source    : Factor w/ 2 levels "Beef","Cereal": 1 1 1 1 1 1 1 1...
 $ type      : Factor w/ 2 levels "High","Low": 2 2 2 2 2 2 2 2 ...
 $ weightgain: int  90 76 90 64 86 51 72 90 95 78 ...
```

Перед любыми статистическими расчетами полезно выполнить разведочный анализ данных и изучить структуру данных на графиках (рис. 61):

```
library(ggplot2)
ggplot(data = weightgain, aes(x = type, y = weightgain)) +
       geom_boxplot(aes(fill = source))
```

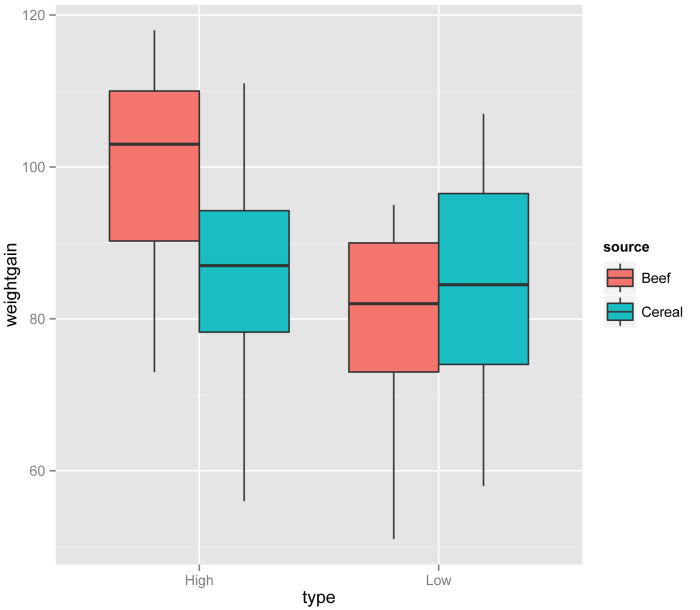


Рисунок 61

Подробнее об использованном в предыдущей команде графическом пакете ggplot2 см., например, http://bit.ly/1GHFsjg.

Также стоит ознакомиться со сводными описательными статистиками, используя, например, возможности пакета doBy:

```
require(doBy)
summaryBy(weightgain ~ type + source, data = weightgain,
          FUN = c(mean, sd, length))
  type source weightgain.mean weightgain.sd weightgain.length
1 High   Beef           100.0       15.13642                10
2 High Cereal            85.9       15.02184                10
3  Low   Beef            79.2       13.88684                10
4  Low Cereal            83.9       15.70881                10
```

Средние значения прироста веса в исследованных группах заметно варьируют. Видно, например, что прирост веса у животных, которых содержали на корме с низкой концентрацией белка животного происхождения, оказался существенно ниже, чем в группе «High - Beef». Задача двухфакторного дисперсионного анализа – выяснить, связаны ли наблюдаемые различия в приросте веса с изучаемыми факторами либо эти различия случайны и не имеют никакого отношения к содержанию белка в корме и его происхождению.

Полезным приемом, позволяющим лучше понять анализируемые эффекты, является также построение графика «плана эксперимента» («*design plot*»). На таком графике отображаются средние значения переменной-отклика в соответствии с каждым уровнем изучаемых факторов (рис. 62):

```
plot.design(weightgain)
```

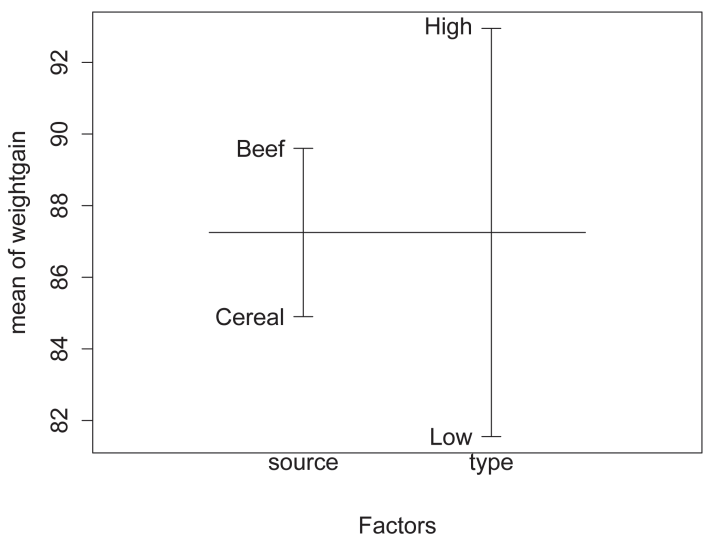


Рисунок 62

Из полученного графика видно, что наибольшая разница в средних приростах веса крыс связана с уровнем содержания белка в корме, тогда как эффект источника происхождения белка выражен в меньшей степени.

Рассматриваемое исследование можно отнести к *полнофакторному эксперименту* («*full factorial experiment*»), поскольку в нем реализуются все возможные сочетания имеющихся уровней факторов. Значительное преимущество такого дизайна эксперимента заключается в том, что он позволяет выяснить наличие взаимодействия между изучаемыми факторами. В рамках дисперсионного анализа под взаимодействием («*interaction*») понимают такую ситуацию, когда переменная-отклик ведет себя по-разному в зависимости от сочетания изучаемых факторов. Понять эту концепцию поможет «график взаимодействий» («*interaction plot*»), который в R можно построить при помощи базовой функции **interaction.plot**() (рис. 63).

```
with(weightgain, interaction.plot(x.factor = type,
                                  trace.factor = source,
                                  response = weightgain))
```

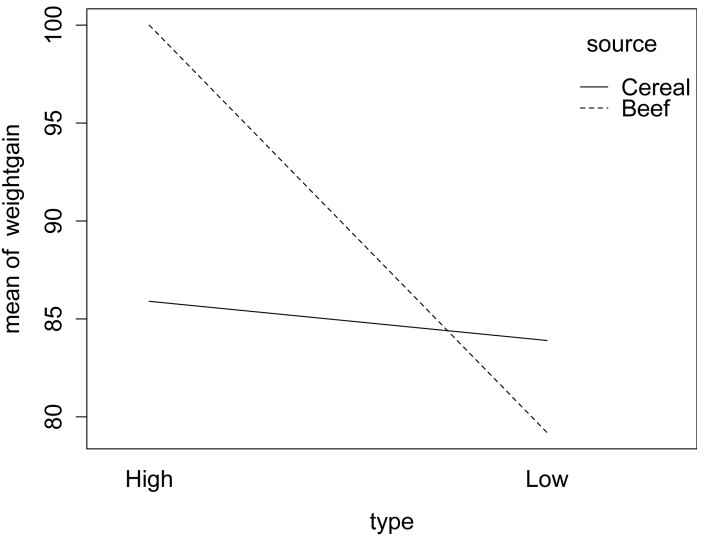


Рисунок 63

По рис. 63 видно, что при высоком содержании белка в корме прирост веса крыс в среднем так же высок, но только если этот белок имеет животное происхождение. Если же содержание белка низкое, то ситуация меняется на противоположную – прирост оказывается несколько выше (хотя и не намного) в группе крыс, получавших корм растительного происхождения.

Помимо того что рассматриваемый эксперимент является полнофакторным, во всех четырех группах имеется также одинаковое число крыс (по 10 в каждой), то есть мы имеем дело со *сбалансированным* набором данных. Для анализа сбалансированных наборов данных мы можем применить классический способ разложения

общей дисперсии на отдельные составляющие, реализованный в уже знакомой нам функции **aov**():

```
M1 <- aov(weightgain ~ source + type + source:type,
                  data = weightgain)
summary(M1)
             Df Sum Sq Mean Sq F value Pr(>F)
source        1    221   220.9   0.988 0.3269
type          1   1300  1299.6   5.812 0.0211 *
source:type   1    884   883.6   3.952 0.0545 .
Residuals    36   8049   223.6
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Подробное объяснение того, как интерпретировать полученную таблицу дисперсионного анализа, было приведено ранее. В целом можно сделать вывод об отсутствии статистически значимой связи между приростом веса крыс и источником белка в корме (p = 0.3269), тогда как влияние уровня содержания белка в корме оказалось значимым (p = 0.0211). Взаимодействие между источником происхождения белка и уровнем его содержания в корме незначимо (p = 0.0545), что, в принципе, согласуется с результатом анализа приведенного выше графика взаимодействий.

Обратите внимание на то, как в формуле, поданной на функцию **aov**(), было задано взаимодействие между двумя факторами: сначала были приведены два главных фактора, разделенные знаком «+», а затем к ним добавлено выражение «source:type». Это стандартный синтаксис для такого рода анализа в R. Однако приведенную формулу можно было бы также сократить до weightgain ~ source*type – результат оказался бы идентичным. Подробнее о синтаксисе формул в R мы расскажем в главе 6, а сводку правил можно получить, например, на странице http://bit.ly/MArWaE (англ. яз.).

Как следует из названия, данный раздел является введением в дисперсионный анализ и его выполнение при помощи R. В главе 6 мы продолжим эту тему и обсудим такие вопросы, как условия применимости параметрического дисперсионного анализа и способы их проверки, множественные сравнения групп, расчет необходимого числа наблюдений для дисперсионного анализа и др.

5.6. Оценка корреляции двух случайных величин

Коэффициент корреляции r – удобный показатель степени взаимосвязи между двумя случайными величинами (X, Y), который представляет собой безразмерную величину, изменяющуюся от –1 до +1. Имея пары значений этих величин, мы можем рассчитать наиболее часто используемый коэффициент корреляции – коэффициент Пирсона («*Pearson correlation coefficient*»):

$$r_{xy} = \frac{Cov(x,y)}{\sqrt{S_x^2}\sqrt{S_y^2}},$$

где $S_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$ и $S_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$ – выборочные дисперсии значений X и Y, а $Cov(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$ – ковариация, характеризующая совместное распределение (X, Y) в двумерном евклидовом пространстве.

При независимом варьировании переменных, когда связь между ними отсутствует, $r = 0$. Чем сильнее связь, тем больше абсолютная величина коэффициента корреляции. При этом положительные значения r указывают на положительную, или «прямую», связь, когда при увеличении значений одной переменной в среднем возрастают значения и другой переменной, а отрицательные r – на отрицательную, или «обратную», связь, когда при возрастании одной переменной другая в среднем уменьшается.

Вычисление коэффициента корреляции в R мы продемонстрируем на примере данных по уровню инвазированности двустворчатого моллюска *Dreissena polymorpha* инфузорией-комменсалом *Conchophthirus acuminatus* (см. также разделы 1.3, 4.3):

```
dat <- read.delim("http://bit.ly/1wMGDOQ")
```

В таблице `dat` нас интересуют, в частности, две переменные: длина раковины моллюска (`ZMlength`, мм) и число обнаруженных в моллюске инфузорий (`CAnumber`). В разделе 1.3 мы с помощью R Commander уже рассчитали коэффициент корреляции и на построенном графике обнаружили положительную связь между этими двумя переменными.

Теперь мы рассмотрим, как можно оценить корреляцию, обратившись непосредственно в среде R к функциям **`cor`**`()` и **`cor.test`**`()`. Различие между этими двумя функциями заключается в том, что **`cor`**`()` позволяет вычислить только сам коэффициент корреляции, тогда как **`cor.test`**`()` выполняет еще и оценку статистической значимости коэффициента, проверяя нулевую гипотезу о равенстве его нулю. Естественно, предпочтительнее использовать именно вторую функцию:

```
attach(dat)
cor.test(CAnumber, ZMlength)
Pearson's product-moment correlation
data:  CAnumber and ZMlength
t = 11.4964, df = 474, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.3935877 0.5343949
sample estimates:
     cor
0.466946
```

Как видим, рассчитанный коэффициент корреляции Пирсона оказался равен 0.467. Несмотря на то что он не очень высок, этот коэффициент статистически значимо отличается от нуля (`p-value < 2.2e-16`). Для нашего удобства программа также автоматически вычислила 95%-ный доверительный интервал для полученного коэффициента корреляции (`95 percent confidence interval: 0.394 0.534`).

Необходимо помнить, что коэффициент корреляции Пирсона основан на следующих важных допущениях:

- обе анализируемые переменные распределены нормально;
- связь между этими переменными линейна.

В разделе 4.3 мы на этом примере показали, что как минимум в отношении значений интенсивности инвазии условие нормальности распределения не выполняется, и для исправления ситуации выполнили преобразование Бокса-Кокса. Здесь мы ограничимся тем, что прологарифмируем обе переменные `ZMlength` и `CAnumber`. Для преобразованных переменных коэффициент корреляции Пирсона составит:

```
cor.test(log(CAnumber+1), log(ZMlength))
Pearson's product-moment correlation
data:  log(CAnumber + 1) and log(ZMlength)
t = 21.5166, df = 474, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.6543961 0.7456953
sample estimates:
      cor
0.7029297
```

Поскольку логарифмирование часто позволяет привести данные к распределению, близкому к нормальному, то неудивительно, что новое значение коэффициента корреляции значительно возросло (0.703 против 0.467).

И все же, несмотря на логарифмирование, значения интенсивности инвазии *C. acuminatus* не подчиняются нормальному распределению. Показать это можно как графически, так и при помощи, например, теста Шапиро-Уилка (подробнее о проверке данных на нормальность см. раздел 4.8):

```
shapiro.test(log(CAnumber+1))
Shapiro-Wilk normality test
data:  log(CAnumber + 1)
W = 0.9508, p-value = 1.734e-11
```

Для ненормально распределенных переменных, а также при наличии нелинейной связи между переменными следует использовать непараметрический *коэффициент корреляции Спирмена* («*Spearman correlation coefficient*»). В отличие от коэффициента Пирсона, этот вариант коэффициента корреляции работает не с исходными значениями переменных, а с их рангами (однако формула при этом используется та же, что и для коэффициента Пирсона). Для вычисления коэффициента Спирмена в R при вызове функции **`cor.test`**() необходимо воспользоваться аргументом `method` со значением `"spearman"`:

```
cor.test(CAnumber, ZMlength, method = "spearman")
Spearman's rank correlation rho
data:  CAnumber and ZMlength
S = 6574110, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
```

```
sample estimates:
      rho
0.6342627
Warning message:
In cor.test.default(CAnumber, ZMlength, method = "spearman"):
  Cannot compute exact p-values with ties
```

Коэффициент корреляции Спирмена составил 0.634 и оказался статистически значимым ($p << 0.001$). Поскольку в данных имеют место значения с одинаковыми рангами («*связанные ранги*», англ. «*ties*»), программа не смогла рассчитать так называемое точное *p*-значение, о чем предупредила нас в сообщении «`Warning message:...`».

В связи с тем, что коэффициент корреляции Спирмена работает с рангами, любое преобразование исходных данных никак не сказывается на его значении. Например, после логарифмирования мы получим результат, идентичный предыдущему:

```
cor.test(log(CAnumber+1), log(ZMlength), method = "spearman")
Spearman's rank correlation rho
data:  log(CAnumber + 1) and log(ZMlength)
S = 6574110, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.6342627
```

Наконец, третий вариант коэффициента корреляции, который можно рассчитать при помощи функции **`cor.test`**`()`, имеет название *коэффициент ранговой корреляции Кендалла* τ («*Kendall's tau*»). Вычисляется он следующим образом. Предположим, что у нас есть набор из парных наблюдений двух переменных: (x_1, y_1), (x_2, y_2) ... (x_n, y_n). Говорят, что две пары наблюдений (x_i, y_i) и (x_j, y_j) являются *конкордантными*, если имеется согласие между рангами соответствующих элементов этих пар, то есть если

$$x_i > x_j \text{ и } y_i > y_j$$

или

$$x_i < x_j \text{ и } y_i < y_j.$$

Если найти число конкордантных пар (*nconc.pairs*) и число *дискордантных* пар (*ndiscord.pairs*), то коэффициент Кендалла составит

$$\tau = 2(nconc.pairs - ndiscord.pairs) / n(n - 1).$$

Коэффициент Кендалла часто используют при анализе согласованности результатов измерений, полученных при помощи разных приборов, результатов голосования разных экспертов по одному и тому же вопросу и т. п. В R коэффициент Кендалла можно вычислить следующим образом:

```
cor.test(CAnumber, ZMlength, method = "kendall")
Kendalls rank correlation tau
data:  CAnumber and ZMlength
z = 14.9307, p-value < 2.2e-16
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.4655232
```

Подробнее об аргументах функции **cor.test**() можно узнать из ее справочного файла (см. ?cor.test).

5.7. Критерий хи-квадрат

Критерий хи-квадрат для таблиц сопряженности размером 2×2

Критерий χ^2 («хи-квадрат», или «критерий согласия Пирсона») имеет чрезвычайно широкое применение в статистике. В общем виде можно сказать, что он используется для проверки нулевой гипотезы о подчинении наблюдаемой случайной величины определенному теоретическому закону распределения. Однако конкретная формулировка проверяемой гипотезы от случая к случаю будет варьировать.

В этом разделе мы опишем принцип работы критерия χ^2 на гипотетическом примере из иммунологии. Представим, что мы выполнили эксперимент по установлению эффективности подавления развития микробного заболевания при введении в организм соответствующих антител. Всего в эксперименте было задействовано 111 мышей, которых мы разделили на две группы, включающие 57 и 54 животных соответственно. Первой группе мышей сделали инъекции патогенных бактерий с последующим введением сыворотки крови, содержащей антитела против этих бактерий. Животные из второй группы служили контролем – им сделали только бактериальные инъекции. После некоторого времени инкубации оказалось, что 38 мышей погибли, а 73 выжили. Из погибших 13 принадлежали первой группе, а 25 – ко второй (контрольной).

Проверяемую в этом эксперименте нулевую гипотезу можно сформулировать так: *введение сыворотки с антителами не оказывает никакого влияния на выживаемость мышей*. Иными словами, мы утверждаем, что наблюдаемые различия в выживаемости мышей (77.2% в первой группе против 53.7% во второй группе) совершенно случайны и не связаны с действием антител. Полученные в эксперименте данные можно представить в виде таблицы:

Группа	Погибло	Выжило	Всего
Бактерии + сыворотка	13	44	**57**
Только бактерии	25	29	**54**
Всего	**38**	**73**	**111**

Таблицы, подобные приведенной, называют *таблицами сопряженности* («*contingency tables*»). В рассматриваемом примере таблица имеет размерность 2×2: есть два класса объектов («Бактерии + сыворотка» и «Только бактерии»), которые исследуются по двум признакам («Погибло» и «Выжило»). Это простейший случай таблицы сопряженности: безусловно, и количество исследуемых классов, и количество признаков в более сложных экспериментах может быть бóльшим.

Для проверки сформулированной выше нулевой гипотезы нам необходимо знать, какова была бы ситуация, если бы антитела действительно не оказывали никакого действия на выживаемость мышей. Другими словами, нужно рассчитать *ожидаемые частоты* для соответствующих ячеек таблицы сопряженности. Как это сделать?

Всего в эксперименте погибло 38 мышей, что составляет 34.2% от общего числа задействованных животных. Если введение антител не влияет на выживаемость мышей, в обеих экспериментальных группах должен наблюдаться одинаковый процент смертности, а именно 34.2%. Рассчитав, сколько составляет 34.2% от 57 и 54, получим 19.5 и 18.5. Это и есть ожидаемые величины смертности в наших экспериментальных группах. Аналогичным образом рассчитываются и ожидаемые величины выживаемости: поскольку всего выжили 73 мыши, или 65.8% от общего их числа, то ожидаемые частоты выживаемости составят 37.5 и 35.5. Составим новую таблицу сопряженности, теперь уже с ожидаемыми частотами:

Группа	Погибшие	Выжившие	Всего
Бактерии + сыворотка	19.5	37.5	**57**
Только бактерии	18.5	35.5	**54**
Всего	**38**	**73**	**111**

Как видим, ожидаемые частоты довольно сильно отличаются от наблюдаемых, то есть введение антител, похоже, все-таки оказывает влияние на выживаемость мышей, зараженных патогенным микроорганизмом. Это впечатление мы можем выразить количественно при помощи критерия согласия Пирсона χ^2:

$$\chi^2 = \Sigma(f_o - f_e)^2/f_e,$$

где f_o и f_e – наблюдаемые и ожидаемые частоты соответственно. Суммирование производится по всем ячейкам таблицы. Так, для рассматриваемого примера имеем

$$\chi^2 = (13 - 19.5)^2/19.5 + (44 - 37.5)^2/37.5 + (25 - 18.5)^2/18.5 + (29 - 35.5)^2/35.5 = \\ = 2.16 + 1.12 + 2.31 + 1.20 = 6.79.$$

Достаточно ли велико полученное значение χ^2, чтобы отклонить нулевую гипотезу? Для ответа на этот вопрос необходимо найти соответствующее критическое значение критерия. Число степеней свободы для χ^2 рассчитывается как $df = (R - 1)(C - 1)$, где R и C – количество строк и столбцов в таблице сопряженности.

В нашем случае $df = (2 - 1)(2 - 1) = 1$. Зная число степеней свободы, мы легко можем узнать критическое значение χ^2 при помощи стандартной R-функции **qchisq**():

```
qchisq(p = 0.95, df = 1)
[1] 3.841459
```

Таким образом, при одной степени свободы только в 5% случаев величина критерия χ^2 превышает 3.841. Полученное нами значение 6.79 значительно превышает это критическое значение, что дает нам право отвергнуть нулевую гипотезу об отсутствии связи между введением антител и выживаемостью зараженных мышей. Отвергая эту гипотезу, мы рискуем ошибиться с вероятностью менее 5%.

Следует отметить, что приведенная выше формула для критерия χ^2 дает несколько завышенные значения при работе с таблицами сопряженности размером 2×2. Причина заключается в том, что распределение самого критерия χ^2 является непрерывным, тогда как частоты бинарных признаков («погибло» / «выжило») по определению дискретны. В связи с этим при расчете критерия принято вводить так называемую *поправку на непрерывность*, или *поправку Йетса* («*Yates' correction for continuity*»):

$$\chi^2_Y = \sum \frac{(|f_0 - f_e| - 0.5)^2}{f_e}.$$

В нашем случае критерий χ^2 с поправкой на непрерывность составил бы 5.792, и нулевая гипотеза об отсутствии эффекта антител все равно была бы отклонена. Возможно, однако, что в других ситуациях сделать это так легко не удалось бы.

Безусловно, нет необходимости выполнять приведенные выше вычисления вручную. В R для этого имеется стандартная функция **chisq.test**(). При работе с этой функцией данные оформляются в виде матрицы, напоминающей приведенную выше таблицу сопряженности:

```
mice <- matrix(c(13, 44, 25, 29), nrow = 2, byrow = TRUE)
mice # просмотр содержимого матрицы
     [,1] [,2]
[1,]   13   44
[2,]   25   29

chisq.test(mice) # тест хи-квадрат
Pearson's Chi-squared test with Yates' continuity correction
data:  mice
X-squared = 5.7923, df = 1, p-value = 0.0161
```

Как видим, R автоматически применяет поправку Йетса на непрерывность (`Pearson's Chi-squared test with Yates' continuity correction`). Рассчитанное программой значение χ^2 составило 5.79213. Мы можем отклонить нулевую гипотезу об отсутствии эффекта антител, рискуя ошибиться с вероятностью чуть более 1% (`p-value = 0.0161`).

Критерий хи-квадрат для таблиц сопряженности размером больше 2×2

Выше мы описали принцип работы критерия χ^2 и привели пример его применения в отношении простейшего случая – анализа частот, сведенных в таблицу сопряженности размером 2×2. Однако на практике можно столкнуться с данными, имеющими и более сложную структуру. В целом можно говорить о таблицах сопряженности размером $R \times C$, где R – количество строк, а C – количество столбцов в таблице. Число степеней свободы в таких таблицах составляет $df = (R - 1)(C - 1)$.

Предположим, мы выполнили исследование трех популяций наземных моллюсков, в ходе которого отмечали цвет раковины. Цвет выражался следующими тремя градациями – «светлый» (`light`), «темный» (`dark`) и «очень темный» (`very dark`). Необходимо выяснить, различаются ли частоты встречаемости каждого из вариантов окраски в исследованных популяциях. Допустим, что учтенные частоты выражались следующими значениями:

```
light <- c(12, 40, 45)
dark <- c(87, 34, 75)
very.dark <- c(3, 8, 2)
```

Соберем все три вектора в одну матрицу:

```
color.data <- matrix(c(light, dark, very.dark), nrow = 3,
             dimnames = list(c("Pop1", "Pop2", "Pop3"),
             c("Light", "Dark", "Very dark")))
color.data
     Light Dark Very dark
Pop1    12   87         3
Pop2    40   34         8
Pop3    45   75         2
```

Дальше достаточно подать эту матрицу на вход функции **`chisq.test`**():

```
chisq.test(color.data)
Pearson's Chi-squared test
data:  color.data
X-squared = 43.4337, df = 4, p-value = 8.411e-09
Warning message:
In chisq.test(color.data): Chi-squared approximation may be incorrect
```

Поскольку полученное `p-value = 8.411e-09` намного меньше 0.05, у нас есть все основания отклонить нулевую гипотезу об отсутствии разницы между популяциями моллюска по частотам встречаемости разных вариантов окраски раковины.

5.8. Точный тест Фишера. Критерии Мак-Немара и Кохрана-Мантеля-Хензеля

Точный тест Фишера

Одно из условий применимости рассмотренного выше критерия χ^2 состоит в том, что ожидаемые частоты в любой из ячеек таблицы сопряженности *не должны быть*

меньше 5 (Гланц, 1999). При небольшом числе наблюдений это условие может не выполняться, и тогда более корректным подходом, с точки зрения получаемых *p*-значений, будет использование *точного теста Фишера* («*Fisher's exact test*»).

Тест основан на переборе всех возможных вариантов заполнения таблицы сопряженности при имеющейся численности групп, и поэтому чем эта численность меньше, тем проще выполнить соответствующие вычисления. Фишер показал, что вероятность получения любого набора частот n_{ij} таблицы размером $R \times C$ задается гипергеометрическим распределением вида

$$P = \frac{(R_1!R_2!...R_r!)(C_1!C_2!...C_c!)}{N!\prod_{i,j} n_{ij}!},$$

где R_i и C_j – суммы по строкам и столбцам соответственно, N – общее число наблюдений, а знак «!», как обычно, означает факториал.

	Столбец 1	**Столбец 2**	**Суммы по строкам**
Строка 1	n_{11}	n_{12}	R_1
Строка 2	n_{21}	n_{22}	R_2
Суммы по столбцам	C_1	C_2	N

Построив все варианты заполнения таблицы, возможные при имеющихся маргинальных суммах, каждый раз рассчитывают соответствующие значения вероятности P по приведенной выше формуле. Вероятности, которые не превышают вероятность исходной таблицы, суммируют. Полученная сумма, включающая также вероятность исходной таблицы, и будет достигнутой статистической значимостью для *двухстороннего* варианта точного теста Фишера. Приведенный ниже пример из книги С. Гланца (1999) поможет лучше понять суть того, как работает этот тест.

Точный тест Фишера, в отличие от критерия χ^2, имеет одно- и двухсторонний варианты. Большинство авторов используют односторонний вариант (возможно, потому, что он дает меньшие значения P, что обычно выглядит более привлекательно в публикациях). Но даже при таком предвзятом подходе многие авторы, к сожалению, оставляют читателя в неведении по поводу использованного ими варианта теста. В работе McKinney et al. (1989, http://1.usa.gov/19odpv8) подсчитано, насколько часто в статьях из двух известных медицинских журналов указан вариант точного теста Фишера:

Журнал	**Вариант теста указан**	**Вариант теста не указан**	**Всего**
New England Journal of Medicine	1	8	**9**
Lancet	10	4	**14**
Всего	**11**	**12**	**23**

Как видим, число наблюдений в первой ячейке таблицы невелико, и критерий χ^2 применять здесь нельзя. Поэтому для анализа частот различных вариантов использования точного теста Фишера мы применим сам этот тест. По приведен-

ной выше формуле рассчитаем вероятность получить *наблюдаемый* набор чисел в ячейках таблицы сопряженности *при имеющихся значениях* маргинальных сумм:

$P = 9!14!11!12 / 23!(1!8!10!4!) = 0.00666.$

Теперь возьмем наименьшее из чисел в ячейках таблицы сопряженности и уменьшим его на 1. Числа в остальных ячейках при этом должны быть изменены так, чтобы маргинальные суммы по строкам и столбцам остались неизменными:

Журнал	Варинт теста указан	Вариант теста не указан	Всего
New England Journal of Medicine	0	9	**9**
Lancet	11	3	**14**
Всего	**11**	**12**	**23**

Вероятность новой таблицы составит:

$P = 9!14!11!12 / 23!(0!9!11!3!) = 0.00027.$

Поскольку наименьшее число в новой таблице равно 0, дальше его уменьшать невозможно. Соответственно, односторонний вариант точного теста Фишера даст

$p = 0.00666 + 0.00027 = 0.00693.$

Для получения двухстороннего варианта теста необходимо перебрать и все остальные возможные варианты заполнения таблицы сопряженности (при неизменных маргинальных суммах), каждый раз рассчитывая соответствующую вероятность. Для рассматриваемого примера мы получили бы еще 8 таблиц, для двух из которых вероятности не превысили бы вероятность исходного заполнения таблицы: $P = 0.00242$ и $P = 0.00007$. Таким образом, у нас есть еще три варианта «маловероятного» заполнения таблицы, кроме исходного. Просуммировав все соответствующие вероятности, получим p-значение для двухстороннего варианта точного критерия Фишера:

$p = 0.00666 + 0.00027 + 0.00242 + 0.00007 = 0.0094.$

Согласно полученным результатам, мы можем заключить, что различие частоты правильного использования точного критерия Фишера в журналах *New England Journal of Medicine* и *Lancet* статистически значимо.

В R для применения точного критерия Фишера служит функция **fisher.test**(). Подобно функции **chisq.test**(), **fisher.test**() принимает в качестве основного аргумента матрицу, соответствующую таблице сопряженности:

```
(X <- matrix(c(1, 10, 8, 4), ncol = 2))
     [,1] [,2]
[1,]    1    8
[2,]   10    4

fisher.test(X)
Fisher's Exact Test for Count Data
```

```
data:  X
p-value = 0.009423
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.001034782 0.656954980
sample estimates:
odds ratio
0.05851868
```

Полученное функцией **fisher.test**() *p*-значение оказалось практически таким же, что и рассчитанное «вручную» (p-value = 0.009423). Таким образом, у нас опять-таки есть все основания отклонить нулевую гипотезу о том, что частоты использования точного критерия Фишера в журналах *New England Journal of Medicine* и *Lancet* не различаются.

Разумеется, при больших значениях частот и размерностях таблицы число возможных перебираемых комбинаций достигает астрономических величин. К счастью, функция **fisher.test**() имеет возможность запустить процесс Монте-Карло и оценить *p*-значение для нулевой гипотезы по ограниченному подмножеству комбинаций. Можно отметить хорошую сходимость результатов этого подхода: в одном из наших расчетов при увеличении параметра числа комбинаций от 1000 до 100 000 полученное *p*-значение варьировало в пределах от 0.0002 до 0.0001. Дополнительную информацию по функции **fisher.test**() можно найти в ее справочном файле (команда ?fisher.test).

Критерий Мак-Немара

Рассмотренный выше критерий χ^2 для анализа таблиц сопряженности размером 2×2 применим только в отношении независимых наблюдений. Если же учет какого-либо бинарного признака выполняется, например, на одних и тех же испытуемых, то вместо критерия χ^2 следует использовать *критерий Мак-Немара*, названный по имени автора – американского психолога и статистика, описавшего этот критерий (McNemar, 1947, http://bit.ly/1Cjt2Ax).

Представим такую ситуацию: 15 испытуемым дали попробовать новый освежительный напиток, который скоро должен будет поступить в продажу. Испытуемых попросили ответить на вопрос, понравился ли им этот напиток – с вариантами ответа «да» или «нет». После этого *тем же* 15 участникам эксперимента показали рекламу, содержащую информацию о пользе и вкусовых качествах напитка. По завершении просмотра испытуемым снова дали выпить напиток и попросили ответить на ранее поставленный вопрос. Производитель напитка желает выяснить эффективность рекламы, изучив изменения в предпочтениях участников эксперимента после ее просмотра. Данные, полученные в ходе эксперимента, могли бы выглядеть так (пример заимствован с сайта http://bit.ly/1x6oeg5):

```
data <- read.table(header = TRUE, con <- textConnection("
 subject time result
       1  pre      0
       1 post      1
       2  pre      1
```

```
...
      15  pre      0
      15 post      1
"))
```

В таблице data столбец subject содержит идентификационные номера испытуемых, столбец time – метки, отражающие время проведения опроса (pre – до показа рекламы, post – после), а столбец result – ответы на вопрос (закодированы как 0 – «нет» и 1 – «да»). Данные умышленно приведены сначала в так называемом «длинном формате» с целью подчеркнуть еще раз, что *одни и те же* испытуемые были опрошены дважды и что их ответы могли измениться после просмотра рекламы.

Подобно критерию χ^2, критерий Мак-Немара работает с таблицей сопряженности размером 2×2. Для преобразования объекта data в такую таблицу необходимо выполнить несколько дополнительных операций. Сначала преобразуем data в таблицу «широкого формата», для чего используем возможности пакета reshape2 (Wickham, 2007, http://bit.ly/1xpysCC):

```
# Если пакет еще не установлен на вашем компьютере:
install.packages("reshape2")
# Преобразуем данные в "широкий формат":
data.wide <- dcast(data, subject ~ time, value.var="result")
data.wide
   subject post pre
1        1    1   0
2        2    1   1
3        3    1   0
...
14      14    0   0
15      15    1   0
```

Теперь эти данные можно легко свести в матрицу, соответствующую таблице сопряженности (использованная ниже функция **table**() входит в базовый набор функций R):

```
ct <- table(data.wide[, c("pre", "post")])
ct
   post
pre 0 1
  0 2 8
  1 1 4
```

Полученная таблица сопряженности отражает изменения предпочтений участников эксперимента до и после показа рекламы (сравните с критерием Стьюдента для парных выборок). Так, например, мы видим, что двоим испытуемым напиток не нравился как до, так и после показа рекламы (пересечение строки 0 и столбца 0); восьми испытуемым напиток не нравился до показа рекламы, но понравился после (пересечение строки 0 и столбца 1) и т. д. В общем виде таблицу сопряженности для критерия Мак-Немара мы можем представить так:

	Тест 2 (–)	**Тест 2 (+)**	**Всего**
Тест 1 (–)	*a*	*b*	*a* + *b*
Тест 1 (+)	*c*	*d*	*c* + *d*
Всего	*a* + *c*	*b* + *d*	*a* + *b* + *c* + *d*

Критерий Мак-Немара предназначен для проверки нулевой гипотезы о том, что маргинальные частоты строк и столбцов таблицы сопряженности не различаются, то есть

$$p(a) + p(b) = p(a) + p(c) \quad \text{и} \quad p(c) + p(d) = p(b) + p(d).$$

После сокращения получаем:

$$H_0: p(b) = p(c) \quad \text{и} \quad H_1: p(b) \neq p(c).$$

Критерий рассчитывается следующим образом:

$$Q = (b - c)^2/(b + c).$$

При больших объемах выборок (примерно 50 и выше) Q имеет распределение, близкое к распределению χ^2 с одной степенью свободы. Более точное приближение достигается при помощи *поправки Эдвардса* (Edwards, 1948, http://bit.ly/1ArwItV):

$$Q = (|b - c| - 1)^2/(b + c).$$

В R приведенные формулы реализованы в базовой функции **mcnemar.test**():

```
mcnemar.test(ct)
McNemar's Chi-squared test with continuity correction
data:  ct
McNemar's chi-squared = 4, df = 1, p-value = 0.0455
```

По умолчанию поправка Эдвардса применяется. При необходимости ее можно отключить, воспользовавшись аргументом correct:

```
mcnemar.test(ct, correct = FALSE)
McNemar's Chi-squared test
data:  ct
McNemar's chi-squared = 5.4444, df = 1, p-value = 0.01963
```

Как видим, в обоих случаях p-value < 0.05, что дает нам основание отклонить нулевую гипотезу об отсутствии эффекта рекламы – число испытуемых, которым напиток понравился после ее просмотра, изменилось.

Обратите внимание: хотя критерий Мак-Немара имеет дело с таблицей сопряженности 2×2, структура этой таблицы в корне отличается от таковой для критерия χ^2. В случае с χ^2 факт повторного учета дихотомического признака на тех же объектах не отражен:

```
table(data[, c("time", "result")])
      result
time    0  1
  post  3 12
  pre  10  5
```

Критерий Кохрана-Мантеля-Хензеля для таблиц сопряженности размером 2×2×K

Как правило, таблицу сопряженности 2×2 получают в ходе единичного эксперимента, направленного на изучение распределения того или иного бинарного признака в двух группах объектов (например, в экспериментальной и контрольной группах). Но что, если один и тот же эксперимент повторен несколько раз?

Например, в ходе клинических испытаний часто эффективность нового препарата исследуют по одинаковой схеме в разных медицинских учреждениях. В результате получают набор из *K* таблиц сопряженности размером 2×2, где *K* – это количество участвовавших в исследовании медицинских центров. По разным причинам можно ожидать, что результаты эксперимента будут несколько варьировать от центра к центру. Соответственно, «медицинский центр» становится важной ковариатой, действие которой мы должны учесть при установлении эффективности испытываемого нового препарата. Одним из статистических методов, позволяющих это сделать, является *критерий Кохрана-Мантеля-Хензеля* («*Cochran-Mantel-Haenszel test*», или просто «*CMH test*» – по фамилии авторов Cochran (1954, http://bit.ly/1xpAliJ) и Mantel & Haenszel (1959, http://bit.ly/1G8y4NM)). Насколько нам известно, устоявшегося перевода фамилии последнего автора в русскоязычной литературе нет – кроме приведенного, встречаются, например, названия «критерий Кохрана-Мантеля-Гензеля» и «критерий Кохрана-Мантеля-Хенселя».

Принцип работы CMH-критерия и соответствующую функцию языка R для его вычисления мы опишем на примере из книги Agresti (2002; http://bit.ly/1BIlpS9). Имеются результаты исследования эффективности действия мази, содержащей новое действующее вещество в отношении определенного инфекционного заболевания. Дизайн эксперимента заключался в следующем: две группы испытуемых подвергались лечению при помощи нового препарата и плацебо. В конце эксперимента учтено количество случаев успешного и неуспешного лечения в обеих группах. Эксперимент был повторен по описанной схеме в 8 медицинских центрах, при разном общем количестве испытуемых и разном их количестве в отдельных экспериментальных группах в каждом центре. Данные представлены в таблице ниже:

Медицинский центр	Экспериментальная группа	Исход лечения	
		Успешное	Неуспешное
1	Препарат	11	25
	Контроль	10	27
2	Препарат	16	4
	Контроль	22	10
3	Препарат	14	5
	Контроль	7	12
4	Препарат	2	14
	Контроль	1	16
5	Препарат	6	11
	Контроль	0	12

Медицинский центр	Экспериментальная группа	Исход лечения	
		Успешное	Неуспешное
6	Препарат	1	10
	Контроль	0	10
7	Препарат	1	4
	Контроль	1	8
8	Препарат	4	2
	Контроль	6	1

Обозначим ячейки каждой k-й таблицы сопряженности следующим образом:

Экспериментальная группа	Успешное лечение	Неуспешное лечение	Всего
Препарат	n_{k11}	n_{k12}	$n_{k1\bullet}$
Контроль	n_{k21}	n_{k22}	$n_{k2\bullet}$
Всего	$n_{k\bullet 1}$	$n_{k\bullet 2}$	$n_{k\bullet\bullet}$

Тогда формулу CMH-критерия можно записать так:

$$\text{CMH} = \left\{ \left| \sum_k (n_{k11} - \frac{n_{k\bullet 1} n_{k1\bullet}}{n_{k\bullet\bullet}}) \right| - 0.5 \right\}^2 / \sum_k \frac{n_{k\bullet 1} n_{k1\bullet} n_{k\bullet 2} n_{k2\bullet}}{n_{k\bullet\bullet}^2 (n_{k\bullet\bullet} - 1)}.$$

Как видно из этой формулы, CMH-критерий очень схож с критерием хи-квадрат. В частности, в числителе формулы мы видим сумму квадратов разностей между наблюдаемой частотой в ячейке n_{k11} и ее ожидаемой частотой при верной нулевой гипотезе. Знаменатель содержит оценку дисперсии этих квадратов разностей. В связи с таким сходством CMH-критерий часто называют еще критерием *хи-квадрат Мантеля-Хензеля*.

Нулевая гипотеза, проверяемая при помощи CMH-критерия, заключается в том, что между двумя анализируемыми качественными признаками (в нашем случае «экспериментальная группа» и «исход лечения») нет никакой связи. С формальной точки зрения, тестируется гипотеза о том, что *отношение шансов* в каждой из K таблиц сопряженности равно 1 (про отношение шансов можно почитать, например, на сайте http://bit.ly/1MCOP7D).

В базовой версии R для выполнения CMH-теста имеется функция **mantelhaen.test()**. Данные подаются на нее в виде трехмерного массива, состоящего из 2×2×K элементов. В нашем случае данные можно ввести следующим образом:

```
drug <-
  array(c(11, 10, 25, 27,
          16, 22, 4, 10,
          14, 7, 5, 12,
          2, 1, 14, 16,
          6, 0, 11, 12,
          1, 0, 10, 10,
          1, 1, 4, 8,
```

```
          4, 6, 2, 1),
        dim = c(2, 2, 8),
        dimnames = list(
          Group = c("Drug", "Control"),
          Response = c("Success", "Failure"),
          Center = c("1", "2", "3", "4", "5", "6", "7", "8")))

drug

, , Center = 1
         Response
Group     Success Failure
  Drug         11      25
  Control      10      27
, , Center = 2
         Response
Group     Success Failure
  Drug         16       4
  Control      22      10
…
, , Center = 8
         Response
Group     Success Failure
  Drug          4       2
  Control       6       1
```

Подадим массив `drug` на функцию **`mantelhaen.test`**():

```
mantelhaen.test(drug)
Mantel-Haenszel chi-squared test with continuity correction
data:  drug
Mantel-Haenszel X-squared = 5.6716, df = 1, p-value = 0.01724
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
 1.177590 3.869174
sample estimates:
common odds ratio
         2.134549
```

Как видим, исход лечения оказался статистически значимо связанным с тем, получали ли испытуемые новый препарат (`p-value = 0.01724`). Иными словами, мы можем принять альтернативную гипотезу, которая заключается в том, что истинное общее отношение шансов (для всех таблиц) не равно 1 (`true common odds ratio is not equal to 1`). Это подтверждается также рассчитанным программой 95%-ным доверительным интервалом для отношения шансов, который не включает 1 (`95 percent confidence interval: 1.177590 - 3.869174`).

Графически мы можем изобразить имеющиеся данные следующим образом (рис. 64):

```
library(reshape) # для функции melt()
drug.df <- data.frame(melt(drug,
                           id = c("Center", "Group", "Response")))
library(ggplot2)
p <- ggplot(data = drug.df,
            aes(x = Center, y = value, fill = Response)) +
            ylab("Fraction")
p + geom_bar(stat = "identity", position = "fill") +
            facet_grid(Group~.)
```

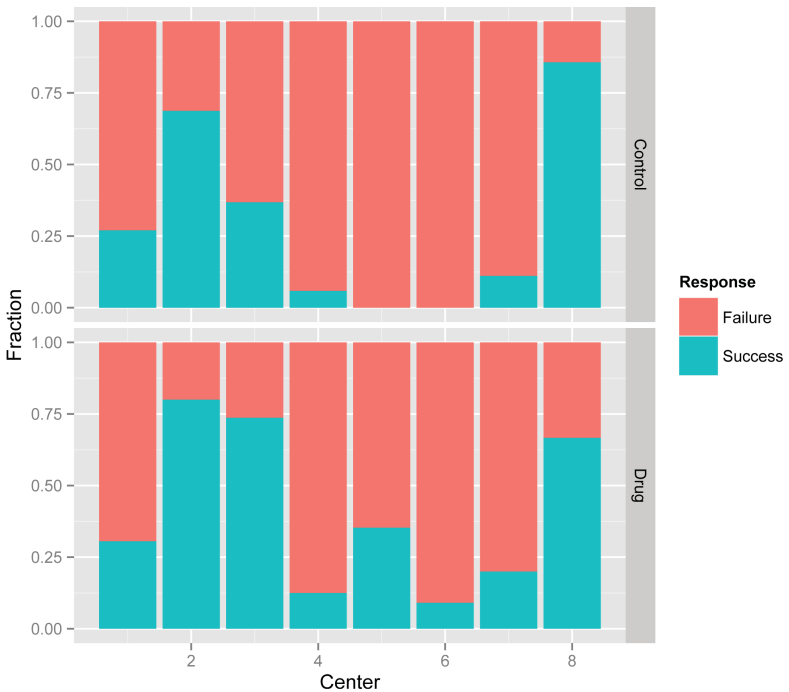


Рисунок 64

По рисунку видно, что частота успешных исходов лечения в целом выше у испытуемых, которые получали новый препарат.

При интерпретации результатов CMH-теста, выполняемого при помощи функции `mantelhaen.test()`, следует иметь в виду следующее:

- *Выполняется ли поправка на непрерывность?* По умолчанию используется поправка в виде вычитания значения 0.5 в числителе – см. выше формулу критерия. При необходимости ее отключить воспользуйтесь аргументом `correction = FALSE` (получаемое в результате этого *p*-значение может измениться, что в некоторых случаях может привести к противоположному заключению относительно справедливости нулевой гипотезы).

- *Какой вариант гипотезы проверяется?* Возможные варианты: `altenative = "two.sided"` (двухсторонний критерий; по умолчанию), `alternative = "greater"` (больше) или `alternative = "less"` (меньше).
- *Однородны ли отношения шансов в каждой из анализируемых таблиц сопряженности?* Функция **`mantelhaen.test`**() автоматически «предполагает», что это условие выполняется. По поводу необходимости проверки однородности отношений шансов однозначного мнения нет (подробнее см. http://bit.ly/1G8BsIL). Тем не менее функции **`cmh_test`**() из пакета `coin` и **`epi.2by2`**() из пакета `epiR` позволяют выполнить специально предназначенный для этого тест Бреслоу-Дэя (Breslow & Day, 1980; http://bit.ly/19okxHV).
- *Наблюдается ли сходный отклик на экспериментальное воздействие во всех имеющихся таблицах сопряженности*? Мощность CMH-критерия значительно снижается, когда имеются разнонаправленные отклики в анализируемых таблицах сопряженности (например, если бы в нашем случае в части таблиц наблюдалось выраженное снижение, а в части таблиц – выраженное повышение частоты случаев успешного лечения среди испытуемых, получавших новый препарат). Это, соответственно, затрудняет интерпретацию высоких *p*-значений (например, > 0.05): вызвано ли наблюдаемое высокое *p*-значение отсутствием эффекта или это результат наличия разнонаправленных откликов?

5.9. Оценка статистической мощности при сравнении частот

В этом разделе мы продолжим тему оценки мощности статистических критериев и покажем, как при помощи R можно выполнить анализ мощности при сравнении частот. Представим, что мы входим в команду кандидата в президенты страны *X*. Согласно результатам опросов, выполненных командой кандидата-соперника, выяснилось, что популярность нашего кандидата у городских жителей выше, чем у жителей села (28% против 20% среди каждых 100 опрошенных респондентов каждой категории). Безусловно, это важная информация, которая может помочь в планировании агитационных мероприятий (например, направление дополнительных ресурсов на агитацию среди сельских жителей). Однако стоит ли доверять информации из лагеря соперника?

Перед нами поставлена задача спланировать независимое исследование для выяснения лояльности городских и сельских жителей в отношении нашего кандидата. Имеющиеся на проведение исследования ресурсы (деньги и прочее) ограничены, и поэтому необходимо определиться с оптимальным количеством респондентов в предстоящем опросе.

Начнем с анализа данных, полученных командой соперника. Как уже было показано в разделе 5.7, две доли мы можем сравнить при помощи критерия «хи-квадрат»:

```
votes <- matrix(c(28, 72, 20, 80), ncol = 2, byrow = T)
votes
     [,1] [,2]
[1,]   28   72
[2,]   20   80

chisq.test(votes)
Pearson's Chi-squared test with Yates' continuity correction
data:  votes
X-squared = 1.3432, df = 1, p-value = 0.2465
```

Выполненный тест хи-квадрат позволяет заключить, что жители города и села статистически не различаются по своим предпочтениям в отношении нашего кандидата (`p-value = 0.2465`).

Казалось бы, это хорошие новости, так как нашей команде, по-видимому, нет необходимости предпринимать дополнительные усилия для обеспечения более высокой лояльности среди сельских жителей. Но что, если этот вывод неверен просто в силу недостаточно большого числа опрошенных? Иными словами, мощность критерия недостаточно велика, и мы совершили ошибку второго рода – не отклонили неверную нулевую гипотезу об отсутствии разницы между опрошенными категориями жителей?

Хотя в базовой комплектации R имеется специальная функция для оценки мощности при сравнении двух долей **`power.prop.test`**`()`, мы воспользуемся возможностями специализированного пакета `pwr`. Реализованные в этом пакете функции следуют формулам из известной книги Cohen (1988, http://bit.ly/19olBLZ). Для анализа мощности критерия хи-квадрат служит функция **`pwr.chisq.test`**`()`, имеющая следующие аргументы:

- `w` – размер эффекта;
- `N` – общее число наблюдений;
- `df` – число степеней свободы;
- `sig.level` – уровень значимости;
- `power` – мощность критерия.

Следует пояснить, что под размером эффекта `w` для χ^2 подразумевается величина, рассчитываемая по формуле $w = \Sigma(p_{0i} - p_{1i})^2/p_{0i}$, где p_{0i} – ожидаемая вероятность для i-й ячейки таблицы сопряженности при верной нулевой гипотезе об отсутствии эффекта, а p_{1i} – наблюдаемая вероятность. Иными словами, `w` измеряет степень отклонения наблюдаемых частот в таблице сопряженности от тех, которые можно было бы ожидать при отсутствии эффекта исследуемого фактора.

Рассчитаем мощность приведенного выше теста χ^2. Для удобства сначала сохраним результаты теста в виде самостоятельного объекта (назовем его `res`):

```
res <- chisq.test(votes)

res
Pearson's Chi-squared test with Yates' continuity correction
data:  votes
X-squared = 1.3432, df = 1, p-value = 0.2465
```

Объект res представляет собой список из нескольких компонентов, включая таблицы с наблюдаемыми и ожидаемыми частотами. Мы можем легко извлечь эти таблицы и рассчитать соответствующие вероятности:

```
(obs <- res$observed)
     [,1] [,2]
[1,]   28   72
[2,]   20   80

(exptd <- res$expected)
     [,1] [,2]
[1,]   24   76
[2,]   24   76

(obs <- obs/200) # здесь и ниже 200 - общее число опрошенных
     [,1] [,2]
[1,] 0.14 0.36
[2,] 0.10 0.40

(exptd <- exp/200)
     [,1] [,2]
[1,] 0.12 0.38
[2,] 0.12 0.38
```

Используя приведенную выше формулу, рассчитаем размер эффекта:

```
sqrt(sum((exptd-obs)^2/exptd))
[1] 0.09365858
```

Приведенные вычисления размера эффекта можно было значительно сократить, воспользовавшись готовой функцией из пакета pwr – **ES.w2**(), на вход которой в виде матрицы подается таблица сопряженности с наблюдаемыми вероятностями:

```
library(pwr)
ES.w2(obs)
[1] 0.09365858
```

Много это или мало – 0.094? Показатель w изменяется от 0 до 1, и чем ближе его значение к 1, тем сильнее эффект исследуемого фактора. В рассматриваемом случае размер эффекта весьма невысок – место проживания респондентов очень слабо связано с их лояльностью в отношении нашего кандидата. При таком размере эффекта и при общем числе опрошенных респондентов, равном 200, мощность выполненного выше теста χ^2 составляет

```
pwr.chisq.test(w = ES.w2(obs), df = 1, N = 200)
     Chi squared power calculation
              w = 0.09365858
              N = 200
             df = 1
```

```
      sig.level = 0.05
          power = 0.2630843
NOTE: N is the number of observations
```

Рассчитанная мощность (~26%) гораздо ниже условно принятого порога в 80%. Таким образом, команда соперника имеет мало оснований утверждать, что наш кандидат менее популярен среди сельских жителей. Скорее всего, число опрошенных ими респондентов было недостаточным для такого вывода при имеющемся небольшом размере эффекта.

Приведенная выше величина мощности критерия χ^2 (26%) называется *достигнутой*, или *наблюдаемой*, *мощностью* («*observed power*»). Следует подчеркнуть, что результат такого ретроспективного расчета мощности статистического теста (то есть расчета, выполненного после проведения исследования) следует интерпретировать с очень большой осторожностью (Thomas, 1997, http://bit.ly/19omxQu). В частности, логически не имеют смысла утверждения вроде следующего: «*Поскольку нулевая гипотеза не была отвергнута и наблюдаемая мощность теста высока, имеющиеся данные свидетельствуют в пользу нулевой гипотезы*». Дело здесь в том, что между мощностью теста и уровнем значимости α по определению имеется тесная функциональная связь: чем выше α, тем ниже мощность (рис. 65). Соответственно, рассчитанные в ходе теста высокие *p*-значения *всегда* сопровождаются низкой мощностью, что делает сам ретроспективный расчет мощности бессмысленным (Hoenig & Heisey, 2001, http://bit.ly/1x6sOuz).

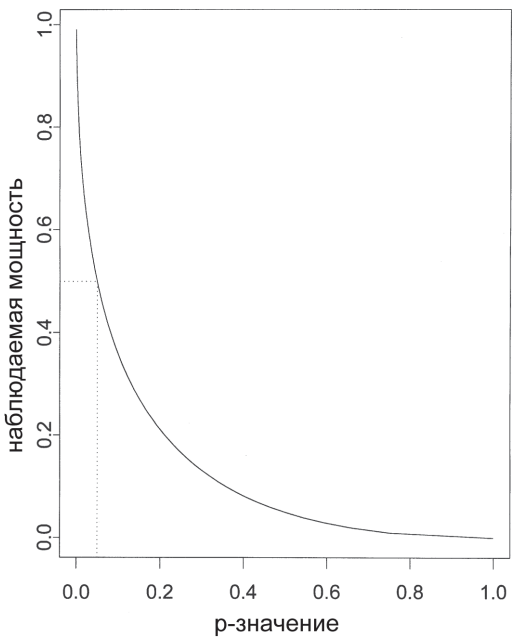


Рисунок 65 (по: Hoenig & Heisey, 2001)

Оценка мощности статистического теста должна выполняться *на стадии планирования исследования*. Выполним такую оценку для нашего независимого опроса общественного мнения. Поскольку имеющиеся у нас ресурсы ограничены, решено, что результаты этого исследования будут приняты во внимание при разработке агитационных мероприятий только в том случае, если выявленный размер эффекта окажется достаточно большим и если мощность критерия χ^2 составит не менее 80% при уровне значимости 0.05. Однако что значит «достаточно большой» эффект? Согласно классификации, предложенной Коэном, $w = 0.10$ следует считать небольшим эффектом, $w = 0.3$ – умеренно большим, а $w = 0.5$ и выше – большим эффектом. Примем, что порог $w = 0.15$ является достаточно значительным эффектом, чтобы обратить на него внимание в нашей ситуации (должность президента на кону, как-никак!). Тогда минимальное число подлежащих опросу респондентов составит:

```
pwr.chisq.test(w = 0.15, N = NULL, df = 1,
               sig.level = 0.05, power = 0.8)
     Chi squared power calculation
              w = 0.15
              N = 348.8382
             df = 1
      sig.level = 0.05
          power = 0.8
NOTE: N is the number of observations
```

Обратите внимание на аргумент `N` – ему было присвоено значение `NULL`, поскольку рассчитывается именно число наблюдений.

Таким образом, всего необходимо будет опросить около 350 случайным образом выбранных респондентов (примерно половина из них будет из числа городских жителей, а другая половина – из числа сельских).

В завершение следует упомянуть еще три функции из пакета `pwr`, созданные для анализа статистической мощности при работе с долями:

- **`pwr.2p.test`**() – анализ мощности для двух долей, рассчитанных по выборкам одинакового размера;
- **`pwr.2p2n.test`**() – анализ мощности для двух долей, рассчитанных по выборкам разного размера;
- **`pwr.p.test`**() – анализ мощности для одной доли (например, при сравнении ее с каким-либо ожидаемым значением).

Мощность теста можно оценить также путем многократных имитаций. Приведенная ниже функция рассчитывает мощность по четырем параметрам – объему выборки в первой группе, объему выборки во второй группе, ожидаемой частоте в первой группе и ожидаемой частоте во второй группе. Генерируется 10 000 таблиц размером 2×2, и на их основе рассчитывается вероятность отвергнуть нулевую гипотезу.

```
prop.power <- function(n1, n2, p1, p2) {
twobytwo <- matrix(NA, nrow = 10000, ncol = 4)
```

```
twobytwo[, 1] <- rbinom(n = 10000, size = n1, prob = p1)
twobytwo[, 2] <- n1 - twobytwo[, 1]
twobytwo[, 3] <- rbinom(n = 10000, size = n2, prob = p2)
twobytwo[, 4] <- n1 - twobytwo[, 3]
p <- rep(NA, 10000)
chisq.test.v <- function(x) as.numeric(chisq.test(
                    matrix(x, ncol = 2), correct=FALSE)[3])
p <- apply(twobytwo, 1, chisq.test.v)
power <- sum(ifelse(p < 0.05, 1, 0))/10000
  return(power)
}
```

Эта функция дает результат, сходный с приведенной выше ретроспективной оценкой мощности:

```
prop.power(100, 100, 0.28, 0.20)
[1] 0.2649
```

Глава 6

Дисперсионный анализ

6.1. Протокол разведочного анализа данных

За последние несколько десятилетий арсенал прикладной статистики значительно пополнился за счет развития новых методов, таких, например, как обобщенные линейные модели (Generalized Linear Models), обобщенные аддитивные модели (Generalized Additive Models), модели со смешанными эффектами (Mixed Effects Models), деревья принятия решений (Regression and Classification Trees), анализ выживаемости (Survival analysis), многомерное шкалирование (Multidimensional scaling) и др. Более того, появление быстрых современных компьютеров и свободного программного обеспечения (вроде R) сделало все эти методы, требующие значительных вычислительных ресурсов, доступными практически для каждого исследователя. Однако такая доступность еще больше обостряет хорошо известную проблему всех статистических методов, которую на английском языке часто описывают как «*garbage in, garbage out*», то есть «мусор на входе – мусор на выходе». Речь здесь идет о следующем: чудес не бывает, и если мы не будем уделять должного внимания тому, как тот или иной метод работает и какие требования предъявляет к анализируемым данным, то получаемые с его помощью результаты нельзя будет воспринимать всерьез. Это утверждение в полной мере справедливо как для дисперсионного анализа, которому посвящена данная глава, так и для методов, рассматриваемых в последующих главах. Поэтому каждый раз исследователю следует начинать свою работу с тщательного ознакомления со свойствами полученных данных и проверки необходимых условий применимости соответствующих статистических методов. Этот начальный этап называют *разведочным анализом данных* (РДА) («*exploratory data analysis*»).

В литературе по статистике можно найти немало рекомендаций по выполнению РДА. В концентрированной форме эти рекомендации недавно были сведены в единый протокол в широко цитируемой работе Zuur et al. (2010, http://bit.ly/1E0Ub8J). Несмотря на то что статья написана для биологов (в частности, для экологов), изложенные в ней принципы будут верны и в отношении других научных дисциплин. Прежде чем перейти к детальному описанию методик построения и интерпретации конкретных статистических моделей, мы приведем выдержки из Zuur et al. (2010) и опишем предложенный авторами РДА-протокол. Подобно тому, как это сделано в оригинальной статье, описание отдельных шагов протокола будет сопровождаться краткими рекомендациями по использованию соответствующих функций и пакетов системы R.

Предлагаемый протокол включает следующие основные элементы:

1) формулировка исследовательской гипотезы и выполнение эксперимента для сбора необходимых данных;
2) разведочный анализ данных:
   - выявление точек-выбросов;
   - проверка однородности дисперсий;
   - проверка нормальности распределения данных;
   - выявление избыточного количества нулевых значений;
   - выявление коллинеарных переменных;
   - выявление характера связи между анализируемыми переменными;
   - выявление взаимодействий между переменными-предикторами;
   - выявление пространственно-временных корреляций между значениями зависимой переменной;
3) применение статистического метода (модели) в соответствии с поставленной задачей и свойствами данных.

Zuur et al. (2010) подчеркивают, что РДА наиболее эффективен при использовании разнообразных графических средств, поскольку графики часто позволяют лучше понять структуру и свойства анализируемых данных, чем формальные статистические тесты.

Выявление точек-выбросов

С рекомендаций по исключению из анализа выскакивающих наблюдений («экстремальных точек», «выбросов», «засорений») начинаются многие руководства по прикладной статистике. Некоторые способы идентификации и «борьбы» с выбросами описаны нами в разделе 4.3.

Очень часто исследователи забывают, что большинство таких процедур предназначено для отбрасывания лишь единичных экстремальных значений. Тем не менее можно встретить работы, в которых, скажем, из шести наблюдений отбрасываются три. Это совершенно недопустимо: отбрасывание аномальных значений должно быть основано на очень серьезных изначальных предположениях. Наличие выбросов часто может означать, что исходное распределение, например, не является простым нормальным, а смесью разных распределений. Для проверки такого предположения потребуется изучение дополнительных выборок большего объема.

Чувствительность разных статистических методов к наличию выбросов в данных неодинакова. Так, при использовании обобщенной линейной модели для анализа зависимой переменной, распределенной по закону Пуассона (например, количество случаев какого-либо заболевания в разных городах), наличие выбросов может вызвать избыточную дисперсию, что сделает модель неприменимой. В то же время при использовании многомерного шкалирования, основанного на индексе Жаккара, все исходные данные переводятся в номинальную шкалу с двумя значениями (1/0), и наличие выбросов никак не сказывается на результатах анализа. Исследователь должен четко понимать эти различия, обусловленные разными методами.

Проверка однородности групповых дисперсий

С проблемой выбросов тесно связан второй пункт протокола разведочного анализа – проверка условия однородности дисперсии («*homogeneity of variance*»). Однородность групповых дисперсий постулируется как важное условие применимости дисперсионного анализа и других линейных моделей регрессионного типа, а также ряда методов многомерной статистики (например, дискриминантного анализа).

На рис. 66 приведены категоризованные диаграммы размахов для значений интенсивности потребления пищи канадским веретенником – птицы из семейства бекасовых (Zuur et al., 2010).

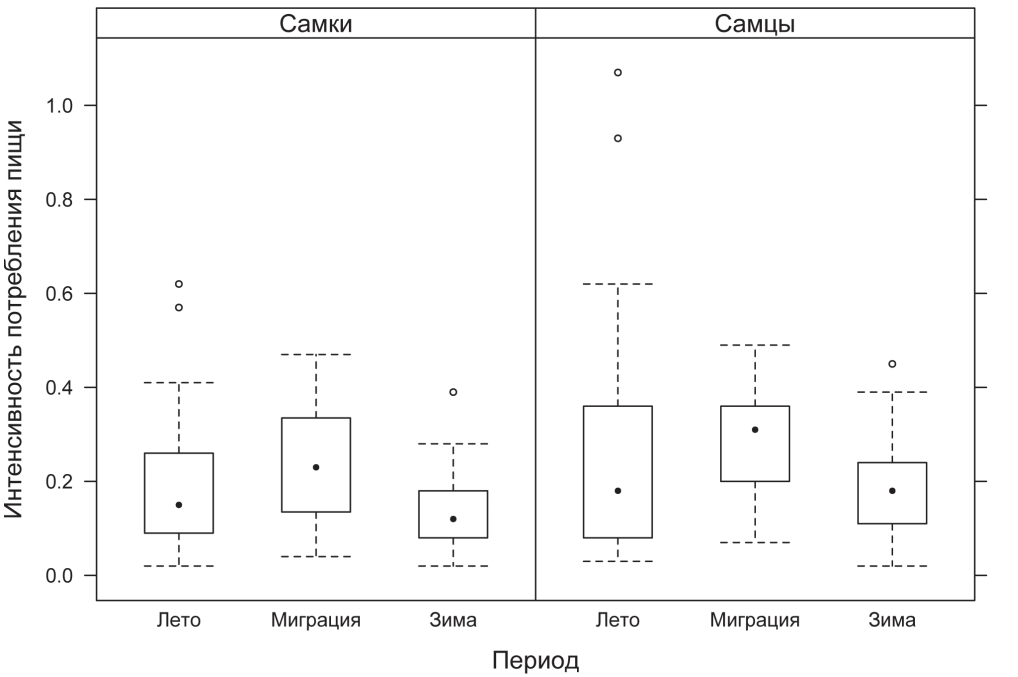


Рисунок 66

Если бы стояла задача применить параметрический дисперсионный анализ для установления эффектов пола и периода наблюдений на интенсивность потребления пищи веретенником (а также взаимодействия между этими двумя факторами), то должны были выполняться следующие условия:

- дисперсии не различаются у самцов и самок;
- дисперсии не различаются между периодами наблюдений;
- дисперсии не различаются между периодами наблюдения у каждого из полов.

Как можно увидеть из графика, разброс значений интенсивности потребления пищи у самцов низок зимой и несколько повышается в летний период, но в целом

в приведенном примере различия групповых дисперсий невелики и не потребуют особых действий со стороны исследователя. Это можно подтвердить не только субъективными визуальными ощущениями, но и с использованием функций для формальной проверки нулевой гипотезы о равенстве дисперсий (см. раздел 5.4).

Конечно, такая ситуация будет наблюдаться далеко не всегда. Если отношение между самой высокой и самой низкой групповыми дисперсиями превышает 4, то оценки параметров моделей, основанные на методе наименьших квадратов, будут значительно смещенными (Fox, 2008). При построении регрессионных моделей проверка условия однородности дисперсии чаще всего выполняется путем графического анализа распределения остатков: по оси ординат откладывают значения остатков, а по оси абсцисс – предсказываемые моделью средние значения зависимой переменной. Точки на таком графике должны располагаться случайным образом, не формируя какого-либо четкого паттерна. При наличии в модели номинальных переменных (факторов) для остатков строят категоризованные диаграммы размахов вроде той, что приведена на рис. 66 (дисперсии остатков в отдельных группах, соответствующих уровням фактора, должны быть сходными).

При выявлении существенной неоднородности дисперсии возможны два решения: определенное преобразование исходных данных (например, логарифмирование) или использование моделей, основанных на обобщенном методе наименьших квадратов («*generalized least squares*») и допускающих неоднородность дисперсии.

Проверка на нормальность распределения

Подчиняются ли анализируемые количественные переменные закону нормального распределения вероятностей? Очень многие статистические методы, включая классический дисперсионный анализ, предполагают положительный ответ на этот вопрос, и поэтому проверка исследуемых переменных на нормальность распределения является важной составной частью РДА.

Проверяя условие нормальности распределения данных, необходимо, однако, хорошо представлять себе, в каких случаях его выполнение является критическим для применения конкретного метода. Так, например, *метод главных компонент* (Principle Components Analysis, PCA; см. главу 7) не требует, чтобы данные были распределены нормально (Joliffe, 2002, http://bit.ly/1BJlPaV). *Линейная регрессия*, хотя и предполагает нормальность распределения зависимой переменной, является достаточно устойчивым методом при незначительных отклонениях от этого условия (Fitzmaurice et al., 2004, http://amzn.to/1CjGfb8). В то же время для успешного применения *дискриминантного анализа* нормальность распределения признаков в каждой группе классифицируемых объектов – условие обязательное (Huberty, 1994, http://bit.ly/1BmzY90).

Определение закона распределения вероятностей, которому подчиняются эмпирические данные, – это отдельная большая тема, и мы ее уже обсуждали в разделах 4.7–4.8. Отличную статью на русском языке с примерами подгонки распределений средствами R можно найти на сайте http://bit.ly/1GT4TSA. Советуем

также обратить внимание на пакет fitdistrplus и на руководство по работе с ним (http://bit.ly/1BmAqnQ).

Выявление избыточного числа нулевых значений

Важным этапом разведочного анализа данных является обнаружение «избыточного» числа нулевых значений. Данные, включающие большое количество нулей, традиционны, например, в экологических исследованиях, когда исследователь оценивает численность популяции того или иного вида. Особи одной популяции редко распределены в пространстве равномерно – чаще они образуют скопления в силу, например, неоднородности распределения необходимых им факторов среды или для повышения выживаемости. Обследуя территорию, на которой обитает популяция с таким типом пространственного распределения, исследователь в большинстве случаев не встретит ни одной особи изучаемого вида, реже ему попадутся единичные экземпляры, и лишь иногда – большие скопления особей.

Для выявления характера распределения данных такого рода («count data» – «*счетные данные*») бывает достаточно построить гистограмму или полигон частот. Часто счетные данные подчиняются отрицательному биномиальному распределению или распределению Пуассона, и поэтому для их анализа не подходят обычные модели линейной регрессии, предполагающие нормальное распределение остатков. Здесь следует использовать *обобщенные линейные модели* (см. главу 8) с соответствующим распределением остатков – см. справочный файл по функции **glm**().

Однако во многих случаях количество нулей оказывается намного большим, чем это предсказывает отрицательное биномиальное распределение или распределение Пуассона. В таких данных имеет место избыточная дисперсия, с которой обычным обобщенным линейным моделям «не справиться». Выходом здесь могут стать так называемые обобщенные линейные модели для данных с *избыточным количеством нулей* (Zero Inflated Generalized Linear Models). Рекомендуем в этой связи ознакомиться с часто цитируемой работой Zeileis et al. (2008, http://bit.ly/1BHD235), в которой рассмотрены возможности R для регрессионного анализа счетных данных, включая Zero Inflated GLM. Много подробных примеров из экологии (с сопутствующим R-кодом) можно также найти в замечательной книге Zuur et al. (2009).

Выявление коллинеарности

Когда цель анализа заключается в нахождении переменных (*предикторов*), связанных со значениями *зависимой переменной*, важным этапом разведочного анализа данных является обнаружение коллинеарности. Под *коллинеарностью* («*collinearity*») понимают наличие линейной зависимости между двумя предикторами. В задачах с несколькими предикторами (например, при выполнении *множественного регрессионного анализа*) говорят также о *мультиколлинеарности* («*multicollinearity*»), то есть наличии линейной зависимости сразу между несколькими переменными.

Наличие *мультиколлинеарности* приводит к проблемам при вычислении регрессионных коэффициентов: оценки параметров модели оказываются неустойчивыми, и становится трудно анализировать вклад каждого отдельного фактора в прогнозируемую величину. Как результат исследователь может столкнуться с «парадоксальной» ситуацией, когда, например, все коэффициенты множественной регрессионной модели статистически незначимы, тогда как сама модель оказывается значимой (то есть проверяемая при помощи *F*-теста гипотеза о равенстве *всех* коэффициентов нулю отвергается).

Традиционным способом оценки мультиколинеарности является анализ корреляционной матрицы. Хорошим способом повысить наглядность отображения корреляции между переменными является использование раскрашенных матриц, создаваемых функцией `corrplot()` из одноименного пакета (рис. 67). Рассмотрим пример с изучением потребности во сне у животных, использованный в разделе 4.4 (в приведенном ниже коде предполагается, что таблица данных с заполненными пропусками `sleep_imp.Rdata` была сохранена для дальнейшего использования; эта таблица также доступна для скачивания на сайте книги):

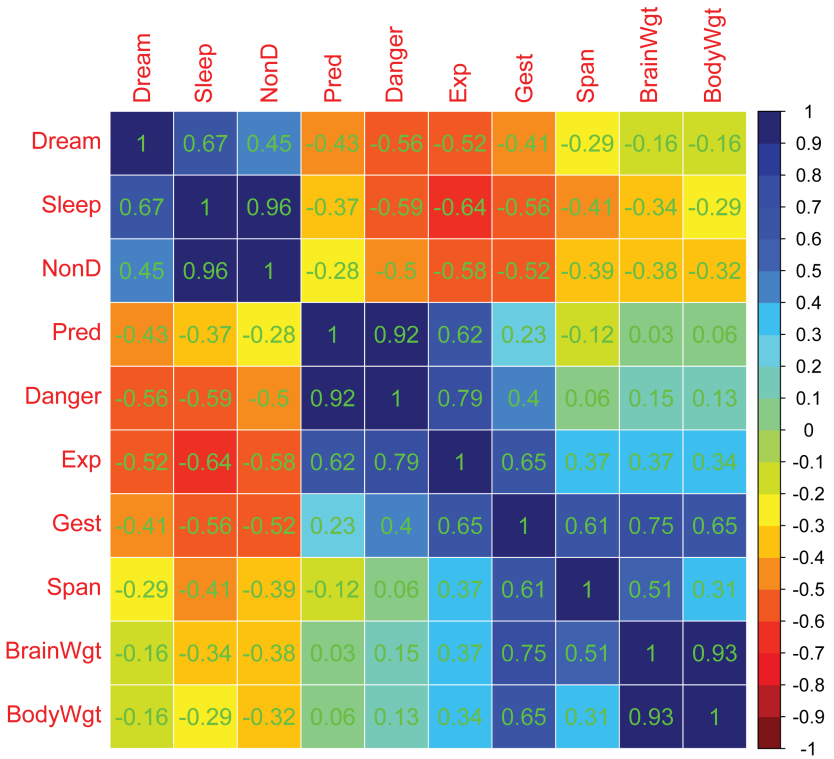


Рисунок 67

```
load(file = "sleep_imp.Rdata")
M <- cor(sleep_imp3)
library(corrplot)
col4 <- colorRampPalette(c("#7F0000", "red", "#FF7F00",
   "yellow", "#7FFF7F", "cyan", "#007FFF", "blue", "#00007F"))
corrplot(M, method = "color", col = col4(20), cl.length = 21,
         order = "AOE", addCoef.col = "green")
```

Нетрудно выделить на рисунке два блока предикторов: таксономические (`BodyWgt + BrainWgt`) и экологические (`Pred + Danger`) показатели, которые высоко коррелируют между собой.

Другим способом оценить степень мультиколлинеарности m-мерного комплекса переменных является вычисление *фактора инфляции дисперсии* («*variance inflation factor*», *VIF*), который для каждого j-го предиктора из m тем выше, чем теснее его линейная связь со всеми остальными $(m - 1)$ независимыми переменными. Значения *VIF* находят следующим образом:

- строится регрессионная модель зависимости переменной X_j от всех остальных предикторов; например, если $j = 1$, то подгоняется модель $X_j = \beta_0 + + \beta_2 X_2 + \beta_3 X_3 + ... + \beta_m X_m + \varepsilon$;
- рассчитывается фактор инфляции дисперсии $VIF(\beta_j) = 1/(1 - R_j^2)$, где R_j^2 – коэффициент детерминации для полученной на предыдущем шаге регрессионной модели;
- предыдущие два шага повторяются последовательно для всех переменных $j = 1, 2, ..., m$.

Принято считать, что значения VIF_j от 1 до 2 (что соответствует R_j^2 от 0 до 0.5) означают отсутствие проблемы мультиколлинеарности для X_j, то есть добавление в модель или удаление из нее любых других независимых переменных не изменяет ни самой оценки коэффициента β_j, ни критериев его статистической значимости. Если $VIF_j = 4$, то ошибка выборочной оценки параметра β_j возрастает в $(4)^{0.5} = 2$ раза (по сравнению с ситуацией, когда мультиколлинеарность отсутствует). Согласно часто используемому эмпирическому правилу, $VIF_j \geq 10$ указывает на наличие четко выраженной проблемы мультиколлинеарности.

Функции для автоматического расчета *VIF* и выполнения перечисленных выше шагов реализованы в нескольких пакетах для R. Одним из примеров таких функций является **`vif`**`()` из пакета `car`:

```
library(car)
vif(lm(Sleep ~ BodyWgt + BrainWgt + Span + Gest + Pred + Exp + Danger,
    data = sleep_imp3))
BodyWgt BrainWgt   Span   Gest   Pred    Exp  Danger
12.363    15.895  2.659  4.124  8.211  4.921  12.879
```

Полученные значения *VIF* в целом указывают на высокую мультиколлинеарность имеющихся предикторов.

Часто на практике используется итерационная процедура построения моделей на основе фактора инфляции дисперсии: значения VIF_j сравниваются с некото-

рым пороговым значением и предиктор с максимальным *VIF*, превышающим это пороговое значение, исключается из модели. Процедура повторяется, пока в модели не останутся предикторы с низкими *VIF*. Способы селекции информативного комплекса предикторов будут более подробно обсуждаться нами в главе 7. Пока же отметим, что к любому автоматическому подбору предикторов стоит относиться критически, поскольку «с водой легко выплеснуть и ребенка». Так, например, нет единого мнения в отношении того, какое значение *VIF* следует считать пороговым (например, упомянутое выше *VIF* = 10 или используемое некоторыми исследователями *VIF* = 5).

Выявление формы связи между переменными

Существующие в природе количественные соотношения между переменными являются, как правило, нелинейными, и часто выполняемое приведение их к линейной форме – лишь одна из возможностей удобной аппроксимации, которая не обязательно будет удовлетворительно моделировать изучаемые процессы по наблюдаемым данным. Поэтому на этапе разведочного анализа данных важно исследовать характер взаимодействия между анализируемыми переменными. Например, получив невысокий коэффициент корреляции Пирсона, мы можем сделать неверный вывод об отсутствии связи между факторами, поскольку эта связь может иметь нелинейный характер (см. раздел 5.6). Обнаруженные на этом этапе закономерности будут определять выбор статистической модели для описания данных, необходимость преобразования нелинейно связанных переменных и т. д.

Характер связи между переменными проще всего выявить, используя соответствующие графические средства. Так, при анализе нескольких количественных переменных очень удобным инструментом являются *матричные диаграммы рассеяния* («*scatterplot matrices*»), или *парные диаграммы рассеяния* («*pair plots*»). Матричные диаграммы рассеяния в R можно построить при помощи нескольких функций. Проще всего воспользоваться базовой R-функцией **pairs**(), которая имеет ряд аргументов для тонкой настройки графика.

В качестве примера рассмотрим первые семь количественных параметров, описывающих разные модели автомобилей, из таблицы `mtcars` (эти данные уже были использованы нами в разделе 3.5). Для облегчения интерпретации характера связи между анализируемыми переменными мы можем добавить сглаживающую кривую к каждой диаграмме рассеяния (аргумент `panel` со значением `panel.smooth`; рис. 68):

```
cars <- mtcars[, 1:7]
pairs(cars, panel = panel.smooth)
```

Аргументы `lower.panel` и `upper.panel` позволяют назначить практически любую функцию для преобразования исходных данных и последующего отображения результатов работы этой функции на графике (ниже или выше центральной диагонали матрицы соответственно). Например, следующим образом мы можем добавить к графику значения коэффициентов корреляции Спирмена, определив собственную функцию заполнения панели:

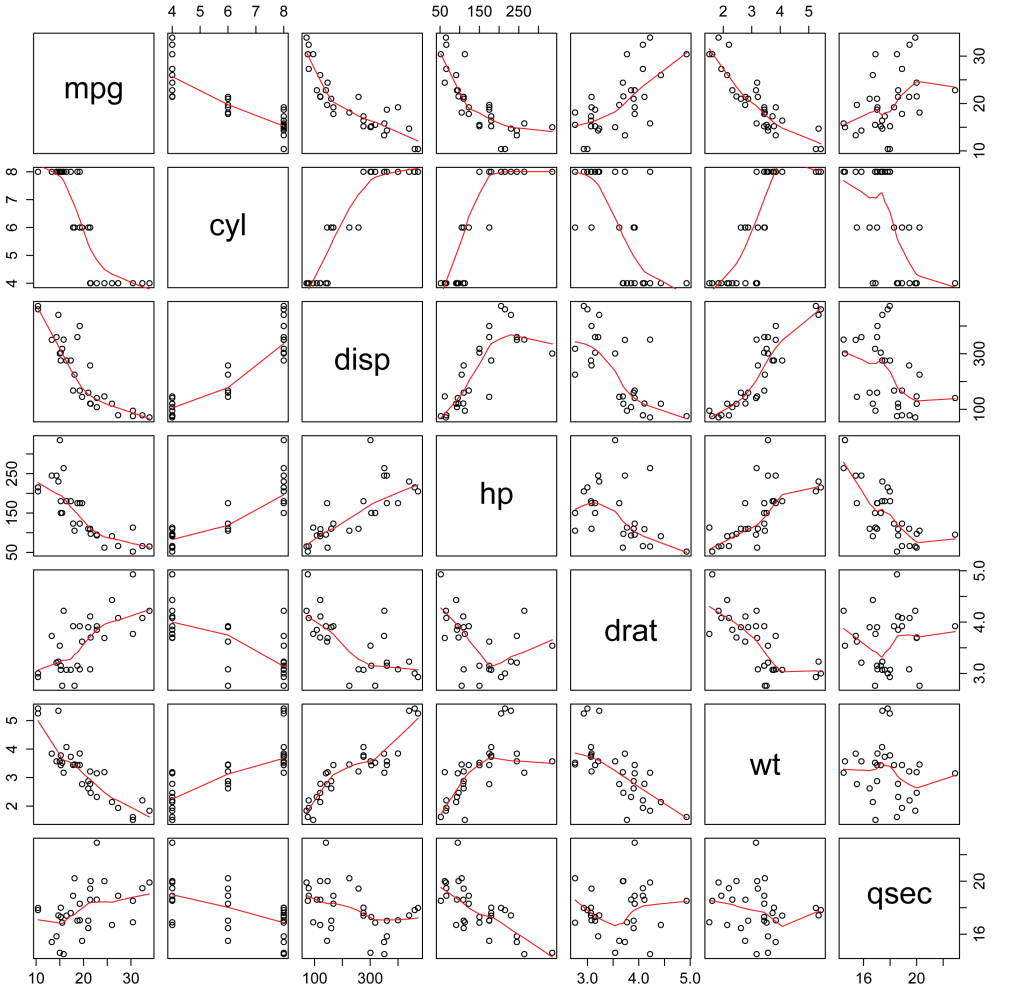


Рисунок 68

```
# функция для оформления панелей с коэффициентом корреляции
panel.cor <- function(x, y, digits = 2,
    prefix = "", cex.cor, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y, method = "spearman"))
    txt <- format(c(r, 0.123456789), digits=digits)[1]
    txt <- paste(prefix, txt, sep = "")
    # эта команда позволяет изменять размер шрифта
    # в соответствии со значением коэффициента корреляции:
```

```
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}
# Строим сам график:
pairs(cars, panel = panel.smooth, lower.panel = panel.cor)
```

Построение матричных диаграмм рассеяния возможно также с использованием функции **splom**() из графического пакета lattice:

```
library(lattice)
splom(cars)
```

Функция **ggpairs**() из пакета GGally, служащего дополнением к одному из лучших графических пакетов для R – ggplot2, позволяет строить как простые матричные диаграммы рассеяния, так и гораздо более сложные графики, включающие, например, двумерные диаграммы ядерной плотности (рис. 69):

```
library(GGally)
ggpairs(cars,
   upper = list(continuous = "density", combo = "box"),
   lower = list(continuous = "points", combo = "dot"))
```

Для количественной оценки справедливости линейной спецификации модели можно использовать значения *эффективных степеней свободы* («*effective degrees of freedom*», *EDF*) нелинейных сглаживающих функций для каждой из независимых переменных. Значения *EDF* легко получить путем подгонки обобщенной аддитивной модели при помощи функции **gam**() пакета mgcv (Wood, 2006) (подробнее этот класс моделей рассматривается в главе 8). Например, для зависимостей между параметрами моделей автомобилей получаем:

```
library(mgcv)
summary(gam(mpg ~ s(hp) + s(qsec), data = cars))
Approximate significance of smooth terms:
          edf Ref.df     F p-value
s(hp)   2.642  3.276 24.08 4.8e-08 ***
s(qsec) 1.228  1.412  3.35   0.065 .
```

Значения оценок *EDF*, превышающие 1.5, свидетельствуют о выраженной нелинейной связи между переменными. По приведенным выше результатам можно заключить, например, что переменная mpg линейно связана с qsec и нелинейно с hp.

Выявление взаимодействий между предикторами

Как было показано выше, вопрос о силе и характере взаимодействия между количественными предикторами решается с использованием корреляционного анализа или функций сглаживания. В случае с категориальными переменными отличным инструментом для выявления взаимодействий между анализируемыми предикторами являются категоризованные графики R (мы подробно рассматривали их в разделе 3.6).

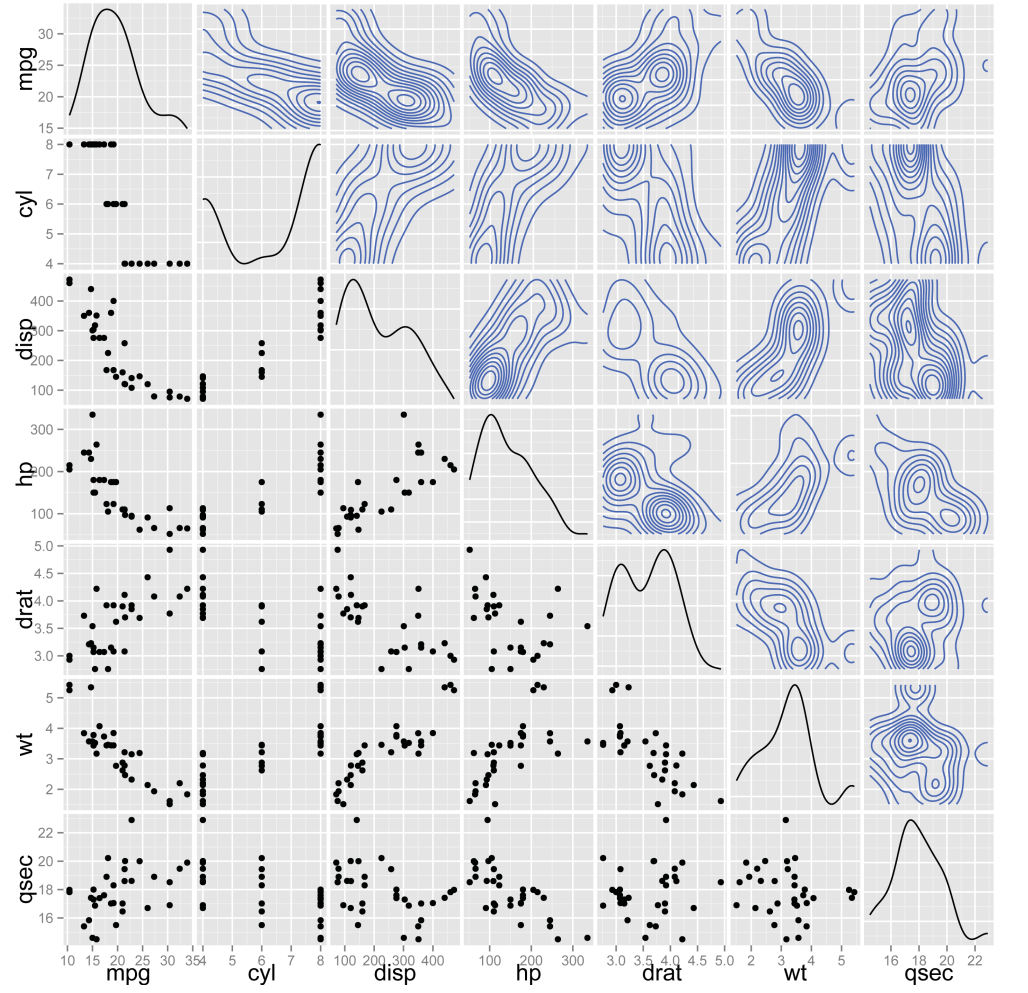


Рисунок 69

Обратимся к орнитологическому примеру, обсуждаемому в работе Zuur et al., (2010). Исследователи задались вопросом о существовании *взаимодействия между* весом воробьев (`weight`), длиной их крыльев (`length`), полом (`sex`) и временем года (`month`).

```
# Загрузка исходных данных из файла
Sparrows <- read.table(file = "SparrowsElphick.txt", header = TRUE)

# Выбираем необходимый комплект данных: воробьи с длиной крыла >= 65
I1 <- Sparrows$SpeciesCode == 1 &
```

```
      Sparrows$Sex != "0" &
      Sparrows$wingcrd < 65
Wing1<- Sparrows$wingcrd[I1]
Wei1 <- Sparrows$wt[I1]
Mon1 <- factor(Sparrows$Month[I1])
Sex1<- factor(Sparrows$Sex[I1])

# Определим месяц и пол как категориальные переменные
fMonth1 <- factor(Mon1, levels = c(5, 6, 7, 8, 9),
                labels = c("Май", "Июнь",
                           "Июль", "Август", "Сентябрь"))
fSex1   <- factor(Sex1, levels = c(4, 5),
                labels = c("Самцы", "Самки"))
```

Силу и направление связи между весом воробьев и длиной крыла в зависимости от их пола и времени проведения измерений наглядно и во всех деталях можно проанализировать на категоризованных диаграммах рассеяния. Для удобства интерпретации этой связи к каждому графику добавим линии регрессии. Если прямые на полученных диаграммах окажутся параллельными, то можно сделать вывод об отсутствии влияния пола воробьев и времени года на связь между длиной крыла и весом птиц (рис. 70; см. также раздел 8.5, посвященный ковариационному анализу).

```
#  Вывод категориальной диаграммы
coplot(Wei1 ~ Wing1 | fMonth1 * fSex1,
       ylab = c("Вес (г)", "Пол"),
       xlab = c("Длина крыла (мм)", "Месяц"),
       panel = function(x, y, ...) {
         tmp <- lm(y ~ x, na.action = na.omit)
         abline(tmp)
         points(x, y) })
```

Линии регрессии на диаграмме не параллельны, что указывает на необходимость включения в статистическую модель соответствующих взаимодействий. На языке R эту модель можно было бы записать в виде `weight ~ length*sex*month`. Подобная запись предполагает наличие в модели эффектов всех трех индивидуальных предикторов, а также всех парных и одного тройного взаимодействия.

Средства визуализации позволяют наглядно установить, что для некоторых сочетаний «месяц/пол» мы имеем недостаточно репрезентативное число наблюдений (например, в сентябре данные для самцов не были получены вовсе), что может подвергнуть сомнению результаты последующего анализа.

Выполним трехфакторный дисперсионный анализ с учетом взаимодействий, причем воспользуемся для этого комбинацией функций **lm**() и **anova**(), о которых речь пойдет в следующих разделах. Пользуясь случаем, покажем, как можно вывести таблицу дисперсионного анализа непосредственно в файл Microsoft Word с применением пакета `stargazer`, который специально предназначен для вывода информации о статистических моделях в форматах LaTeX, HTML и ASCII. Мно-

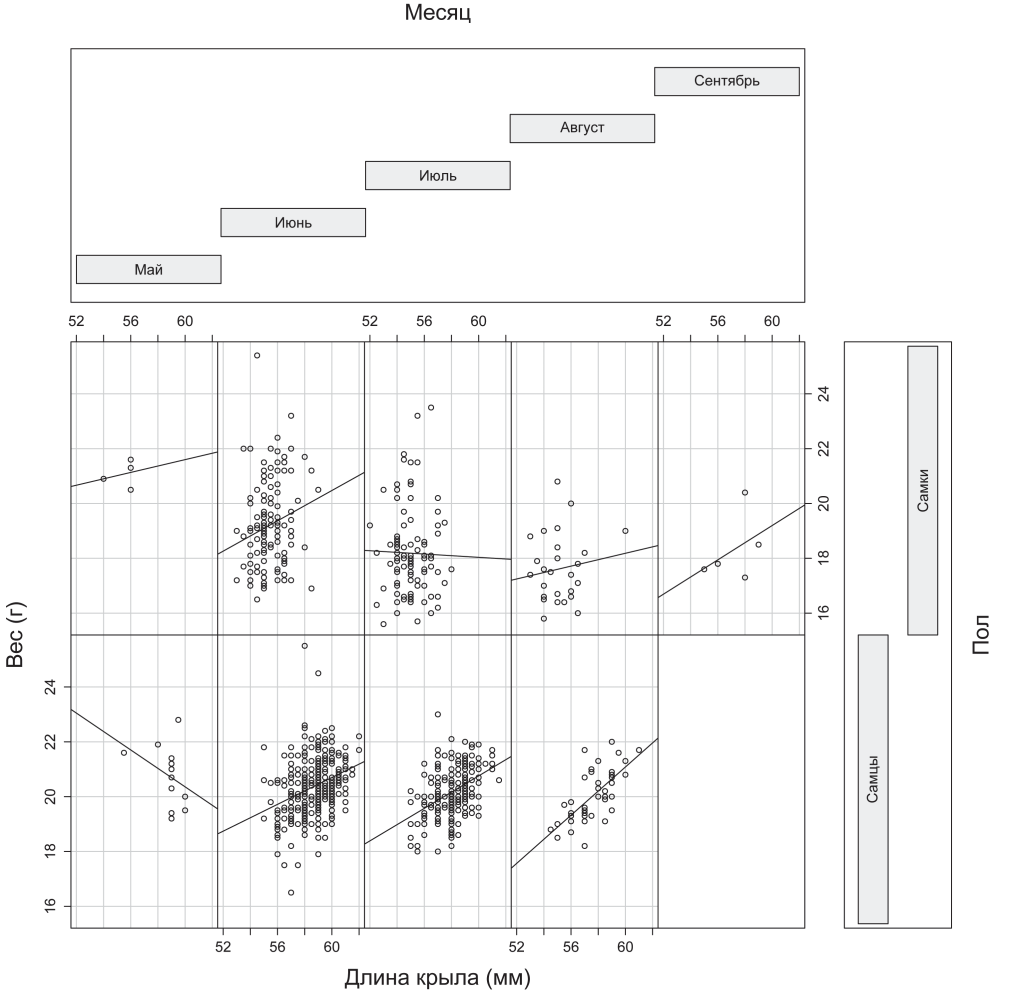


Рисунок 70

гочисленные управляющие параметры функции **stargazer**() позволяют получить в желаемом виде необходимые результаты расчетов:

```
#  Удаляем данные за май и сентябрь
df <- data.frame(weight = Wei1, length = Wing1,
                 sex = fSex1, month = fMonth1)
df1 <- df[df$month != "May" & df$month != "Sep", ]

#  Строим линейную модель и получаем дисперсионную таблицу
M1 <- lm(weight ~ length*month*sex, data = df1)
DT <- anova(M1)
```

```
#  Выводим результаты расчетов в таблицы файлов Word
require(stargazer)
stargazer(M1, type = "html", out = "M1.doc")
stargazer(DT, type = "html", out = "DT.doc", summary = FALSE)
```

Как следует из полученной таблицы, статистически значимым является как влияние всех трех независимых предикторов, так и эффекты трех парных и тройного взаимодействий:

Analysis of Variance Table					
	Df	**Sum Sq**	**Mean Sq**	**F value**	**Pr(> F)**
Response: weight					
length	1	537.216	537.216	446.246	0
month	2	25.233	12.617	10.480	0.00003
sex	1	64.997	64.997	53.991	0
length:month	2	23.375	11.687	9.708	0.0001
length:sex	1	5.446	5.446	4.524	0.034
month:sex	2	26.887	13.443	11.167	0.00002
length:month:sex	2	8.683	4.342	3.607	0.028
Residuals	872	1,049.763	1.204		

Влияние пространственно-временных факторов на анализируемую переменную

Важным условием корректности статистического анализа является решение проблемы «мнимых повторностей» (или псевдорепликации), широкое обсуждение которой (во всяком случае, среди биологов) началось с известной статьи С. Хелберта (Hulbert, 1984). Перевод этой статьи вместе с подробными комментариями представлен в сборнике «Проблемы экологического эксперимента» (2008, http://bit.ly/1ErnT79). Псевдорепликация является одним из наиболее широко цитируемых и недооцененных на практике понятий и определяется как «*использование дедуктивной статистики для оценки влияния фактора, когда данные эксперимента фактически не имеют повторностей или эти повторности не являются статистически независимыми...*».

Например, мы ставим своей целью выявить влияние качества воды на популяционную плотность рыбных сообществ, но этот показатель зависит, в свою очередь, от сезонного периода и гидрологических характеристик участка реки, откуда брались пробы. Поэтому при оценке статистической значимости воздействия загрязнения на встречаемость рыб необходимо обязательно учесть вариацию значений отклика, обусловленную пространственно-временными факторами. Если игнорировать их влияние, то можно значительно увеличить вероятность ошибки первого рода: например, при выполнении регрессионного анализа эта ошибка может возрастать до 400% (Ostrom, 1990, http://bit.ly/1BNI67s).

Существует несколько основных принципов обеспечить статистическую независимость повторностей эксперимента от контролируемых внешних факторов:

1) *рандомизация*, то есть планирование экспериментальных групп таким образом, чтобы они различались между собой (в среднем) лишь уровнем изучаемого воздействия, а прочие факторы влияли бы на сформированные группы случайно и равновероятно;
2) использование *моделей со смешанными эффектами*, о которых мы расскажем в главе 8: в них присутствуют особые члены – случайные факторы, которые аккумулируют в себе всю возможную пространственно-временную изменчивость (Pinheiro & Bates, 2000; Zuur et al., 2009);
3) анализ временной и пространственной *автокорреляции* отклика и построение *кригинговых моделей*, в которых регулярная составляющая изменчивости рядов данных учитывается посредством специально сконструированных предикторов.

В динамических рядах данных неизбежно содержатся существенные периодические колебания (суточные, сезонные, многолетние) вокруг некоторой общей тенденции, которая и сама может иметь периодический характер. Известный русский математик Е. Е. Слуцкий (1927), положивший начало анализу периодичностей, писал: «*Наличие синусоидальных волн различных порядков, начиная с длинных, обнимающих десятилетия, продолжая циклами примерно от пяти до десяти лет длиною и кончая совсем короткими волнами, остается как факт, требующий объяснения*».

Один из наиболее простых подходов оценки автокорреляции между значениями анализируемой переменной заключается в построении графика зависимости отклика от времени или пространственных координат. Наличие каких-либо четких паттернов на таком графике будет указывать на существование корреляции между значениями этой переменной. На рис. 71 слева показаны два небольших временных ряда, отражающих динамику численности двух видов птиц на одном из побережий Аргентины (Zuur et al., 2010). Динамика численности первого вида демонстрирует четкую закономерность, чего нельзя сказать о втором виде.

Формальным способом проверки наличия временной взаимосвязи между значениями анализируемой переменной является анализ графика *автокорреляционной функции* (АКФ). При оценке этой функции последовательно происходит расчет коэффициента корреляции Пирсона между значениями того же временного ряда, но каждый раз взятых со сдвигом по времени на определенную величину k (лаг). В среде R для расчета АКФ служит функция **acf**():

```
# Загрузка данных и определение оси времени
Waders <- read.table(file = "wader.txt", header = TRUE)
Time <- seq(1,25)
# Построение четырех графиков в одном окне
par(mfrow = c(2, 2), mar = c(5, 4, 3, 2))
plot(Time, Waders$C.fuscicolis, type = "l", xlab =
    "Время (2 недели)", ylab = "Численность C. fuscicollis")
```

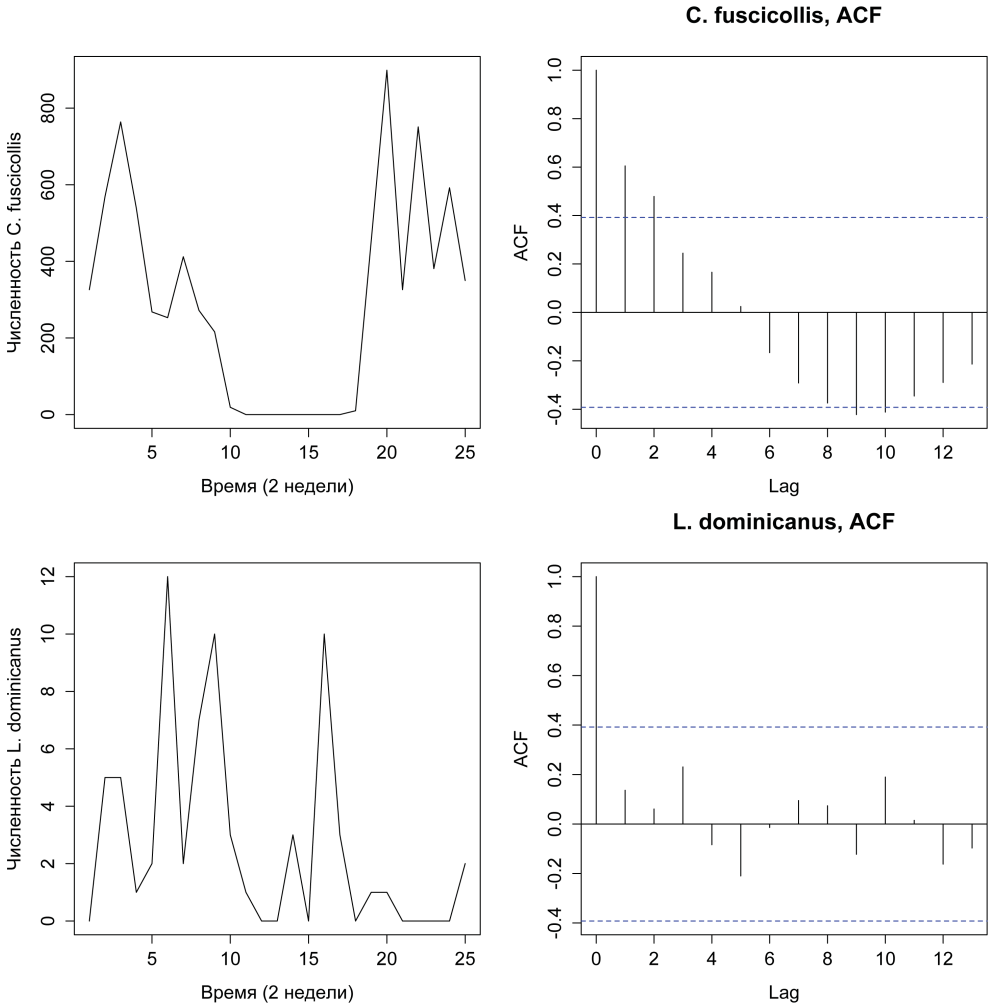


Рисунок 71

```
acf(Waders$C.fuscicolis, main = "C. fuscicollis, ACF")
plot(Time, Waders$L.dominicanus, type = "l", xlab =
    "Время (2 недели)", ylab = "Численность L. dominicanus")
acf(Waders$L.dominicanus, main = "L. dominicanus, ACF")
```

Результатом работы этой функции является построение графиков, приведенных на рис. 71 справа, где хорошо видна тесная временная взаимосвязь между значениями численности *C. fuscicollis* на протяжении первых четырех недель (то есть до $k = 2$; величина лага k соответствует двум неделям). В то же время для второго вида (*L. dominicanus*) подобная зависимость не наблюдается.

Эффективным способом визуальной оценки пространственной анизотропии изучаемого явления является построение так называемых *вариограмм*. Функции для построения вариограмм входят в состав нескольких пакетов для R (например, **Variogram**() из стандартного пакета nlme; **variog**() из пакета geoR; **variogram**() из пакета gstat).

На этом мы завершаем рассмотрение протокола РДА, описанного в работе Zuur et al. (2010). Что же делать, если в ходе анализа своих данных вы обнаружили некоторые отклонения от пунктов представленного протокола? В общем виде ответ таков: использовать современные статистические методы, позволяющие корректно учесть нарушение соответствующих предположений о нормальности, однородности и независимости. Речь прежде всего идет об активном использовании рандомизации, бутстрепа, байесовских методов, робастных форм регрессии, обобщенных линейных и аддитивных моделей и др.

6.2. Дисперсионный анализ как линейная модель

В разделе 5.5, который представлял собой введение в дисперсионный анализ (ANOVA), мы упомянули о том, что в R этот анализ можно выполнить как с помощью функции **aov**(), так и **lm**(), которая предназначена для построения линейных регрессионных моделей. Напомним, что под *статистическими моделями* понимаются математические выражения, описывающие связь между анализируемыми случайными переменными. Модель считается *линейной*, если отклик системы на сумму независимых воздействующих факторов равен сумме взвешенных откликов на каждое воздействие.

Суть «классического» однофакторного дисперсионного анализа мы рассматривали на примере эксперимента из книги Maindonald & Braun (2010), в котором томаты выращивали при трех разных экспериментальных условиях: на воде, в среде с добавлением удобрения, а также в среде с добавлением удобрения и гербицида. В конце эксперимента был измерен вес растений из этих трех групп. Мы выяснили, что каждое измеренное значение веса растений можно разложить на следующие составляющие:

$$x_{ij} = \bar{x}_{..} + (\bar{x}_{i.} - \bar{x}_{..}) + (x_{ij} - \bar{x}_{i.}), \quad (1)$$

где $(\bar{x}_{i.} - \bar{x}_{..})$ – отклонения групповых средних от общего среднего значения, вычисленного по всем имеющимся значениям веса растений, а $(x_{ij} - \bar{x}_{i.})$ – отклонения отдельных наблюдений от среднего значения группы, к которой они принадлежат.

Используя эти отклонения, а также учитывая число анализируемых групп и число наблюдений в каждой группе, мы далее рассчитали меж- и внутригрупповую дисперсии и сравнили их при помощи *F*-критерия Фишера. Исходя из величины полученного критерия, был сделан вывод о том, что у нас нет оснований отклонить нулевую гипотезу об отсутствии разницы между групповыми средними значениями веса растений.

Уравнение (1) для однофакторного дисперсионного анализа можно записать в более простом виде:

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij}, \quad (2)$$

где y_{ij} – вес растения j из экспериментальной группы i, μ – среднее значение веса всех задействованных в эксперименте растений, α_i – разность между μ и средним значением растений в группе i, а ε_{ij} – это *остатки*, отражающие отклонения отдельных наблюдений в группе i от среднего значения этой группы. Предполагается, что остатки имеют нормальное распределение со средним 0 и стандартным отклонением σ, причем это стандартное отклонение *одинаково во всех группах*.

Можно заметить, что уравнение (2) почти ничем не отличается от уравнения простой регрессии (3), которая, выражаясь терминами статистики, является линейной моделью, в которой *переменная-отклик* аддитивна относительно эффектов, оказываемых *предикторами* (то есть независимыми переменными по правую сторону равенства):

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad (3)$$

где y_i – это i-е значение зависимой *количественной переменной* y; x_i – значение *количественного предиктора* x, соответствующее i-му значению y; β_0 – значение, которое принимает y при $x = 0$ (геометрически это точка, в которой линия регрессии пересекает ось Y (отсюда англ. термин для этого коэффициента – «*intercept*», что значит «пересечение»)); β_1 – *коэффициент регрессии*, который показывает, насколько изменяется y при изменении x на одну единицу; ε_i – *остатки*, то есть разница между наблюдаемыми значениями y_i и средним значением, предсказанным моделью по x_i.

Наиболее важные отличия между уравнениями (2) и (3) заключаются в следующем:

- в обоих случаях зависимая переменная y является количественной. Однако в дисперсионном анализе значения y сгруппированы в соответствии с i уровнями изучаемого фактора – отсюда индекс i в обозначении y_{ij}, указывающий, к какой группе принадлежит каждое наблюдение;
- в ANOVA независимая переменная является *качественной*, тогда как в регрессионной модели мы имеем дело с *количественным* предиктором. Степень взаимосвязи между предиктором и независимой переменной в регрессионной модели определяется *коэффициентом регрессии* β_1 в уравнении (3). В дисперсионном анализе *размер эффекта* изучаемого фактора (то есть степень отклонения групповой средней от общего среднего значения) выражен при помощи α_i.

К сожалению, с моделью в уравнении (2) имеется одна загвоздка – не все ее параметры *идентифицируемы*. Это значит, что существует бесконечное множество возможных (*неуникальных*) значений параметров модели. Например, если мы добавим к μ 2.5 (или любую другую константу) и соответственно вычтем 2.5 из каждого α_i, то получим точно те же значения y_{ij}. Однако проблему неидентифицируемости можно легко обойти, наложив определенные ограничения на параметры модели.

При работе с категориальными предикторами (факторами) возможны несколько разных подходов к наложению подобных ограничений (Faraway, 2004). Во многих статистических программах, включая R, величину эффекта для первого уровня анализируемого фактора («*базового уровня*», англ. «*reference level*» или «*baseline level*») по умолчанию принимают равной 0: $\alpha_1 = 0$. Тогда μ будет представлять собой среднее значение зависимой переменной для базового уровня фактора (например, в контрольной группе какого-либо эксперимента), тогда как α_i для $i \neq 1$ будут отражать разницу между средним значением базового уровня и средними значениями остальных уровней. Используя этот подход, мы можем переписать уравнение (2) в виде регрессионного уравнения следующим образом:

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_{k-1} x_{k-1} + \varepsilon_i, \quad (4)$$

где β_0 – среднее значение зависимой переменной для базового уровня изучаемого фактора (например, в контрольной группе); $\beta_1 ... \beta_{k-1}$ – коэффициенты, отражающие разницу между средним значением базового уровня и средними значениями остальных уровней; ε_i – остатки.

Ключевой момент для понимания здесь состоит в том, что уровни изучаемого фактора (всего имеется k таких уровней) закодированы при помощи дополнительной *переменной-индикатора* x («*dummy variable*» – дословно «переменная-пустышка»), которая может принимать только два значения – 0 или 1, в зависимости от того, к какому уровню принадлежит конкретное наблюдение y_i. Например, если наблюдение y_3 принадлежит к уровню x_1, тогда x_1 в уравнении (4) примет значение 1, а $x_2 ... x_{k-1}$ все будут равны 0, и, следовательно, для наблюдения y_3 регрессионное уравнение примет вид:

$$y_3 = \beta_0 + \beta_1 \times 1 + \varepsilon_3 = \beta_0 + \beta_1 + \varepsilon_3.$$

Уравнение (4) как раз и описывает ту линейную модель, которая рассчитывается в R при выполнении однофакторного дисперсионного анализа при помощи функции **lm**(). Посмотрим, как это все работает, вернувшись к примеру с растениями томатов, которые выращивали при разных экспериментальных условиях.

```
# Создадим таблицу с данными:
   tomato <-
      data.frame(weight=
               c(1.5, 1.9, 1.3, 1.5, 2.4, 1.5, # water
                 1.5, 1.2, 1.2, 2.1, 2.9, 1.6, # nutrient
                 1.9, 1.6, 0.8, 1.15, 0.9, 1.6), # nutrient+24D
             trt = rep(c("Water", "Nutrient", "Nutrient+24D"), c(6, 6, 6)))
```

В зависимости от стоящей задачи исследователь волен выбрать в качестве базового любой уровень изучаемого фактора. В рассматриваемом примере имеет смысл проводить сравнения с уровнем Water (растения, выращенные на воде). Если базовый уровень не задан исследователем однозначно, то функция R автоматически

расположит названия всех уровней в алфавитном порядке и выберет в качестве базового первый уровень из этого списка (например, в нашем случае это был бы уровень Nutrient). Задать базовый уровень фактора в R позволяет функция **relevel**():

```
# Автоматически R выберет Nutrient в качестве базового уровня:
levels(tomato$trt)
[1] "Nutrient"     "Nutrient+24D" "Water"

# Изменим базовый уровень на Water:
tomato$trt <- relevel(tomato$trt, ref = "Water")
levels(tomato$trt)
[1] "Water"        "Nutrient"     "Nutrient+24D"
```

Теперь рассчитаем коэффициенты линейной модели для рассматриваемого эксперимента:

```
# Сохраним модель в виде объекта с именем M:
M <- lm(weight ~ trt, data = tomato)

# Просмотрим результаты вычислений:
summary(M)
Call:
lm(formula = weight ~ trt, data = tomato)
Residuals:
    Min      1Q  Median      3Q     Max
-0.5500 -0.3500 -0.1792  0.2750  1.1500
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.68333    0.20849   8.074 7.69e-07
trtNutrient      0.06667    0.29485   0.226    0.824
trtNutrient+24D -0.35833    0.29485  -1.215    0.243
---
Residual standard error: 0.5107 on 15 degrees of freedom
Multiple R-squared: 0.1381, Adjusted R-squared: 0.02321
F-statistic: 1.202 on 2 and 15 DF,  p-value: 0.328
```

Разберемся с основными элементами результатов анализа. Прежде всего программа напоминает нам, к какой модели относятся выводимые результаты (это бывает очень удобно при одновременном построении большого количества моделей): Call: lm(formula = weight ~ trt, data = tomato). Далее следует сводная информация о распределении остатков (Residuals: ...; на эту тему мы поговорим позднее).

Самая интересная часть таблицы с результатами анализа представлена под заголовком Coefficients. Здесь приведены оценки коэффициентов построенной нами модели $y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon_i$. В первой строке (Intercept) мы видим информацию, относящуюся к коэффициенту β_0. В столбце Estimate («оценка») представлено рассчитанное значение этого коэффициента (1.683), которое равно среднему

весу растений в группе Water (заданный нами ранее базовый уровень изучаемого фактора). Во второй строке (trtNutrient) мы находим коэффициент β_1, который отражает разницу между базовым уровнем и средним значением в группе Nutrient (1.683 + 0.067 = 1.750). Наконец, в третьей строке (trtNutrient+24D) мы видим коэффициент β_2, который отражает разность между базовым уровнем и средним значением в группе Nutrient+24D (1.683 – 0.358 = 1.325). Верность описанной интерпретации полученных коэффициентов мы можем легко проверить, рассчитав средние значения для каждой экспериментальной группы:

```
tapply(tomato$weight, tomato$trt, mean)
      Water     Nutrient Nutrient+24D
   1.683333     1.750000     1.325000
```

В столбцах Std. Error, t value и Pr(>|t|) представлены стандартные ошибки рассчитанных коэффициентов, значения *t*-критерия Стьюдента, который используется для проверки гипотезы о равенстве коэффициента нулю, и *p*-значения для каждого *t*-критерия соответственно. Как видим, ни один из исследованных уровней фактора «условия выращивания» не отличается от базового уровня по среднему значению веса растений – в обоих случаях *p*-значения регрессионных коэффициентов оказались > 0.05, и это указывает на то, что наблюдаемое их отличие от нуля с большой долей вероятности является случайным. Другими словами, исследованные условия выращивания в среднем не оказали никакого существенного влияния на вес растений, по сравнению с растениями, выращенными на воде.

Далее в таблице с результатами анализа мы видим:

- Residual standard error – оценка стандартного отклонения остатков. Вспомним: предполагается, что остатки имеют нормальное распределение со средним значением 0 и стандартным отклонением σ. В данной строке результатов анализа как раз и приводится оценка значения σ;
- Multiple R-squared и Adjusted R-squared – коэффициент детерминации и коэффициент детерминации с поправкой на число параметров модели соответственно;
- F-statistic – значение критерия Фишера, при помощи которого проверяется нулевая гипотеза о том, что все коэффициенты истинной модели равны 0.

Значение *F*-критерия для линейных моделей рассчитывается как отношение дисперсии в данных, «объясненной» моделью, к дисперсии остатков (то есть части общей дисперсии, которую модель «не объясняет»). Это легко увидеть, подав объект M на функцию **anova**() (не путать с **aov**()):

```
anova(M)
Analysis of Variance Table
Response: weight
          Df Sum Sq Mean Sq F value Pr(>F)
trt        2 0.6269 0.31347  1.2019  0.328
Residuals 15 3.9121 0.26081
```

Полученная дисперсионная таблица идентична таблице с результатами однофакторного дисперсионного анализа, которую мы представили в разделе 5.5 с использованием функции **aov**().

Идентичность результатов неудивительна, поскольку в конечном счете мы в обоих случаях оцениваем одни и те же наиболее важные величины – дисперсию в данных, ассоциированную с действием изучаемого фактора, и остаточную дисперсию, которую мы не можем приписать действию этого фактора. С точки зрения программного кода, **aov**() является лишь «функцией-упаковщиком» (*«wrapper function»*), назначение которой сводится к вызову функции **lm**(). Различие между этими двумя функциями состоит только в том, как получаемые с их помощью результаты выводятся при подаче на **print**() и **summary**(). В случае с **aov**() результаты выводятся в виде классической таблицы дисперсионного анализа, а не в виде списка с коэффициентами линейной модели.

Следует также отметить, что функция **aov**() предназначена для анализа *сбалансированных* наборов данных, то есть таких ситуаций, когда имеется одинаковое число наблюдений для каждого уровня изучаемого фактора. Если сбалансированность в данных отсутствует, следует использовать функцию **lm**() (подробнее см. справочный файл, доступный по команде `?lm`).

Описанное тесное сходство между моделями классического дисперсионного анализа и линейной регрессией неслучайно, поскольку обе являются частными случаями *общей линейной модели* (ОЛМ, англ. General Linear Model; не путать с *обобщенными линейными моделями* – Generalized Linear Models). Понимание концепции ОЛМ очень важно для осмысленного использования **lm**() и других функций R, позволяющих создавать линейные модели. Поэтому стоит остановиться на ОЛМ более подробно.

Общую линейную модель можно выразить в матричном виде следующим образом:

$$\mathbf{Y} = \mathbf{XB} + \mathbf{U}, \tag{5}$$

где **Y** – многомерная матрица предсказываемых моделью значений; **X** – так называемая *матрица модели* (*«model matrix»*), или *матрица плана* (*«design matrix»*), которая содержит значения входящих в модель предикторов; **B** – матрица параметров модели, которые оцениваются на основе имеющихся данных, а **U** – матрица остатков модели.

В случае однофакторного дисперсионного анализа мы имеем дело с одномерной зависимой переменной, так что матрицы **Y** и **U** в приведенном уравнении (5) будут представлять собой векторы из n значений каждый (где n – это объем наблюдений), тогда как размер матрицы модели **X** будет определяться количеством уровней анализируемого фактора и объемом наблюдений.

Для рассматриваемого нами примера с томатами мы можем переписать уравнение (4) в матричной форме, следуя концепции ОЛМ:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{15} \\ y_{16} \\ y_{17} \\ y_{18} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}. \quad (6)$$

В уравнении (6) по левую сторону от знака «=» мы видим все измеренные значения веса растений из соответствующих экспериментальных групп (сначала идут 6 значений из контрольной группы, затем 6 значений из группы `Nutrient` и, наконец, 6 значений из группы `Nurtient+24D`).

Сразу после знака «=» представлена матрица плана модели размером 18×3. Как было отмечено выше, размерность этой матрицы определяется объемом наблюдений (n = 18) и числом подлежащих оценке параметров модели (в нашем случае p = 3). Матрица модели содержит значения индикаторной переменной x, которая принимает только два значения – 0 и 1 – и служит для обозначения принадлежности каждого наблюдения y_i к определенной экспериментальной группе. Матрица модели умножается на матрицу с параметрами модели, которая в рассматриваемом примере представлена вектором из 3 величин – β_0, β_1 и β_2.

Просмотреть матрицу той или иной подогнанной линейной модели позволяет функция **`model.matrix()`**. Для созданной нами выше модели `M` эта функция выведет матрицу, в точности соответствующую таковой в уравнении (6):

```
model.matrix(M)
   (Intercept) trtNutrient trtNutrient+24D
1            1           0               0
2            1           0               0
...  ...  ...
18           1           0               1
```

```
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$trt
[1] "contr.treatment"
```

О значении в приведенном выше результате такого важного выражения, как «contr.treatment», мы расскажем в следующем разделе.

Оценки параметров модели M, полученные выше с использованием функции **lm()**, представляют собой вектор **B** = {1.683, 0.067, –0.358}. Таким образом, модель предсказывает, что в контрольной группе средний вес растений составит 1.683 кг, в группе Nutrient он будет на 0.067 кг выше (1.683 + 0.067 = 1.750 кг), а в группе Nurtient+24D) – на 0.358 кг ниже (1.683 – 0.358 = 1.325 кг), чем в контрольной группе.

Конечно, реальные измеренные значения веса будут в большинстве случаев несколько отличаться от предсказываемых моделью средних значений для каждой группы. Отсюда мы и получаем то, что называется остатками, – это разности между наблюдаемыми и предсказанными значениями, которые не могут быть объяснены включенными в модель предикторами.

Согласно правилам умножения матриц, каждое значение y_i получается путем поэлементного произведения чисел из i-й строки матрицы модели на вектор, содержащий параметры модели. Например, прогнозируемый вес растения для первого значения из контрольной группы и остаток для него ε_1 мы получим следующим образом:

$$y_1 = \beta_0 \times 1 + \beta_1 \times 0 + \beta_2 \times 0 = 1.683 \times 1 + 0.067 \times 0 - 0.358 \times 0 = 1.683;$$
$$\varepsilon_1 = 1.5 - 1.683 = -0.183;$$

для первого значения веса из группы Nutrient:

$$y_7 = \beta_0 \times 1 + \beta_1 \times 1 + \beta_2 \times 0 = 1.683 \times 1 + 0.067 \times 1 - 0.358 \times 0 = 1.750;$$
$$\varepsilon_7 = 1.5 - 1.750 = -0.250;$$

для первого значения веса из группы Nurtient+24D:

$$y_{13} = \beta_0 \times 1 + \beta_1 \times 0 + \beta_2 \times 1 = 1.683 \times 1 + 0.067 \times 0 - 0.358 \times 1 = 1.325;$$
$$\varepsilon_{13} = 1.9 - 1.325 = 0.575.$$

Очевидно, что при большом объеме наблюдений и числе параметров модели полная запись модели по образцу уравнения (6) будет неэкономичной (представьте себе, например, ситуацию с 1000 измеренных значений зависимой переменной и 10 параметрами модели!). В связи с этим в статистике и применяют компактную матричную форму записи линейных моделей, приведенную выше в уравнении (5).

Универсальность матричной формы состоит еще и в том, что матрица модели может включать значения предикторов любого типа – как количественных, так и качественных. Так, в случае дисперсионного анализа мы имеем дело только с *качественными* переменными-предикторами и матрица модели содержит только

значения 1 и 0, отражающие принадлежность того или иного наблюдения к конкретной группе. В случае множественной регрессии матрица модели состоит из значений нескольких *количественных* предикторов. Наконец, при наличии *как количественных, так и качественных предикторов* мы имеем дело с так называемым ковариационным анализом («*analysis of covariance*», ANCOVA) и матрица модели содержит значения предикторов обоих типов. Вне зависимости от того, о каком из вариантов ОЛМ идет речь, модель всегда может быть записана в виде уравнения (5). О множественной регрессии и ковариационном анализе мы расскажем в главах 7 и 8 соответственно.

6.3. Структура модельных объектов дисперсионного анализа

Рассмотрим на примере классического однофакторного дисперсионного анализа, где в соответствующем объекте R хранятся результаты моделирования и как получить к ним доступ, чтобы, в частности, проанализировать адекватность построенной модели.

В качестве примера используем хорошо знакомые нам данные эксперимента по изучению эффективности 6 разных инсектицидов (таблица `InsectSprays` из базовой версии R; см. также раздел 3.3). Каждым из этих препаратов (фактор `spray`) обработали по 12 растений, после чего подсчитали количество выживших на растениях насекомых (отклик `count`). Нулевая гипотеза, которую мы проверим при помощи дисперсионного анализа, заключается в том, что исследованные инсектициды не различаются по эффективности, то есть наблюдаемые различия в групповых средних значениях числа выживших насекомых совершенно случайны.

Линейную модель, параметры которой нам предстоит оценить, можно записать следующим образом:

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \varepsilon_i,$$

где y_i – это число насекомых на i-м растении, β_0 – среднее значение числа насекомых для базового уровня изучаемого фактора (по умолчанию R выберет в качестве базового инсектицид `A`), β_1, β_2, β_3 и т. д. – коэффициенты, отражающие разницу между средним значением числа насекомых на растениях из группы `A` и средними значениями в других группах (`B`, `C` ... `F`), а ε_i – остатки модели.

Оценим параметры приведенной модели при помощи функции **`lm`**`()`:

```
M <- lm(count ~ spray, data = InsectSprays)
```

В ходе выполнения приведенной команды внешне ничего не происходит. На самом же деле мы сохранили результаты вычислений в виде самостоятельного объекта `M` – *модельного объекта* («*model object*»).

Такой подход к выводу и хранению результатов подгонки линейных и других моделей является одним из существенных преимуществ R в сравнении с боль-

шинством других статистических программ, которые просто выводят результаты вычислений на экран, без возможности их непосредственного использования в дальнейшем анализе. Рассмотрим, как устроены модельные объекты в R и как извлекать из них отдельные элементы.

Результат работы функции **lm**() – это достаточно сложно устроенный объект, который с точки зрения представления данных в нем является списком. Соответственно, при необходимости мы можем без труда извлечь отдельные элементы из этого списка и использовать их в последующих вычислениях. Строение любого объекта R легко выяснить при помощи функции **str**() («*structure*» – структура):

```
str(M)
List of 13
 $ coefficients : Named num [1:6] 14.5 0.833 -12.417 -9.583 -11 ...
  ..- attr(*, "names")= chr [1:6] "(Intercept)" "sprayB" "sprayC" "sprayD" ...
 $ residuals    : Named num [1:72] -4.5 -7.5 5.5 -0.5 -0.5 ...
  ..- attr(*, "names")= chr [1:72] "1" "2" "3" "4" ...
 $ effects      : Named num [1:72] -80.6 22.1 -24.2 -19.9 -34.2 ...
  ..- attr(*,"names")= chr [1:72] "(Intercept)" "sprayB" "sprayC" "sprayD" ...
 $ rank         : int 6
 $ fitted.values: Named num [1:72] 14.5 14.5 14.5 14.5 14.5 ...
  ..- attr(*, "names")= chr [1:72] "1" "2" "3" "4" ...
 $ assign       : int [1:6] 0 1 1 1 1 1
 $ qr           :List of 5
  ..$ qr   : num [1:72, 1:6] -8.485 0.118 0.118 0.118 0.118 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:72] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:6] "(Intercept)" "sprayB" "sprayC" "sprayD" ...
  .. ..- attr(*, "assign")= int [1:6] 0 1 1 1 1 1
  .. ..- attr(*, "contrasts")=List of 1
  .. .. ..$ spray: chr "contr.treatment"
  ..$ qraux: num [1:6] 1.12 1.05 1.06 1.07 1.09 ...
  ..$ pivot: int [1:6] 1 2 3 4 5 6
  ..$ tol  : num 1e-07
  ..$ rank : int 6
  ..- attr(*, "class")= chr "qr"
 $ df.residual  : int 66
 $ contrasts    :List of 1
  ..$ spray: chr "contr.treatment"
 $ xlevels      :List of 1
  ..$ spray: chr [1:6] "A" "B" "C" "D" ...
 $ call         : language lm(formula = count ~ spray, data = InsectSprays)
 $ terms        :Classes 'terms', 'formula' length 3 count ~ spray
  .. ..- attr(*, "variables")= language list(count, spray)
  .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:2] "count" "spray"
  .. .. .. ..$ : chr "spray"
```

```
  .. ..- attr(*, "term.labels")= chr "spray"
  .. ..- attr(*, "order")= int 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. ..- attr(*, "predvars")= language list(count, spray)
  .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "factor"
  .. .. ..- attr(*, "names")= chr [1:2] "count" "spray"
 $ model        :'data.frame': 72 obs. of  2 variables:
  ..$ count: num [1:72] 10 7 20 14 14 12 10 23 17 20 ...
  ..$ spray: Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "terms")=Classes 'terms', 'formula' length 3 count ~ spray
  .. .. ..- attr(*, "variables")= language list(count, spray)
  .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. .. ..$ : chr [1:2] "count" "spray"
  .. .. .. .. ..$ : chr "spray"
  .. .. ..- attr(*, "term.labels")= chr "spray"
  .. .. ..- attr(*, "order")= int 1
  .. .. ..- attr(*, "intercept")= int 1
  .. .. ..- attr(*, "response")= int 1
  .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. .. ..- attr(*, "predvars")= language list(count, spray)
  .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "factor"
  .. .. .. ..- attr(*, "names")= chr [1:2] "count" "spray"
 - attr(*, "class")= chr "lm"
```

От обилия выведенных результатов может закружиться голова, но переживать не стоит, так как далеко не все строки потребуют пристального внимания. Главное здесь то, что модельный объект `M`, как хорошо видно, представляет собой список из 13 основных элементов (`List of 13`). Отметим также, что похожий сложно устроенный объект может быть создан при выполнении дисперсионного анализа при помощи функции **`aov`**`()`.

Некоторые из этих элементов сами являются списками – отсюда такая сложная структура. Тем не менее, используя стандартные для R приемы индексации списков, мы можем легко извлечь любой желаемый элемент объекта `M`. На практике наиболее интересными и важными для анализа являются, в частности, такие элементы, как `coefficients` (коэффициенты подогнанной линейной модели), `residuals` (остатки) и `fitted.values` (предсказанные моделью значения).

Используя оператор `$`, мы можем, например, извлечь остатки модели и предсказанные ею значения и сохранить их в виде самостоятельных векторов:

```
M.res <- M$residuals
M.fit <- M$fitted.values
```

Эти новые векторы мы далее можем использовать для других целей – например, отобразить хранящиеся в них значения графически для проверки условия гетероскедастичности в отношении дисперсии остатков модели:

```
plot(M.fit, M.res, pch = 19, col = 4,
     xlab = "Предсказанные значения", ylab = "Остатки")
```

Для более быстрого извлечения определенных элементов из модельных объектов, созданных при помощи **aov**(), **lm**() и многих других функций, в R имеются соответствующие функции-экстракторы («*extractor functions*»). В частности, наиболее полезны:

- **coefficients**() или, в сокращенном виде, **coef**() – извлекает коэффициенты линейной модели;
- **residuals**() или, в сокращенном виде, **resid**() – извлекает остатки модели;
- **fitted**() – извлекает предсказанные моделью значения.

Так, приведенная ниже команда позволяет рассчитать коэффициент корреляции Пирсона между исходными данными и предсказанными моделью значениями:

```
cor.test(fitted(M2), InsectSprays$count)
Pearsons product-moment correlation
data:  fitted(M2) and InsectSprays$count
t = 13.5657, df = 70, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7716214 0.9044641
sample estimates:
      cor
0.8511398
```

6.4. Оценка адекватности модели дисперсионного анализа

Выше было показано, что дисперсионный анализ представляет собой частный случай общей линейной модели, оценку параметров которой позволяет выполнить базовая R-функция **lm**(). Однако с подгонки модели серьезный анализ только начинается. Необходимо осуществить комплексную оценку *адекватности модели* (англ. «*model adequacy*»), то есть проверить условия выполнения ряда допущений, традиционно используемых при построении линейных моделей, выполнить интерпретацию оценок параметров, вывести графики распределения остатков и т. д.

Проверка исходных предположений общей линейной модели

Перед тем как начать содержательно интерпретировать рассчитанные параметры и выдвигать научные гипотезы на их основе, необходимо выяснить, выполняются ли лежащие в основе модели *допущения* («*model assumptions*»).

Здесь для нас наибольший интерес представляет строка в таблице с результатами подгонки модели, обозначенная как `F-statistic`. В этой строке представлено расчетное значение *F*-критерия (`34.7` при `5` и `66` числах степеней свободы), при помощи которого проверяется гипотеза об одновременном равенстве всех коэффициентов истинной модели нулю. Как видно из соответствующего *p*-значения это-

го критерия (`p-value: < 2.2e-16`), вероятность того, что эта гипотеза верна, крайне мала. Другими словами, мы можем заключить, что эффективность исследованных инсектицидов существенно различается.

Однако насколько мы можем доверять полученному *F*-значению (и соответствующему ему *p*-значению), чтобы сделать такой вывод? Другими словами, насколько рассчитанное значение *F*-критерия близко к тому значению, которое следует ожидать, исходя из свойств *F*-распределения и верной нулевой гипотезы? Это соответствие будет полным при одновременном выполнении следующих трех условий:

- во всех группах значения переменной-отклика (число выживших насекомых) являются независимыми;
- во всех анализируемых группах (инсектициды `A`, `B`, ..., `F`) значения зависимой переменной распределены нормально;
- групповые дисперсии статистически не различаются (так называемое «условие однородности», или «гомоскедастичности», дисперсии).

Формально проверить (и скорректировать) первое условие сложно – его выполнение следует обеспечивать на стадии планирования исследования (например, путем случайного назначения каждого растения в ту или иную экспериментальную группу) и в ходе сбора данных (например, используя строго один и тот же метод подсчета насекомых на растениях из всех экспериментальных групп).

Для проверки остальных двух допущений существуют графические и формальные методы, которые мы подробно рассмотрели ранее (разделы 4.8 и 5.4 соответственно). Ниже мы обсудим методологическую сторону проблемы применительно к дисперсионному анализу.

Проверка условия нормальности распределения

Условие нормальности в отношении дисперсионного анализа предполагает, что значения зависимой переменной имеют нормальное распределение в пределах каждой группы, определяемой уровнями изучаемого фактора (или факторов) (так, в рассматриваемом эксперименте с инсектицидами значения числа выживших насекомых должны быть нормально распределены в каждой из 6 экспериментальных групп). Если изучаемый фактор оказывает существенное влияние на интересующую нас зависимую переменную, то значения этой переменной, взятые все вместе (то есть без разделения их по группам), совсем необязательно будут иметь нормальное распределение.

Поясним это явление на примере имитированных данных. Предположим, что у нас имеется некоторая количественная переменная, значения которой разбиты на 3 группы в соответствии с уровнями некоторого гипотетического фактора. В каждую группу включим по 100 наблюдений, случайным образом сгенерированных из нормально распределенных совокупностей. Кривые плотностей распределения вероятности для этих данных приведены на рис. 72. На том же рисунке справа приведено распределение плотности вероятности для всех данных, объединенных без учета их групповой принадлежности. Очевидно, что хотя исходные наблюдения в каждой группе A, B и C имели нормальное распределение, после их объединения в одну большую совокупность свойство нормальности исчезает.

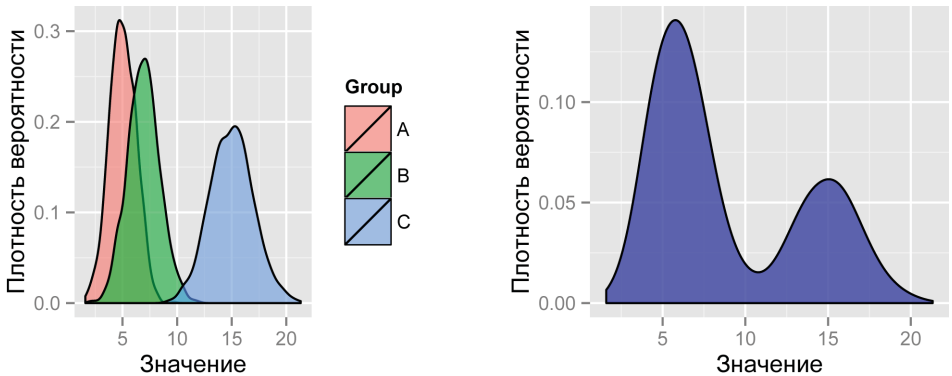


Рисунок 72

К сожалению, на практике во многих исследованиях (в частности, эколого-биологического профиля) не удается сформировать экспериментальные группы с достаточно большим числом измерений. При малых объемах выборок (от 8 до 20) проверка предположения о нормальности распределения с использованием формальных статистик Шапиро-Уилка, Андерсона-Дарлинга и других – скорее ритуал, чем объективно корректная процедура. В этих условиях они обладают ложной чувствительностью и «*...фактически не способны различать гипотезы H_0 и H_1*» (Лемешко, 2005, http://bit.ly/1AGVIxo). Здесь возрастает роль графических методов, основанных на удивительной способности человеческого глаза обнаруживать сходство или различия геометрических образов.

Визуальная оценка нормальности распределения, которая обычно сводится к изучению формы гистограммы, диаграммы плотности вероятности значений анализируемой переменной или графика квантилей, при малой численности групп также весьма затруднительна. Например, в эксперименте с инсектицидами имеется лишь по 12 наблюдений на группу. Попытка использовать гистограммы для проверки условия нормальности распределения в этом случае ни к чему определенному не приведет (рис. 73).

Более информативно выглядят квантильные графики, хотя и с их помощью однозначный вывод о нормальности распределения значений в *каждой* экспериментальной группе сделать также затруднительно (рис. 74).

Как же быть? На помощь приходят свойства нормального распределения. Дело в том, что если значения зависимой переменной в каждой экспериментальной группе распределены нормально, то *нормально будут распределены и значения остатков соответствующей линейной модели*. При этом предполагается, что распределение остатков ε_i имеет среднее значение, равное 0, и некоторое стандартное отклонение σ_ε. Формально мы можем записать это так: $\varepsilon_i \sim N(0, \sigma_\varepsilon)$. Таким образом, вместо проверки нормальности распределения значений зависимой переменной в каждой группе достаточно проверить нормальность распределения остатков модели (число которых, по определению, равно общему объему наблюдений).

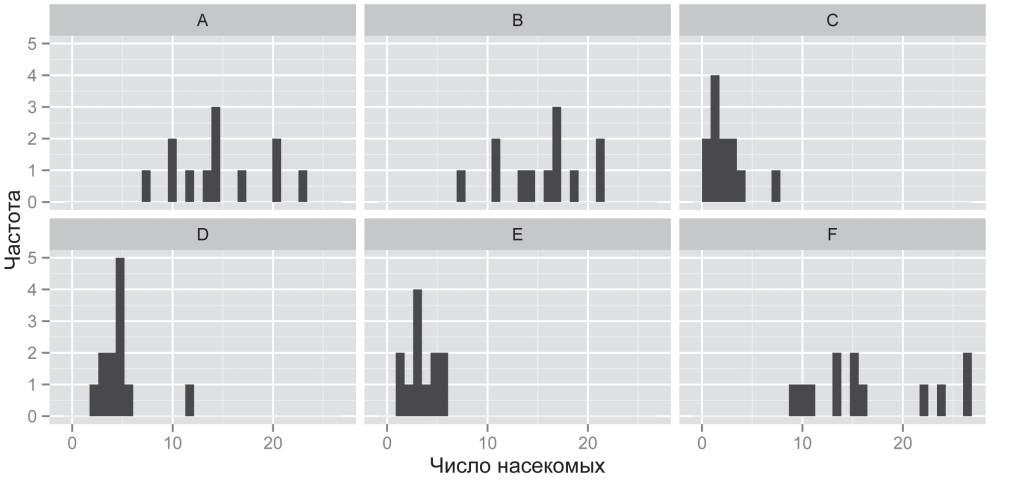


Рисунок 73

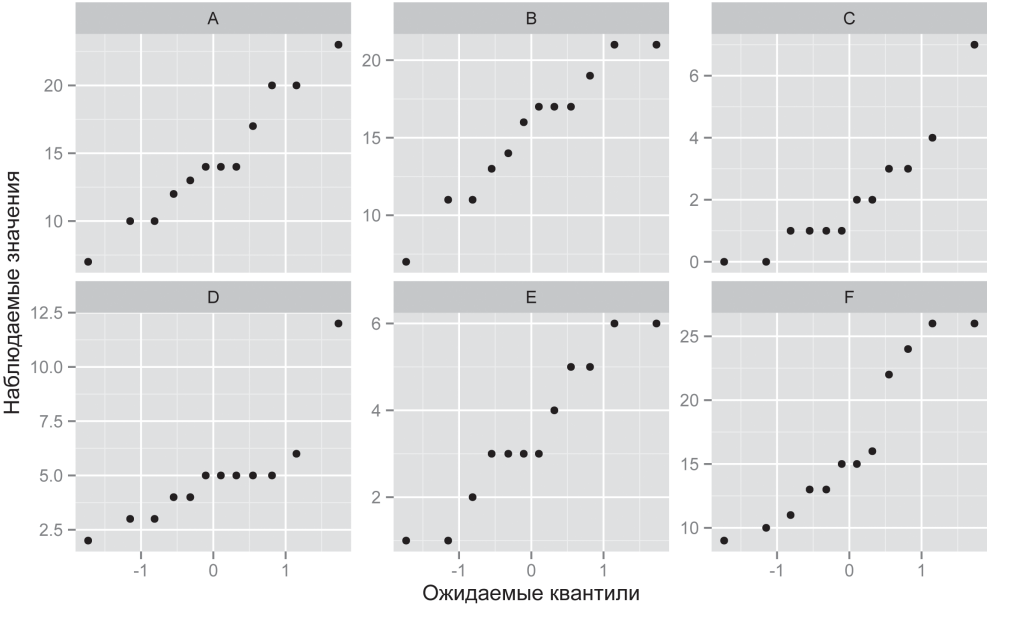


Рисунок 74

На рис. 75 приведены гистограмма с наложенной на нее кривой плотности вероятности и график квантилей для остатков подогнанной нами ранее модели `M`.

По рис. 75 можно сделать предварительное заключение о том, что распределение остатков модели `M` не соответствует нормальному (обратите внимание на на-

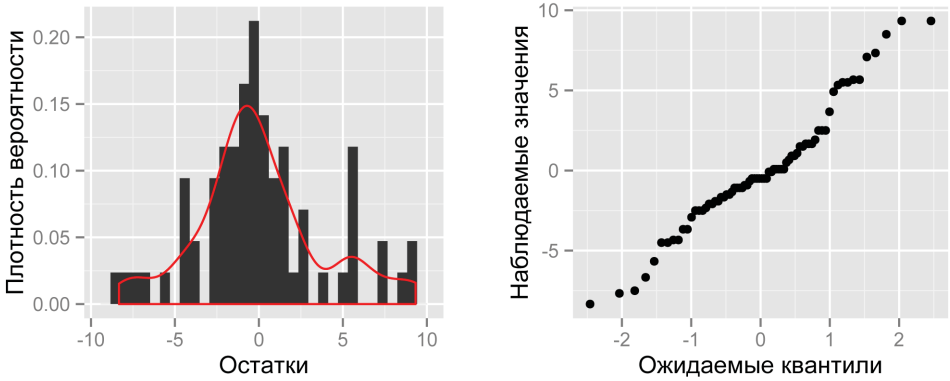


Рисунок 75

личие второго «пика» в правой части гистограммы, а также на *S*-образную фигуру на графике квантилей). Впрочем, это и неудивительно, поскольку в данном случае мы имеем дело с числом выживших насекомых – такого рода дискретные количественные переменные по определению не подчиняются нормальному закону распределения.

Формально мы можем проверить этот вывод, например, при помощи теста Шапиро-Уилка (для всех 72 измерений он уже обладает достаточной мощностью):

```
shapiro.test(resid(M))
Shapiro-Wilk normality test
data: resid(M) W = 0.9601, p-value = 0.02226
```

Как видно из полученного *p*-значения критерия W (`p-value = 0.022`), распределение остатков модели `M` отличается от нормального, что согласуется с визуальной оценкой характера их распределения.

Таким образом, условие нормальности в отношении модели `M` не выполняется. Это значит, что модель `M` не является адекватной для описания анализируемых данных, и, соответственно, мы не можем делать строгих выводов на ее основе. Далее будут даны некоторые рекомендации по поводу того, как поступать в таких ситуациях.

Проверка условия однородности групповых дисперсий

Вторым важным условием применимости классического дисперсионного анализа является *однородность*, или «*гомоскедастичность*», групповых дисперсий (англ. «*homogeneity of variance*», или «*homoscedasticity of variance*»). Речь здесь идет о том, что, помимо нормального распределения в каждой группе, значения зависимой переменной должны также иметь одинаковую степень разброса. Необходимость выполнения этого условия определяется способом вычисления внутри- и межгрупповых дисперсий, применяемым в классическом дисперсионном анализе (подробнее см. раздел 5.5): при значительно различающихся групповых дисперсиях используемые формулы просто не будут работать корректно.

Способы проверки однородности групповых дисперсий включают как графические, так и формальные методы (см. разделы 3.3 и 5.4). При большом числе наблюдений в экспериментальных группах это условие можно проверить при помощи диаграмм `boxplot()`: сравнение интерквартильных размахов позволяет сделать заключение о том, насколько велики межгрупповые различия в разбросе данных. Рисунок 76 иллюстрирует данный подход на примере эксперимента с инсектицидами. Из этого рисунка хорошо виден неодинаковый разброс наблюдений в разных группах, что указывает на несоблюдение условия однородности дисперсии.

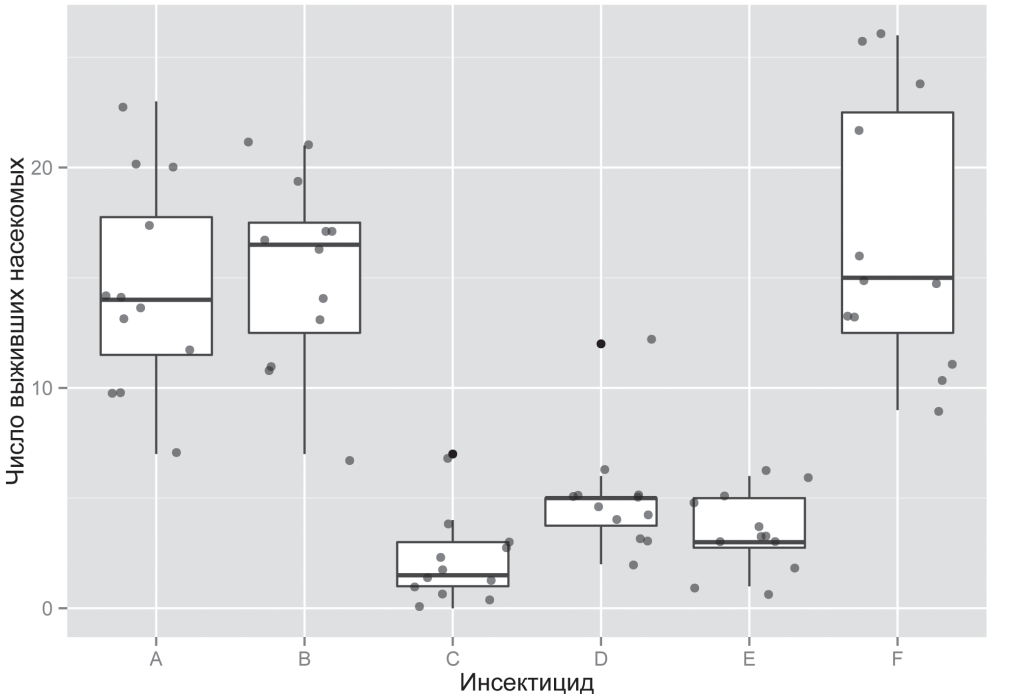


Рисунок 76

Подобно тому, как это было сделано нами при проверке условия нормальности, вместо исходных данных мы можем использовать остатки модели и для проверки условия однородности групповых дисперсий. Как и в случае с исходными данными, визуально оценить разброс остатков в каждой группе позволяет диаграмма размахов.

Другим часто используемым способом является изучение графика, на котором по оси X откладывают предсказанные моделью значения зависимой переменной, а по оси Y – соответствующие остатки (рис. 77). При соблюдении условия однородности дисперсии точки на таком графике не должны формировать каких-либо выраженных паттернов.

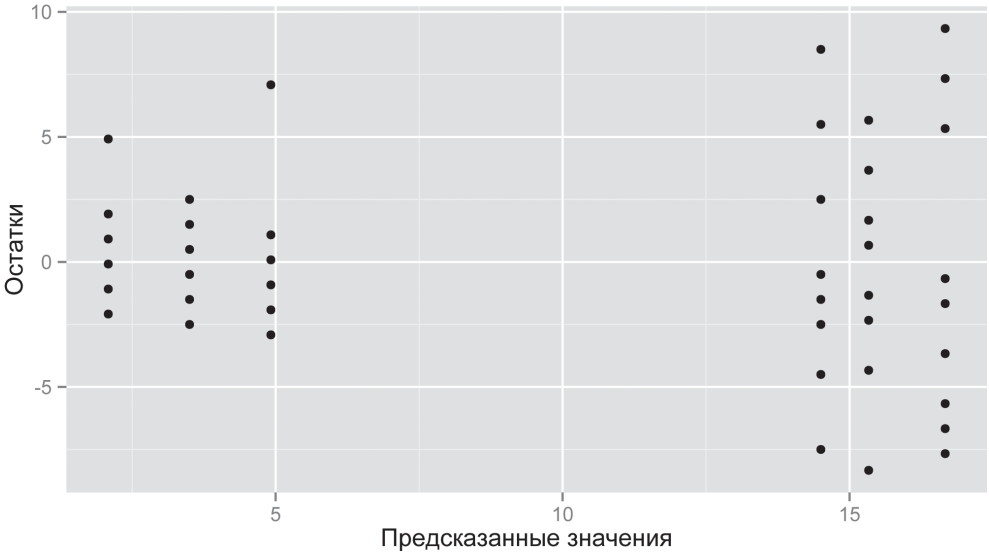


Рисунок 77

По рис. 77 хорошо видно, что по мере увеличения предсказанных моделью средних групповых значений разброс остатков также увеличивается. Такая «воронкообразная» форма распределения остатков типична для случаев, когда условие однородности групповых дисперсий не выполняется. Напомним еще раз, что в данном случае это вполне ожидаемо, поскольку мы имеем дело с дискретными значениями, скорее всего, распределенными по закону Пуассона.

Неудивительно, что тест Левене также оказался существенно значимым, указывая на выраженную неоднородность групповых дисперсий:

```
library(car)
leveneTest(InsectSprays$count, InsectSprays$spray)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value   Pr(>F)
group  5  3.8214 0.004223 **
      66
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Таким образом, для данных из эксперимента по изучению эффективности инсектицидов не выполняется ни условие нормальности, ни условие однородности групповых дисперсий. Как следствие обычный однофакторный дисперсионный анализ не является адекватным методом для этих данных.

Что делать, когда однофакторный дисперсионный анализ неприменим?

Следует отметить, что однофакторный дисперсионный анализ в определенной степени все же устойчив к нарушениям условий нормальности и гомоскедастичности. В частности, при одинаковом (или «близком» к одинаковому) числе наблюдений во всех группах метод будет относительно устойчив к нарушениям условия однородности групповых дисперсий. Если размер экспериментальных выборок достаточно велик (например, несколько десятков), то требование нормальности распределения групповых значений также ослабевает (этот эффект обеспечивается центральной предельной теоремой). Часто в таких условиях проверка равенства математических ожиданий сравниваемых групп с помощью критерия Фишера дает правильные результаты, независимо от того, выполнены предпосылки нормальности и равенства дисперсий или нет (Zar, 2010; Шитиков, Розенберг, 2014).

Однако какие-либо четкие рекомендации по поводу того, в каких случаях нарушения условий нормальности и гомоскедастичности следует считать допустимыми, дать очень сложно. Все будет определяться конкретными данными и стоящими перед исследователем задачами. Тем не менее при серьезных нарушениях этих условий использование параметрического дисперсионного анализа для установления влияния того или иного фактора на изучаемое явление будет некорректным. В зависимости от того, какие именно проблемы возникают при выполнении анализа, решения могут быть разными. Рассмотрим возможные ситуации.

- *Распределение в группах нормальное, дисперсии неоднородны.* Когда данные в каждой экспериментальной группе распределены нормально, но условие однородности групповых дисперсий не выполняется, можно воспользоваться методом Уэлча («*Welch's test*»). Этот метод позволяет внести определенную поправку в число степеней свободы, что приводит и к соответствующему изменению итогового значения *F*-критерия. В R однофакторный дисперсионный анализ по Уэлчу выполняется при помощи функции **`oneway.test`**`()`. В качестве альтернативы поправке Уэлча можно также использовать *перестановочные тесты* («*permutation tests*», или «*randomization tests*»). Подробно на русском языке эти тесты описаны в книге Шитиков, Розенберг (2013). Одним из пакетов R, позволяющих выполнять перестановочные тесты для линейных моделей, является `lmPerm`.
- *Распределение в группах ненормальное, дисперсии однородны.* Когда распределение значений зависимой переменной в экспериментальных группах отличается от нормального, часто помочь может определенная трансформация этих значений. Наиболее обычным способом трансформации количественных переменных является логарифмирование. При работе с относительными частотами и долями часто используется также извлечение квадратного корня из исходных значений. Эти, а также некоторые другие способы являются частными случаями более общего подхода – степенной

трансформации по Боксу-Коксу (англ. *Box-Cox transformation*). Одним из преимуществ такого преобразования является также то, что помимо приведения данных к нормальному распределению оно помогает еще «стабилизировать» групповые дисперсии (то есть позволяет достичь выполнения условия гомоскедастичности). Пример того, как это работает, приведен ниже.

Если при помощи трансформации не удается привести данные к нормальному распределению, вместо классического однофакторного дисперсионного анализа можно применить его непараметрический аналог – ранговый дисперсионный анализ по Краскелу-Уоллису. Этот метод проверяет нулевую гипотезу о равенстве центров (медиан) сравниваемых групп, используя ранги исходных значений, и поэтому особенно хорошо работает при наличии выбросов. Хотя формально метод Краскела-Уоллиса предполагает однородность дисперсий сравниваемых групп, на практике нарушение этого условия слабо влияет на конечные результаты (Logan, 2010). Поэтому этот метод можно также использовать в ситуациях, когда не выполняются условия как нормальности, так и гомоскедастичности.

- *Распределение в группах ненормальное, дисперсии неоднородны.* Предположим, что трансформация исходных значений зависимой переменной не позволила привести их к нормальному распределению в каждой группе и стабилизировать групповые дисперсии. Пусть также есть основания полагать, что метод Краскела-Уоллиса не даст надежных результатов, например из-за существенной неоднородности групповых дисперсий. Тогда единственным выходом будет использование статистических моделей, которые не предполагают соблюдения этих условий. Речь идет о классе обобщенных линейных моделей (см. главу 8), где дополнительно задается некоторая *функция связи* $g(y^{-1})$, конкретный выбор которой осуществляется в зависимости от природы распределения отклика y. Тем самым обобщенные линейные модели существенно расширяют диапазон приложений линейного моделирования, охватывая широкий класс альтернативных распределений вероятностей.

В рассмотренном выше примере с инсектицидами мы выяснили, что условия нормальности и гомоскедастичности не выполняются. Для исправления этой ситуации мы можем попробовать трансформировать исходные данные путем логарифмирования и подогнать новую линейную модель на основе трансформированных значений:

```
M.log <- lm(log(count + 1) ~ spray, data = InsectSprays)
# Обратите внимание: к исходным значениям добавлена 1.
# Это обычный прием при логарифмировании данных,
# содержащих нулевые значения (поскольку логарифм нуля, как известно, не определен).
summary(M.log)
Call:
lm(formula = log(count + 1) ~ spray, data = InsectSprays)
Residuals:
     Min       1Q   Median       3Q      Max
```

```
-0.95261 -0.25946  0.01131  0.25935  1.12683
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.6967     0.1181  22.829  < 2e-16 ***
sprayB        0.0598     0.1671   0.358    0.721
sprayC       -1.7441     0.1671 -10.440 1.30e-15 ***
sprayD       -0.9834     0.1671  -5.887 1.45e-07 ***
sprayE       -1.2705     0.1671  -7.605 1.35e-10 ***
sprayF        0.1189     0.1671   0.712    0.479
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.4092 on 66 degrees of freedom
Multiple R-squared:  0.7771, Adjusted R-squared:  0.7602
F-statistic: 46.01 on 5 and 66 DF,  p-value: < 2.2e-16
```

Обратите внимание на то, что значение *F*-критерия, рассчитанное при помощи новой модели (`M.log`), превышает исходное (46.01 против 34.7, рассчитанного по модели `M`). Хотя в данном случае вывод, который мы делаем в отношении различий инсектицидов по эффективности, не меняется, по сравнению с первоначальным выводом, такая ситуация будет наблюдаться после трансформации данных не всегда.

Тест Шапиро-Уилка показывает, что условие нормальности распределения остатков модели `M.log` выполняется:

```
shapiro.test(resid(M.log))
Shapiro-Wilk normality test
data:  resid(M.log)
W = 0.9847, p-value = 0.5348
```

Аналогично тест Левене показывает, что условие однородности групповых дисперсий теперь также соблюдено:

```
leveneTest(log(InsectSprays$count + 1), InsectSprays$spray)
Levenes Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  5  1.8821 0.1093
      66
```

Таким образом, логарифмирование исходных данных привело к выполнению обоих важных условий применимости однофакторного дисперсионного анализа, и мы можем обоснованно использовать модель `M.log` для того, чтобы делать заключения по поводу эффективности исследованных инсектицидов.

6.5. Дисперсионный анализ по Краскелу-Уоллису

Дисперсионный анализ по Краскелу-Уоллису («*Kruskal-Wallis ANOVA by ranks*», или «*Kruskal-Wallis rank sum test*») является непараметрическим аналогом ANOVA (Kruskal & Wallis, 1952, http://bit.ly/1HuFZZM). В русскоязычной литературе

для этого метода используются также названия «критерий Крускала-Уоллиса», «*H*-критерий Крускала-Уоллиса» и даже иногда «критерий Крускала-Валлиса».

Поскольку дисперсионный анализ по Краскелу-Уоллису относится к группе непараметрических методов статистики, это значит, что при выполнении соответствующих расчетов параметры того или иного вероятностного распределения (например, нормального) никак не задействованы. Вместо этого используются ранги исходных значений и их суммы в сравниваемых группах. В частности, метод Краскела-Уоллиса основан на вычислении *H*-критерия:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} R_i^2 - 3(N+1),$$

где n_i – число наблюдений в группе *i*, N – общее число наблюдений во всех k группах, а R_i – сумма рангов наблюдений в группе *i*. Ранг представляет собой порядковый номер конкретного наблюдения в ряду упорядоченных по возрастанию (или по убыванию, что не важно) наблюдений. Чем больше значение *H*-критерия, тем больше у нас оснований отклонить нулевую гипотезу об отсутствии разницы между сравниваемыми группами.

Если рассчитанное по выборочным данным значение *H* превышает определенное *критическое значение*, нулевая гипотеза отклоняется. Критическое значение определяется с учетом принятого в исследовании уровня значимости и числа степеней свободы: в частности, при $m > 5$ *H*-критерий сравнивается с критическими значениями критерия «хи-квадрат» для числа степеней свободы $m - 1$. При меньшем числе сравниваемых групп вносятся определенные поправки (например, поправка Имана-Давенпорта).

Интересно, что если бы мы выполнили обычный дисперсионный анализ, используя ранговые номера исходных значений анализируемой переменной, то *результат совпал бы с результатом теста Краскела-Уоллиса* (Zar, 2010). Кроме того, при наличии двух сравниваемых групп тест Краскела-Уоллиса будет идентичен тесту Манна-Уитни.

Если бы анализируемые данные удовлетворяли условиям нормальности и однородности групповых дисперсий, то статистическая мощность теста Краскела-Уоллиса в отношении таких данных составила бы примерно 95% от обычного параметрического дисперсионного анализа. Однако при нарушении этих условий мощность теста Краскела-Уоллиса может оказаться даже выше, чем у обычного дисперсионного анализа.

В теории для расчета несмещенных оценок *H*-критерия Краскела-Уоллиса значения анализируемой переменной должны иметь *одинаковый разброс* и *форму распределения* во всех сравниваемых группах. Однако, как отмечалось выше, на практике нарушение этих условий мало сказывается на качестве выводов, получаемых при помощи *H*-критерия, и ими обычно пренебрегают.

В R дисперсионный анализ по Краскелу-Уоллису выполняется при помощи функции **`kruskal.test`**`()`. В качестве примера, как и в предыдущем разделе, исполь-

зуем данные эксперимента по изучению эффективности 6 разных инсектицидов. Нулевая гипотеза, которую мы проверим при помощи теста Краскела-Уоллиса, заключается в том, что исследованные инсектициды не различаются по эффективности (то есть наблюдаемые различия в групповых *медианных значениях* числа выживших насекомых совершенно случайны).

```
kruskal.test(count ~ spray, data = InsectSprays)
 Kruskal-Wallis rank sum test
data:  count by spray
Kruskal-Wallis chi-squared = 54.6913, df = 5, p-value = 1.511e-10
```

Как видно из приведенного результата, вероятность получить столь высокое наблюдаемое значение *H*-критерия при верной нулевой гипотезе крайне мала, и, следовательно, мы можем смело эту гипотезу отклонить.

Следует подчеркнуть, что, подобно классическому дисперсионному анализу, тест Краскела-Уоллиса позволяет сделать заключение только следующего вида: либо «сравниваемые группы статистически значимо различаются» (например, при $p < 0.05$), либо «статистически значимых различий между группами нет» (например, при $p > 0.05$). Ни один из этих методов сам по себе не позволяет сказать, какие именно группы различаются. Чтобы выяснить это, необходимо выполнить соответствующие *апостериорные тесты* (*«post hoc tests»*), речь о которых пойдет ниже.

6.6. Модели двух- и многофакторного дисперсионного анализа

В многофакторных экспериментах каждый уровень варьирования одного фактора комбинируется со всеми возможными уровнями остальных факторов. В отличие от однофакторного дисперсионного анализа, который ставит своей основной целью проверить статистическую значимость эффекта группировки, многофакторный ANOVA смещает акцент в сторону оценки *степени влияния* отдельных факторов или *взаимодействий* между ними. В этом смысле он еще более близок множественной регрессии с ее аппаратом селекции наиболее информативного комплекса объясняющих переменных.

Методологию многофакторного анализа в значительной мере определяют наличие повторностей и характер распределения общего количества наблюдений по «ячейкам» (то есть экспериментальным группам), в связи с чем выделяют следующие типы *дисперсионного комплекса*:

- комплексы с единственным наблюдением в ячейке, когда расчет статистической ошибки невозможен и в качестве таковой используется взаимодействие факторов;
- равномерные или сбалансированные комплексы, когда в каждой ячейке содержится одинаковое количество наблюдений (наиболее желательный для анализа вариант);

- пропорциональные комплексы, когда соотношение числа наблюдений на всех уровнях фактора одинаковое (например, во всех группах число особей мужского пола ровно в 2 раза больше, чем женского);
- неравномерные или несбалансированные комплексы с разным числом наблюдений в ячейках;
- комплексы с пропущенными значениями, когда есть ячейки без наблюдений, которые могут возникнуть вследствие потери части экспериментального материала или в специальных экспериментальных планах с ограничениями на рандомизацию.

Последние две категории дисперсионных планов часто некорректно анализируются в рамках классического дисперсионного анализа, основанного на расчете средних квадратов групповых отклонений и оценке значения *F*-критерия. При работе с любыми *несбалансированными* наборами данных способ выполнения дисперсионного анализа, реализованный в функции **aov**(), будет давать смещенные оценки *p*-значений (подробнее см. справочный файл по этой функции – `?aov`). Дополнительные трудности возникают при включении в модель двух или более случайных эффектов (см. раздел 8.6).

Использование общей линейной модели существенно упрощает технику дисперсионного анализа. Во-первых, при создании модели исследователь получает доступ к детальным результатам расчетов, что позволяет ему лучше понять, где именно сосредоточены различия между сравниваемыми группами. Во-вторых, использование синтаксиса формул R дает возможность гибко управлять селекцией значимых фиксированных эффектов и их взаимодействий, а также выбирать и оперативно перестраивать различные многофакторные схемы. Наконец, имеется возможность строить модели с неполными повторными измерениями, когда число наблюдений для разных объектов различно, а также с произвольным законом распределения отклика. Дальнейшее развитие математического аппарата моделирования в направлении обобщенных линейных («*linear mixed-effects model*», LMEM) и нелинейных (NMEM) моделей со смешанными эффектами еще больше расширяет возможности общей линейной модели (Pinheiro & Bates, 2000; Wood, 2006), в том числе становится возможным:

- анализ коррелированных наблюдений и данных с непостоянной дисперсией;
- моделирование не только средних значений, но также дисперсии и ковариации, что помогает оценить долю изменчивости отклика, обусловленную влиянием случайных факторов (например, эффект пространственно-временной корреляции).

Синтаксис объекта «формула»

Рассмотрим предварительно правила составления формул в R (на основе обучающих материалов, представленных на сайте http://bit.ly/MArWaE). Формула представляет собой объект R, задающий в виде символьного выражения отношения между откликом, или зависимой переменной (слева), и предикторами, или не-

зависимыми переменными (справа), которые разделяются знаком ~ («тильда»). В момент начала подгонки модели переменные, указанные в формуле, должны быть представлены объектами (векторами, матрицами), «видимыми» относительно *поискового пути* R, или заданы параметром `data = <таблица данных>`.

Основное назначение формул – спецификация различных статистических моделей регрессионного или дисперсионного анализа. В этих случаях в правой части формулы могут быть представлены объекты трех типов:

- подмножество числовых векторов, объединенных знаком «+» (каждому такому вектору соответствует один рассчитываемый коэффициент регрессионной модели β);
- численная матрица, для каждого из столбцов которой оценивается коэффициент регрессии;
- подмножество факторов, каждому уровню которых соответствует свой коэффициент модели (если факторы связаны знаком «+», то в модель включаются только главные факторы без учета их взаимодействия – см. ниже).

Если члены формулы связаны символом «`:`», то в модель включается только их парный эффект (так, `A:B` отражает взаимодействие факторов `A` и `B`). Использование в формуле символа умножения «`*`» приводит к включению как главных эффектов, так и всех возможных комбинаций взаимодействий (то есть выражение `A*B` эквивалентно `A + B + A:B`, а `A*B*C` эквивалентно `A + B + C + A:B + A:C + B:C + A:B:C`). Возможно использование и других операторов в формуле, в частности «`%in%`» для «гнездового дисперсионного анализа» («*nested ANOVA*»), «`|`» – для определения случайных эффектов и т. д.

Примеры формул:

- рассчитываются свободный член уравнения и коэффициенты регрессии двух переменных (β_0, β_1 и β_2):

```
Y ~ X1 + X2;
```

- та же модель регрессии, но без свободного члена β_0:

```
Y ~ 0 +X1 + X2 или Y ~ -1 + X1 + X2;
```

- предикторами являются все столбцы таблицы `mydata`, кроме отклика `Y`:

```
Y ~ ., data = mydata;
```

- двухфакторная модель без парных взаимодействий (рассчитывается столько коэффициентов, сколько есть уровней в факторах `group1` и `group2`):

```
Y ~ group1 + group2;
```

- двухфакторная модель с главными эффектами, всеми комбинациями парных взаимодействий и эффектом совместного действия всех трех факторов:

```
Y ~ group1*group2*group3;
```

- гнездовая модель, учитывающая вариацию Y под влиянием фактора River и фактора Site, который варьирует только в пределах уровней фактора River:

```
Y ~ River + Site %in% River;
```

- смешанная модель, включающая фиксированный фактор Water и два случайных фактора RiverType и Season:

```
Y ~ Water + (1 | RiverType) + (1 | Season).
```

Как отмечалось выше, просмотреть матрицу плана модели можно с помощью функции **model.matrix**().

Кроме объявленных типов объектов, в состав формулы можно включить любые выражения, принятые в R, в том числе различные преобразования и вызовы функций. Например:

- **cut**(Age,3) – количественная переменная Age конвертируется в фактор с тремя уровнями;
- **I**(age^2) – числовой вектор age возводится в квадрат; общий синтаксис таких команд: **I**(<любое допустимое выражение>);
- **poly**(Age,3) – переменная Age трансформируется в полином 3-й степени;
- **smooth**(x, k = s) – строится функция сглаживания вектора x с заданной степенью.

Такая гибкость синтаксиса объекта «формула» обеспечивает его широкое использование не только в моделях регрессии/ANOVA, но и при построении комплексных диаграмм, деревьев классификации и во многих других приложениях R.

Выполнение двухфакторного дисперсионного анализа при помощи функции lm()

Как было показано ранее в разделе 5.5, двухфакторный дисперсионный анализ («*two-way analysis of variance*», или «*two-way ANOVA*») позволяет установить одновременное влияние двух факторов, а также взаимодействие между этими факторами. В качестве примера были рассмотрены данные из таблицы weightgain, содержащей измерения прироста массы лабораторных крыс (weightgain) в четырех группах с разным характером питания. Корма имели разное содержание белка (фактор type с двумя уровнями: низкое содержание – Low, высокое содержание – High) и разное происхождение (фактор source с двумя уровнями: beef – говядина, cereal – злаки).

Теперь применим для этих данных функцию **lm**():

```
data(weightgain, package = "HSAUR2")
M2 <- lm(weightgain ~ type*source, data = weightgain)

summary(M2)
Call:
lm(formula = weightgain ~ type * source, data = weightgain)
Residuals:
```

```
   Min     1Q Median     3Q    Max
-29.90  -9.90   2.05  10.85  25.10
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         100.000      4.729  21.148  < 2e-16 ***
typeLow             -20.800      6.687  -3.110  0.00364 **
sourceCereal        -14.100      6.687  -2.109  0.04201 *
typeLow:sourceCereal  18.800      9.457   1.988  0.05447 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 14.95 on 36 degrees of freedom
Multiple R-squared:   0.23, Adjusted R-squared:  0.1658
F-statistic: 3.584 on 3 and 36 DF,  p-value: 0.02297
```

Оцененные параметры модели можно интерпретировать подобно тому, как мы это делали в случае однофакторной модели. В первой строке таблицы с параметрами модели (`Intercept`) представлена информация, относящаяся к среднему значению прироста веса в группе крыс, которым давали корм с высоким содержанием белка (`High`) животного происхождения (`Beef`) (для простоты обозначим эту базовую группу как «`High - Beef`»). Видим, что средний прирост веса в этой группе составил 100 г и что этот прирост значимо отличается от 0 (стандартная ошибка = 4.729 г, значение t-критерия Стьюдента = 21.148, P-значение << 0.001).

Вторая строка (`typeLow`) содержит оценку величины эффекта, связанного с низким содержанием в корме белка животного происхождения: в этой группе крыс прирост веса оказался существенно ниже (на 20.8 г, p = 0.00364), чем в группе «`High - Beef`» (100 – 20.8 = 79.2 г). Из третьей строки таблицы (`sourceCereal`) мы видим, что у крыс, которым давали корм с высоким содержанием белка растительного поисхождения, прирост веса был статистически значимо ниже (на 14.1 г, p = 0.04201), чем в группе «`High - Beef`» (100 – 14.1 = 85.9 г). Наконец, в четвертой строке мы находим оценку эффекта взаимодействия между факторами «`type`» и «`source`» – 18.8 г.

Сложив первые три коэффициента с эффектом взаимодействия, получим среднее значение прироста веса крыс в группе, которым давали корм с низким содержанием белка растительного происхождения: 100 – 20.8 – 14.1 + 18.8 = 83.9 (см. также приведенную ранее таблицу в разделе 5.5 со средними значениями по каждой группе). Как видим, этот средний прирост лишь незначительно (p = 0.05447) превысил таковой в группе крыс, которым давали корм с низким содержанием белка растительного происхождения (83.9 против 79.2).

Приведенные результаты хорошо согласуются со значениями, которые мы получили ранее в ходе расчета описательных статистик по каждой группе, а также при выполнении графического разведочного анализа данных (см. раздел 5.5).

Результаты дисперсионного анализа, выполненного при помощи функции **`lm`**`()`, можно представить также и в виде классической таблицы ANOVA. Для этого соответствующий модельный объект необходимо использовать как входящий параметр функции **`anova`**`()`:

```
anova(M2)
Analysis of Variance Table
Response: weightgain
            Df Sum Sq Mean Sq F value  Pr(>F)
type         1 1299.6 1299.60  5.8123 0.02114 *
source       1  220.9  220.90  0.9879 0.32688
type:source  1  883.6  883.60  3.9518 0.05447 .
Residuals   36 8049.4  223.59
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Представленные в этой таблице результаты идентичны таковым в таблице, полученной ранее при помощи функции **aov**().

Порядок перечисления предикторов в формуле модели

В ходе спецификации модели M2 мы сначала указали фактор type, а затем source (то есть weightgain ~ type*source). Что произойдет, если этот порядок изменить?

```
M3 <- lm(weightgain ~ source*type, data = weightgain)

anova(M3)
Analysis of Variance Table
Response: weightgain
            Df Sum Sq Mean Sq F value  Pr(>F)
source       1  220.9  220.90  0.9879 0.32688
type         1 1299.6 1299.60  5.8123 0.02114 *
source:type  1  883.6  883.60  3.9518 0.05447 .
Residuals   36 8049.4  223.59
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Результаты идентичны тем, что были получены для модели M2. Единственная разница состоит в том, что строки, содержащие информацию по факторам type и source, поменялись местами.

Такое полное совпадение результатов будет наблюдаться только при работе со сбалансированными наборами данных. Для демонстрации этого утверждения изменим исходные данные путем удаления, например, первых 6 наблюдений и последних 7 наблюдений:

```
weightgain2 <- weightgain[-c(1:6, 34:40), ]

# Модели с разным порядком указания предикторов:
M4 <- lm(weightgain ~ type*source, data = weightgain2)
M5 <- lm(weightgain ~ source*type, data = weightgain2)

# ANOVA-таблица для модели M4
anova(M4)
Analysis of Variance Table
Response: weightgain
```

```
            Df Sum Sq Mean Sq F value  Pr(>F)
type         1  758.0  758.02  3.1655 0.08843 .
source       1  584.8  584.76  2.4419 0.13179
type:source  1  744.5  744.54  3.1092 0.09113 .
Residuals   23 5507.6  239.46
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# ANOVA-таблица для модели M5
anova(M5)
Analysis of Variance Table
Response: weightgain
            Df Sum Sq Mean Sq F value  Pr(>F)
source       1 1188.8 1188.83  4.9645 0.03594 *
type         1  153.9  153.95  0.6429 0.43087
source:type  1  744.5  744.54  3.1092 0.09113 .
Residuals   23 5507.6  239.46
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Как видим, выводы, сделанные на основании этих двух новых моделей, оказываются совершенно разными: ни один из параметров модели `M4` не является статистически значимым, тогда как модель `M5` указывает на существование значимого эффекта источника происхождения белка. Такая разница обусловлена алгоритмом, по которому происходит разложение общей дисперсии на составляющие при анализе несбалансированных наборов данных. Какая из последних двух моделей отражает действительность?

Ответ на этот вопрос в значительной степени будет определяться тем, как именно мы хотим представить результаты анализа. Если дисперсионный анализ рассматривать как частный случай общей линейной модели и представлять его результаты в виде таблицы с оценками параметров такой модели, тогда никакой разницы между `M4` и `M5` нет. Соответствующие оценки лишь изменяют свое положение в таблице результатов – это еще раз подчеркивает универсальность концепции общих линейных моделей:

```
summary(M4)
Call:
lm(formula = weightgain ~ type * source, data = weightgain2)
Residuals:
   Min     1Q Median     3Q    Max
-27.00 -10.82   2.00  11.18  23.10
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         100.000      4.893  20.435 3.02e-16 ***
typeLow             -16.250      9.155  -1.775   0.0891 .
sourceCereal        -24.000     10.187  -2.356   0.0274 *
typeLow:sourceCereal  24.150     13.696   1.763   0.0911 .
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 15.47 on 23 degrees of freedom
Multiple R-squared:  0.2748, Adjusted R-squared:  0.1802
F-statistic: 2.906 on 3 and 23 DF,  p-value: 0.05643
```

```
summary(M5)
Call:
lm(formula = weightgain ~ source * type, data = weightgain2)
Residuals:
   Min     1Q Median     3Q    Max
-27.00 -10.82   2.00  11.18  23.10
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        100.000      4.893  20.435 3.02e-16 ***
sourceCereal       -24.000     10.187  -2.356   0.0274 *
typeLow            -16.250      9.155  -1.775   0.0891 .
sourceCereal:typeLow  24.150    13.696   1.763   0.0911 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 15.47 on 23 degrees of freedom
Multiple R-squared:  0.2748, Adjusted R-squared:  0.1802
F-statistic: 2.906 on 3 and 23 DF,  p-value: 0.05643
```

Если же мы хотим представить результаты анализа несбалансированных данных в виде классической таблицы ANOVA, следует уделить особое внимание разведочному анализу данных. Особенно полезным, в частности, будет график дизайна эксперимента (см. раздел 5.5). Тот фактор, который, судя по такому графику, оказывает наибольшее влияние на изучаемую переменную-отклик, следует включать в модель первым. Для нашей видоизмененной таблицы с данными получаем (рис. 78):

```
plot.design(weightgain2)
```

Поскольку источник происхождения имеет больший эффект на прирост веса крыс, чем уровень содержания белка, предпочтение следует отдать модели `M5`.

Многофакторный дисперсионный анализ

При наличии более двух факторов говорят о *многофакторном дисперсионном анализе*. Тщательно спланированный и выполненный многофакторный дисперсионный анализ может оказаться очень полезным для понимания изучаемого явления. Однако следует помнить о возникающих при этом затруднениях. Уже при наличии трех основных факторов (X, Y и Z) в модель войдут четыре дополнительных параметра, отражающих взаимодействия между этими факторами (XY, XZ, YZ, и XYZ). Чем больше параметров мы оцениваем, тем больше степеней свободы нам нужно. Другими словами, требуется собрать больше данных, что не всегда возможно.

Кроме того, при проверке статистических гипотез в отношении большого числа параметров возрастает вероятность сделать неверное заключение о значимо-

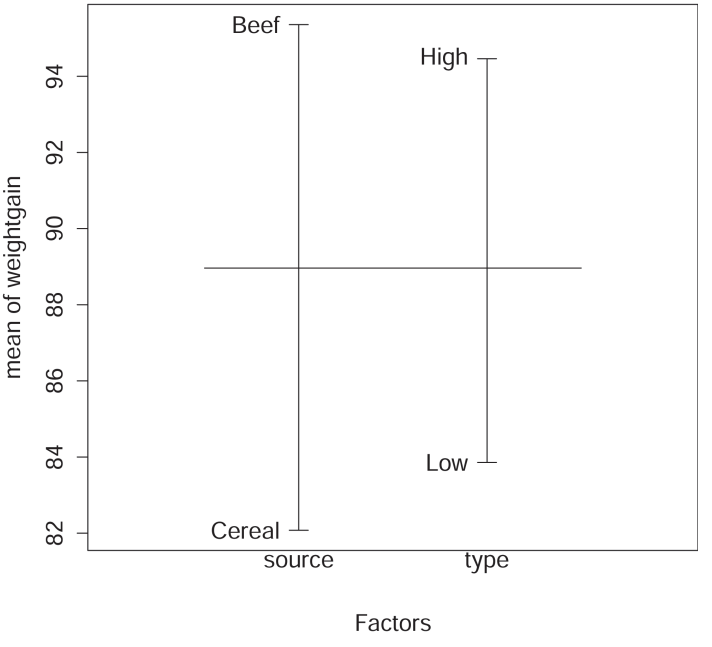


Рисунок 78

сти как минимум некоторых из них. Но даже если мы имеем «достаточно» много данных для надежной оценки большого числа параметров, возникает также проблема с интерпретацией этих параметров – в частности, с интерпретацией взаимодействий высокого порядка (> 2–3). К счастью, однако, в большинстве случаев взаимодействия высокого порядка оказываются незначимыми, что дает нам возможность исключить их из модели. Выбор оптимальных моделей – это отдельная большая тема, к которой мы еще вернемся.

6.7. Контрасты в линейных моделях, содержащих категориальные предикторы

Как было показано выше, однофакторный дисперсионный анализ представляет собой частный случай общей линейной модели, в которой единственный предиктор представлен категориальной переменной (фактором) с двумя или более уровнями. В случае многофакторного дисперсионного анализа имеется несколько интересующих нас факторов. Категориальные предикторы могут быть также включены в модели с количественными предикторами, и тогда мы будем иметь дело с ковариационным анализом.

Важным понятием при работе с категориальными предикторами, которому, к сожалению, уделяется недостаточно внимания в соответствующей методиче-

ской литературе, является понятие «*контрастов*» («*contrasts*»). Ниже мы приводим небольшое введение на эту тему, с примерами применения контрастов в R. Для простоты изложения речь будет идти только об однофакторном дисперсионном анализе.

Основные понятия

Контрасты представляют собой взвешенные суммы средних значений каждого уровня какой-либо категориальной переменной:

$$L = c_1\bar{x}_1 + c_2\bar{x}_2 + ... + c_k\bar{x}_k,$$

где c_1, ..., c_k – весовые коэффициенты для каждого из k уровней изучаемого фактора, а $\bar{x}_1$, ..., $\bar{x}_k$ – групповые средние значения.

В зависимости от того, как заданы их весовые коэффициенты, контрасты позволяют проверять разные гипотезы о средних значениях отдельных уровней или комбинаций уровней анализируемого фактора (факторов). Обычно (но не всегда – см. пример ниже) используют нормированные весовые коэффициенты, то есть $\sum_{i=1}^{k} c_i = 0$. Для этого должны выполняться следующие обязательные условия:

- коэффициенты сравниваемых уровней (или комбинаций уровней) должны иметь разные знаки (+ или –);
- коэффициенты уровней, не представляющих для нас интереса, приравниваются к нулю.

Например, если изучаемый фактор имеет три уровня – A, B и C – и мы хотим «выявить контраст» между средними значениями уровней A и C (то есть сравнить эти значения статистически), то весовым коэффициентам можно придать следующие значения:

```
  [,1]
A   -1
B    0
C    1
```

Аналогично при сравнении средних значений уровней A и B коэффициенты соответствующего контраста составят:

```
  [,1]
A   -1
B    1
C    0
```

Обратите внимание на то, что выбор конкретных значений весовых коэффициентов совершенно произволен – главное, чтобы в сумме они давали 0. Так, в первом из приведенных примеров коэффициенты вполне могли бы составить

```
  [,1]
A -0.5
B  0.0
C  0.5
```

Также обратите внимание на то, что при наличии трех уровней фактора мы можем иметь только два контраста. Это и неудивительно, поскольку, сравнив A с B и A с C, мы, пусть и косвенным образом, но уже сравнили и B с C. Поэтому при попарном сравнении средних значений фактора с тремя уровнями мы получаем следующую *матрицу контрастов*:

```
contr.matrix <- matrix(c(-1, 0, 1, -1, 1, 0), ncol = 2)

rownames(contr.matrix) <- c("A", "B", "C")

contr.matrix
  [,1] [,2]
A   -1   -1
B    0    1
C    1    0
```

При необходимости мы можем сравнивать также *комбинации из нескольких уровней* фактора с другими его уровнями (или комбинациями уровней), что, как несложно догадаться, открывает широкие возможности для проверки самых разнообразных гипотез в отношении анализируемых данных. Так, по тем или иным причинам мы могли бы захотеть сравнить среднее значение уровня A с общим средним значением уровней B и C. Тогда коэффициенты контраста можно было бы выразить следующим образом:

```
  [,1]
A    2
B   -1
C   -1
```

В базовой версии R реализованы четыре основных типа стандартных контрастов:

- *treatment contrasts*: контрасты комбинаций условий;
- *sum contrasts*: контрасты сумм;
- *Helmert contrasts*: контрасты Хелмерта;
- *polynomial contrasts*: полиномиальные контрасты.

Выбор типа контраста зависит от того, является ли фактор упорядоченным («*ordered factors*»), то есть являются ли его уровни естественным образом упорядоченными по возрастанию или убыванию. Например, в отношении ежемесячного дохода можно выделить три уровня: «низкий» < «средний» < «высокий». По умолчанию в R для неупорядоченных факторов применяются так называемые «контрасты комбинаций условий», а для упорядоченных факторов – полиномиальные контрасты. Мы можем выяснить текущие глобальные настройки контрастов при помощи функции **options**():

```
options()$contrasts
        unordered           ordered
"contr.treatment"      "contr.poly"
```

Та же функция позволяет изменить глобальные настройки контрастов:

```
options(contrasts = c("contr.treatment", "contr.poly"))
```

Полиномиальные контрасты применяются гораздо реже других и здесь не рассматриваются. Подробнее об этом, а также о других типах контрастов можно почитать на сайте http://bit.ly/1ebHQ8O (англ. яз.).

Контрасты комбинаций условий (treatment contrasts)

Как отмечено выше, этот тип контрастов используется в R по умолчанию. Собственно, о лежащем в его основе принципе мы уже рассказали в разделе 6.2 при рассмотрении дисперсионного анализа как частного случая общей линейной модели. Применение этого типа контрастов предполагает, что один из уровней изучаемого фактора (например, контрольная группа) принимается в качестве базового, а все остальные уровни сравниваются с этим базовым уровнем.

В качестве примера используем хорошо знакомые нам данные эксперимента по изучению эффективности 6 разных инсектицидов. Рассмотрим еще раз представленную в разделе 3.3 диаграмму размахов переменной `count` для каждого варианта обработки `spray`.

```
boxplot(count ~ spray, data = InsectSprays, col = "coral",
        xlab = "Инсектицид",
        ylab = "Число выживших насекомых")
```

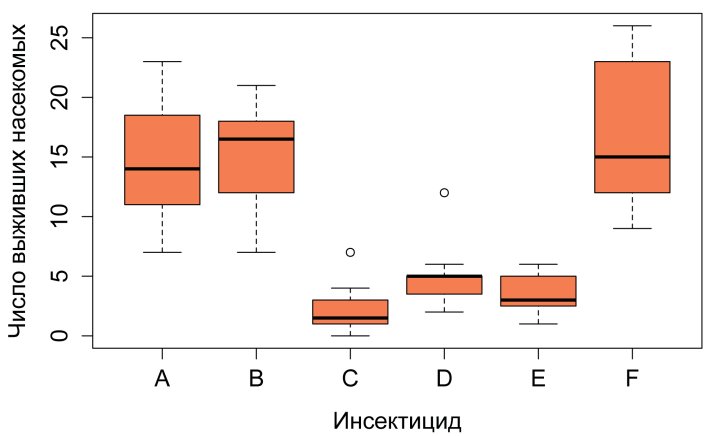


Рисунок 79

В разделах 5.4 и 6.4 мы подробно рассмотрели результаты дисперсионного анализа для проверки нулевой гипотезы об отсутствии разницы между инсектицидами по их эффективности и, поскольку *p*-значение для *F*-критерия было очень низким (`<2e-16`), сделали заключение о значимости различий. Процедура подгонки линейной модели при помощи функции **`lm`**`()` выполнила множественные сравнения и позволила нам выяснить, какие именно пары инсектицидов отличались от базового уровня. Уточним механизм этих сравнений.

При рассмотрении структуры построенной модели обратим внимание на строку:

```
str(M)
     ..$ spray: chr "contr.treatment"
```

Функция **contrasts**() позволяет ознакомиться с матрицей контрастов для того или иного фактора:

```
contrasts(InsectSprays$spray)
  B C D E F
A 0 0 0 0 0
B 1 0 0 0 0
C 0 1 0 0 0
D 0 0 1 0 0
E 0 0 0 1 0
F 0 0 0 0 1
```

Таким образом, мы имеем типичный пример матрицы, содержащей весовые коэффициенты контрастов комбинаций условий ("contr.treatment").

Контрасты комбинаций условий – как раз тот случай, когда все сравнения выполняются относительно базового уровня (в качестве такового автоматически выбрана группа наблюдений для препарата A, поскольку буква A идет первой по алфавиту). Для базового уровня проверяется статистическая гипотеза H_0: $\beta_0 = \mu_1 = 0$, то есть гипотеза о равенстве 0 среднего значения count при использовании препарата A (такая проверка, как правило, особого содержательного смысла не имеет).

Число столбцов приведенной матрицы контрастов равно числу оцениваемых коэффициентов линейной модели, за исключением β_0. Для оценки статистической значимости коэффициента β_1 берут первый столбец матрицы контрастов, умножают его на вектор групповых средних и сравнивают результат по *t*-критерию с β_0. Следовательно, при такой структуре матрицы контрастов проверяется следующая статистическая гипотеза: *средние значения отклика в группах A и B статистически не различаются*. Аналогично проводится тестирование коэффициентов модели β_2–β_5 (то есть отсутствие различий между группой A и каждой из групп C, D, E и F).

Как отмечено выше, суммы весовых коэффициентов по каждому столбцу матрицы контрастов не всегда должны быть равны нулю, и контрасты комбинаций условий – пример такого случая. Следующая таблица обобщает принцип, заложенный в основу контрастов комбинаций условий:

Параметр модели	Оценка	Проверяемая нулевая гипотеза
β_0	Среднее значение для базового уровня $\bar{x}_1$	H_0: $\mu_1 = 0$
β_1	Среднее значение второго уровня за вычетом среднего значения базового уровня: $\bar{x}_2 - \bar{x}_1$	H_0: $\mu_2 - \mu_1 = 0$
β_2	Среднее значение третьего уровня за вычетом среднего значения базового уровня: $\bar{x}_3 - \bar{x}_1$	H_0: $\mu_3 - \mu_1 = 0$
...	...	...
...	...	...

Контрасты сумм (sum contrasts)

При параметризации модели дисперсионного анализа с помощью *контрастов сумм* базовый уровень, с которым сравниваются остальные уровни, представляет собой среднее значение из средних по каждой группе. Разберемся с этим типом контрастов, продолжив пример с инсектицидами.

Используя функцию **constrasts**() в сочетании с **contr.sum**() (подробнее см. ?constr.sum), мы можем изменить исходное поведение R в плане того, какие контрасты программа автоматически применяет в отношении *конкретного* фактора. Еще раз обратите внимание: для изменения глобальных настроек следует использовать функцию **options**() – см. выше.

```
contrasts(InsectSprays$spray) <- contr.sum(n = 6)
```

В итоге получаем следующую матрицу контрастов сумм:

```
contrasts(InsectSprays$spray)
  [,1] [,2] [,3] [,4] [,5]
A    1    0    0    0    0
B    0    1    0    0    0
C    0    0    1    0    0
D    0    0    0    1    0
E    0    0    0    0    1
F   -1   -1   -1   -1   -1
```

Как видим, в отличие от матрицы с контрастами комбинаций условий, суммы по всем столбцам матрицы контрастов сумм равны нулю.

Выполнив дисперсионный анализ, мы увидим, что интерпретация оцененных параметров модели теперь также изменилась:

```
M3 <- lm(count ~ spray, data = InsectSprays)

summary(M3)
Call:
lm(formula = count ~ spray, data = InsectSprays)
Residuals:
   Min     1Q Median     3Q    Max
-8.333 -1.958 -0.500  1.667  9.333
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.5000     0.4622  20.554  < 2e-16 ***
spray1        5.0000     1.0335   4.838 8.22e-06 ***
spray2        5.8333     1.0335   5.644 3.78e-07 ***
spray3       -7.4167     1.0335  -7.176 7.87e-10 ***
spray4       -4.5833     1.0335  -4.435 3.57e-05 ***
spray5       -6.0000     1.0335  -5.805 2.00e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 3.922 on 66 degrees of freedom
Multiple R-squared: 0.7244, Adjusted R-squared: 0.7036
F-statistic:  34.7 on 5 and 66 DF,  p-value: < 2.2e-16
```

Первая строка в таблице с результатами анализа (Intercept) содержит среднее значение числа выживших насекомых, рассчитанное по средним значениям каждой группы. Мы можем легко это проверить, выполнив команду

```
with(InsectSprays, mean(tapply(count, spray, mean)))
[1] 9.5
```

Во второй строке содержится информация о том, насколько среднее число выживших насекомых в следующей группе (B) отличается от среднего значения, вычисленного для всех групп («общее среднее»). Аналогично в третьей строке представлена информация о том, насколько среднее число выживших насекомых в группе C отличается от общего среднего, и т. д. Заметьте: по причинам, описанным выше в подразделе «Определение понятия», в таблице с результатами анализа представлено только пять сравнений с общим средним значением (хотя имеется 6 инсектицидов). Следующая таблица обобщает принцип, заложенный в основу контрастов сумм:

Параметр модели	Оценка	Проверяемая нулевая гипотеза
β_0	Среднее значение из средних по каждой группе («общее среднее»): $\sum_k \bar{x}_k/k$	H_0: $\sum_k \bar{x}_k/k = 0$
β_1	Среднее значение первого уровня за вычетом общего среднего: $\bar{x}_1 - \sum_k \bar{x}_k/k$	H_0: $\bar{x}_1 - \sum_k \bar{x}_k/k = 0$
β_2	Среднее значение второго уровня за вычетом общего среднего: $\bar{x}_2 - \sum_k \bar{x}_k/k$	H_0: $\bar{x}_2 - \sum_k \bar{x}_k/k = 0$
...	...	...
...	...	...

Контрасты Хелмерта

При параметризации модели дисперсионного анализа с помощью *контрастов Хелмерта* базовый уровень представляет собой среднее значение из средних по каждой группе («общее среднее»). Сравнения групповых средних с этим базовым уровнем выполняются согласно схеме, обобщенной в следующей таблице:

Параметр модели	Оценка	Проверяемая нулевая гипотеза
β_0	Среднее значение из средних по каждой группе («общее среднее»): $\sum_k \bar{x}_k/k$	H_0: $\sum_k \bar{x}_k/k = 0$
β_1	$\bar{x}_2 - (\bar{x}_1 + \sum_k \bar{x}_k/k)$	H_0: $\bar{x}_2 - (\bar{x}_1 + \sum_k \bar{x}_k/k) = 0$
β_2	$\bar{x}_3 - (\bar{x}_1 + \bar{x}_2 + \sum_k \bar{x}_k/k)$	H_0: $\bar{x}_3 - (\bar{x}_1 + \bar{x}_2 + \sum_k \bar{x}_k/k) = 0$
...	...	...
...	...	...

Применим контрасты Хелмерта к данным по инсектицидам:

```
# изменение контрастов для фактора spray на контрасты Хелмерта:
contrasts(InsectSprays$spray) <- contr.helmert(n = 6)
```

```
contrasts(InsectSprays$spray)
  [,1] [,2] [,3] [,4] [,5]
A   -1   -1   -1   -1   -1
B    1   -1   -1   -1   -1
C    0    2   -1   -1   -1
D    0    0    3   -1   -1
E    0    0    0    4   -1
F    0    0    0    0    5
```

Обратите внимание: суммы по всем столбцам матрицы контрастов Хелмерта равны нулю. Кроме того, сумма произведений элементов любых двух столбцов матрицы также равна нулю (контрасты с такими свойствами называются *ортогональными*):

```
mat <- contrasts(InsectSprays$spray)
sum(mat[, 1]*mat[, 2])
[1] 0
```

Рассчитаем теперь параметры линейной модели с применением контрастов Хелмерта:

```
M4 <- lm(count ~ spray, data = InsectSprays)

summary(M4)
Call:
lm(formula = count ~ spray, data = InsectSprays)
Residuals:
   Min     1Q Median     3Q    Max
-8.333 -1.958 -0.500  1.667  9.333
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.5000     0.4622  20.554  < 2e-16 ***
spray1        0.4167     0.8006   0.520    0.604
spray2       -4.2778     0.4622  -9.255 1.53e-13 ***
spray3       -1.4306     0.3268  -4.377 4.38e-05 ***
spray4       -1.1417     0.2532  -4.510 2.73e-05 ***
spray5        1.4333     0.2067   6.934 2.12e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 3.922 on 66 degrees of freedom
Multiple R-squared: 0.7244, Adjusted R-squared: 0.7036
F-statistic:  34.7 on 5 and 66 DF,  p-value: < 2.2e-16
```

Интерпретация полученных параметров модели выполняется в соответствии с последней из приведенных выше таблиц. Следует отметить, что, несмотря на такие привлекательные математические свойства, как ортогональность, контрасты Хелмерта редко применяются на практике, поскольку получаемые параметры модели сложно интерпретировать применительно к конкретным проблемам. Тем не менее в некоторых статистических программах этот тип контрастов является

настройкой по умолчанию (например, в S-PLUS – коммерческой реализации языка S, который является предшественником R).

Контрасты, задаваемые пользователем

Помимо рассмотренных стандартных типов, пользователь имеет возможность задавать свои собственные контрасты, предназначенные для проверки определенных гипотез в отношении анализируемых данных.

Продолжая работать с данными по эффективности инсектицидов, предположим, что нас интересуют ответы на три конкретных вопроса (пример заимствован с сайта http://bit.ly/1BbUiLv):

- Различаются ли исследованные препараты по своей эффективности в целом (выше мы уже выяснили, что так оно и есть, но допустим, что пока мы этого не знаем)?
- Различается ли средняя эффективность трех первых инсектицидов (A, B и C) от трех последних (D, E и F)?
- Имеются основания ожидать, что средняя эффективность препаратов A, B и F в среднем ниже (то есть количество выживших насекомых после обработки ими выше), чем эффективность трех остальных препаратов. Так ли это?

Ответ на первый вопрос мы можем получить, применив функцию **aov**() и любые из рассмотренных выше контрастов. А вот ответ на два последних вопроса потребует создания соответствующих пользовательских контрастов:

```
# коэффициенты контраста для ответа на второй вопрос:
con1 <- c(1, 1, 1, -1, -1, -1)
# коэффициенты контраста для ответа на третий вопрос:
con2 <- c(1, 1, -1, -1, -1, 1)
# объединение обоих векторов в матрицу контрастов:
con.matrix <- cbind(con1, con2)
con.matrix
     con1 con2
[1,]    1    1
[2,]    1    1
[3,]    1   -1
[4,]   -1   -1
[5,]   -1   -1
[6,]   -1    1
```

Сообщим программе, что созданная нами матрица контрастов `con.matrix` должна использоваться для всех моделей, включающих фактор `spray`:

```
contrasts(InsectSprays$spray) <- con.matrix
```

Выполним дисперсионный анализ при помощи функции **aov**():

```
M5 <- aov(count ~ spray, data = InsectSprays)
```

Как всегда, результаты анализа можно вывести на экран при помощи функции **summary**(). Однако, учитывая специфичность выполняемых межгрупповых срав-

нений, мы дополнительно воспользуемся аргументом split этой функции (подробнее см. ?summary.aov):

```
summary(M5, split = list(spray =
        list("Первые три против остальных" = 1,
             "ABF против CDE" = 2)))
                                    Df Sum Sq Mean Sq F value Pr(>F)
spray                                5 2668.8   533.8  34.702 <2e-16 ***
  spray: Первые три против остальных  1   93.4    93.4   6.072 0.0164 *
  spray: ABF против CDE               1 2558.7  2558.7 166.349 <2e-16 ***
Residuals                            66 1015.2    15.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Числа 1 и 2 в выражении, задающем значение аргумента split, указывают столбцы матрицы контрастов con.matrix, где содержатся соответствующие весовые коэффициенты.

Из полученных результатов следует, что исследованные инсектициды в целом существенно различаются по своей эффективности (см. первую строку в таблице – Pr(>F) <2e-16). Кроме того, первые три препарата в среднем значительно отличаются по эффективности действия от трех других препаратов (вторая строка: Pr(>F) = 0.0164), а препараты A, B и F в среднем отличаются от группы препаратов C, D и E (третья строка: Pr(>F) <2e-16).

6.8. Проблема множественных проверок статистических гипотез

В практике статистического анализа часто возникает ситуация, когда на *одном и том же* наборе данных выполняется проверка большого числа гипотез. Например, интерес может представлять выполнение всех возможных попарных сравнений средних значений нескольких экспериментальных групп. В других случаях несколько экспериментальных групп могут сравниваться с одной контрольной группой. Особенно большие количества одновременно проверяемых гипотез можно встретить в некоторых областях биологии: так, при работе с данными, которые получают при помощи технологии микрочипов, одновременно проверяются гипотезы в отношении уровней экспрессии нескольких тысяч генов (см., например, статью на сайте http://bit.ly/1xdLDvz).

По определению, при проверке каждой статистической гипотезы закладывается возможность совершения ошибки первого рода (то есть отклонения верной нулевой гипотезы). Чем больше гипотез мы проверяем на одних и тех же данных, тем больше будет вероятность допустить как минимум одну такую ошибку. Это явление называют *эффектом множественных сравнений* («*multiple comparisons*», или «*multiple testing*»). Рассмотрим проблему оценки групповой вероятности ошибки первого рода с математической точки зрения.

Предположим, что мы проверяем истинность m нулевых гипотез, которые можно обозначить как $H_1, H_2, H_3, ..., H_m$. В отношении каждой из этих гипотез мы применяем определенный статистический критерий (например, t-критерий Стьюдента) и делаем заключение о том, верна ли она (то есть выполняем m одинаковых тестов на одних и тех же данных и либо принимаем, либо отвергаем каждую гипотезу). Результаты такого анализа можно свести в таблицу следующего вида:

	Число принятых гипотез	**Число отвергнутых гипотез**	**Всего**
Число верных гипотез	U	V	m_0
Число неверных гипотез	T	S	$m - m_0$
Всего	W	R	m

В этой таблице:

- m_0 – число верных нулевых гипотез («*true null hypotheses*»);
- $m - m_0$ – число истинных альтернативных гипотез («*true alternative hypotheses*»);
- U – число безошибочно принятых гипотез («*true negatives*»);
- V – число ошибочно отвергнутых гипотез («*false positives*») (ошибка первого рода);
- T – число ошибочно принятых гипотез («*false negatives*») (ошибка второго рода);
- S – число безошибочно отвергнутых гипотез;
- W – общее число принятых гипотез;
- R – общее число отвергнутых гипотез.

Представленную таблицу следует интерпретировать следующим образом. В первой строке мы видим, что всего имеется m_0 верных нулевых гипотез. В ходе выполнения анализа определенная их часть (V) ошибочно отвергается, тогда как остальные U гипотез классифицируются правильно. Аналогично во второй строке мы видим, что всего имеется $m - m_0$ альтернативных гипотез, из которых в ходе анализа S гипотез безошибочно отвергаются, а T гипотез ошибочно принимаются. Обратите внимание на то, что общие количества отвергнутых (R) и принятых (W) гипотез нам известны, тогда как m_0, U, V, T и S – ненаблюдаемые случайные величины.

Таким образом, при одновременной проверке группы, или «семейства» («*family*»), статистических гипотез задача заключается в том, чтобы минимизировать число ложных отклонений V и ложных принятий T. Традиционно исследователи пытаются минимизировать величину V. Если $V \geq 1$, мы совершаем как минимум одну ошибку первого рода. Вероятность допущения такой ошибки при множественной проверке гипотез называют «*групповой вероятностью ошибки*» P_F («*familywise error rate*», *FWER*; реже используется термин «*experiment-wise error rate*»). По определению, $P_F = P(V \geq 1)$. Соответственно, когда мы говорим, что хотим *контролировать* групповую вероятность ошибки на определенном уровне значимости α, мы подразумеваем, что должно выполняться неравенство $P_F \leq \alpha$.

Контроль над групповой вероятностью ошибки первого рода позволяют обеспечить *методы множественной проверки гипотез*.

Представим, что у нас имеются три группы, подвергшиеся разным уровням воздействия определенного фактора. Как выяснить влияние этого фактора на интересующую нас переменную-отклик? При выполнении определенных условий данную задачу можно решить, сравнив средние значения имеющихся групп при помощи однофакторного дисперсионного анализа. Однако дисперсионный анализ позволит сделать только вывод типа «да/нет», то есть эффект изучаемого фактора либо имеется, либо отсутствует. Допустим, что результаты дисперсионного анализа указывают на наличие эффекта. Как теперь выяснить, какие именно группы различаются между собой?

Одним из (очень гибких) способов ответить на этот вопрос является использование линейных контрастов, описанное выше. Эта концепция мало знакома многим исследователям, но зато им хорошо знаком t-критерий Стьюдента, который легко рассчитать для каждой пары сравниваемых групп. Выполняя тест Стьюдента, исследователь хочет проверить нулевую гипотезу об отсутствии разницы между генеральными средними сравниваемых групп A и B с вероятностью ошибки менее 5%. Точно такой же риск ошибиться он устанавливает и при сравнении B с C и A с C (рис. 80).

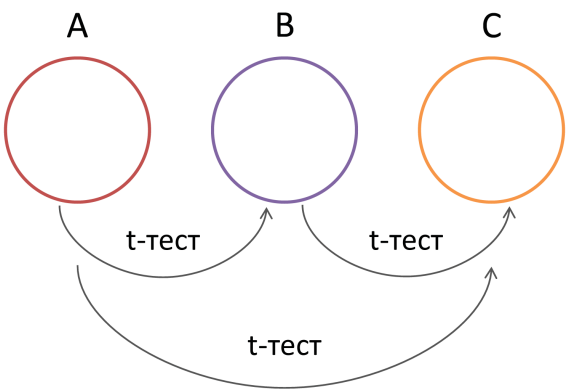


Рисунок 80

Принципиальная проблема, возникающая при выполнении подобных множественных сравнений, заключается в том, что в действительности вероятность ошибки первого рода может значительно превышать принятое критическое значение 5%. Так, для рассматриваемого примера вероятность ошибиться хотя бы в одном из трех сравнений составит $P_F = 1 - (1 - 0.05)^m = 1 - (1 - 0.05)^3 = 0.143$. Очевидно, что увеличение числа одновременно проверяемых гипотез будет неизбежно сопровождаться дальнейшим возрастанием P_F.

Для устранения эффекта множественных сравнений существует большой арсенал методов, обеспечивающих контроль над групповой вероятностью ошибки

P_F на определенном уровне α, но различающихся по своей мощности. Рассмотрим наиболее распространенные из этих методов.

Поправка Бонферрони

Метод Бонферрони, названный так в честь предложившего его итальянского математика К. Бонферрони (Carlo Emilio Boferroni), является одним из наиболее простых и известных способов контроля над групповой вероятностью ошибки.

Предположим, что мы применили определенный статистический критерий 3 раза (например, сравнили при помощи критерия Стьюдента средние значения групп A и B, A и C, и B и C – см. рис. 80) и получили следующие три *p*-значения: 0.01, 0.02 и 0.005. Если мы хотим, чтобы групповая вероятность ошибки при этом не превышала определенного уровня значимости α (например, 0.05), то, согласно методу Бонферрони, мы должны сравнить каждое из полученных *p*-значений не с α, а с α/m, где m – число проверяемых гипотез. Деление исходного уровня значимости α на m – это и есть *поправка Бонферрони*. В рассматриваемом примере каждое из полученных *p*-значений необходимо было бы сравнить с $0.05/3 = 0.017$. В результате мы выяснили бы, что *p*-значение для второй гипотезы (0.02) превышает 0.017, и, соответственно, у нас не было бы оснований отклонить эту гипотезу.

Вместо деления изначально принятого уровня значимости на число проверяемых гипотез мы могли бы умножить каждое из исходных *p*-значений на это число. Сравнив такие скорректированные *p*-значения («*adjusted p-values*»; обычно обозначаются буквой q) с α, мы пришли бы к точно тем же выводам:

- $0.01 \times 3 = 0.03 < 0.05$: гипотеза отклоняется;
- $0.02 \times 3 = 0.06 > 0.05$: гипотеза принимается;
- $0.005 \times 3 = 0.015 < 0.05$: гипотеза отклоняется.

В ряде случаев при умножении исходных *p*-значений на m результат может превысить 1. По определению, вероятность не может быть больше 1, и если это происходит, то получаемое значение просто приравнивают к 1.

В базовой версии R имеется функция **p.adjust**(), которая позволяет легко применять как метод Бонферрони, так и ряд других методов, обеспечивающих контроль групповой вероятности ошибки на определенном уровне (задаются при помощи аргумента method). При реализации метода Бонферрони с помощью этой функции применяется второй из описанных выше подходов, то есть исходные *p*-значения умножаются на число проверяемых гипотез. Функция **p.adjust**() принимает на входе вектор с исходными *p*-значениями и возвращает скорректированные значения:

```
# Скорректированные p-значения:
p.adjust(c(0.01, 0.02, 0.005), method = "bonferroni")
[1] 0.030 0.060 0.015

# Какие из проверяемых гипотез следует отвергнуть?
alpha <- 0.05
p.adjust(c(0.01, 0.02, 0.005), method = "bonferroni") < alpha
[1]  TRUE FALSE  TRUE
```

Хотя метод Бонферрони очень прост в реализации, он обладает одним существенным недостатком: при возрастании числа проверяемых гипотез мощность этого метода резко снижается. Другими словами, при возрастании числа гипотез нам будет все сложнее и сложнее отвернуть многие из них, даже если они неверны и должны быть отвергнуты. Например, при проверке 10 гипотез применение поправки Бонферрони привело бы к снижению исходного уровня значимости до $0.05/10 = 0.005$. Соответственно, для отклонения той или иной гипотезы *все* соответствующие p-значения должны были бы оказаться меньше 0.005, что случалось бы нечасто. В связи с этим метод Бонферрони не рекомендуется использовать в случаях, когда число проверяемых гипотез превышает 7–8.

Метод Холма

Для преодоления проблем, связанных с низкой мощностью метода Бонферрони, С. Холм (Holm, 1978, http://bit.ly/1N8mu7z) предложил гораздо более мощную его модификацию (часто этот метод называют еще методом Холма-Бонферрони). Этот модифицированный метод основан на последовательном алгоритме проверки групповой гипотезы, который включает следующие шаги:

- исходные p-значения упорядочивают по возрастанию: $p(1) \le p(2) \le ... \le \le p(m)$. Эти p-значения соответствуют проверяемым гипотезам $H(1)$, $H(2)$, ..., $H(m)$;
- если $p(1) \ge \alpha/m$, все нулевые гипотезы $H(1)$, $H(2)$, ..., $H(m)$ принимаются, и процедура останавливается. Иначе следует отвергнуть гипотезу $H(1)$ и продолжить;
- если $p(2) \ge \alpha/(m-1)$, нулевые гипотезы $H(2)$, $H(3)$, ..., $H(m)$ принимаются, и процедура останавливается. Иначе гипотеза $H(2)$ отвергается, и процедура продолжается;
- ...
- если $p(m) \ge \alpha$, нулевая гипотеза $H(m)$ принимается, и процедура останавливается.

Описанную процедуру называют *нисходящей* («*step-down*»): она начинается с наименьшего p-значения в упорядоченном ряду и последовательно «спускается» вниз к более высоким значениям. На каждом шаге соответствующее значение $p(i)$ сравнивается со скорректированным уровнем значимости $\alpha/(m+i-1)$.

Как и в случае с поправкой Бонферрони, вместо корректировки уровня значимости мы можем скорректировать непосредственно p-значения – конечный результат (в смысле принятия или отклонения той или иной гипотезы) окажется идентичным. Соответствующая поправка выполняется в виде $q_i = p(i)(m+i-1)$. Так, для рассмотренного выше примера с тремя p-значениями получаем:

- $q_1 = p(1)(m-1+1) = 0.005 \times 3 = 0.015$;
- $q_2 = p(2)(m-2+1) = 0.01 \times 2 = 0.02$;
- $q_3 = p(3)(m-3+1) = 0.02 \times 1 = 0.02$.

Именно последний подход реализован в R-функции **p.adjust**():

```
# Скорректированные p-значения:
p.adjust(c(0.01, 0.02, 0.005), method = "holm")
```

```
[1] 0.020 0.020 0.015
# (обратите внимание: функция автоматически упорядочивает
# итоговые p-значения по убыванию)

# Какие из проверяемых гипотез следует отвергнуть?
p.adjust(c(0.01, 0.02, 0.005), method = "holm") < alpha
[1] TRUE TRUE TRUE
```

Сравните результаты, полученные при помощи методов Бонферрони и Холма: в последнем случае мы отвергаем все три гипотезы, тогда как при использовании поправки Бонферрони отклонены были бы только две из трех проверяемых гипотез. Этот простой пример указывает на то, что метод Холма обладает большей мощностью, чем метод Бонферрони.

Метод Беньямини-Хохберга

Сегодня проверка действительно большого числа гипотез (десятков тысяч и даже миллионов) стала рутинной операцией в самых разных областях, таких как генетика (анализ данных, получаемых при помощи технологии микрочипов), протеомика (данные масс-спектрометрии), нейробиология (анализ изображений мозга), экология, астрофизика, и др. Недостаточная мощность традиционных процедур множественной проверки гипотез приводит к тому, что при больших значениях m критический уровень значимости становится очень низким и многие нулевые гипотезы, которые следовало бы отвергнуть, не отвергаются. В итоге исследователь может пропустить интересные «открытия» (*«discoveries»*), достойные более подробного изучения. Например, в ходе сравнения уровней экспрессии генов у больных и здоровых испытуемых результаты для некоторых потенциально важных генов могли бы оказаться ложно-отрицательными.

В 1995 г. израильские исследователи И. Беньямини и Й. Хохберг (Benjamini & Hochberg, 1995, http://bit.ly/1CSvr5x) опубликовали статью, в которой был предложен принципиально иной подход к проблеме множественных проверок статистических гипотез (эта работа входит в список 25 наиболее цитируемых статей по статистике – см. http://bit.ly/1FS4HBh). Суть предложенного подхода заключается в том, что вместо контроля над групповой вероятностью ошибки первого рода выполняется контроль над так называемой ожидаемой *долей ложных отклонений* (*«false discovery rate»*, *FDR*) из числа R всех отклоненных гипотез:

$$FDR = \mathbf{E}[V/R],$$

где V – число ошибочно отвергнутых гипотез (см. таблицу в разделе 6.8).

В отличие от уровня значимости α, каких-либо «общепринятых» значений *FDR* не существует. Многие исследователи по аналогии контролируют *FDR* на уровне 5%. В генетических исследованиях часто встречается также уровень 10%. Интерпретация порогового значения *FDR* очень проста: например, если в ходе анализа данных отклонено 1000 гипотез, то при $q = 0.10$ ожидаемая доля ложно отклоненных гипотез не превысит 100. Процедура Беньямини-Хохберга сводится к следующему:

- исходные p-значения упорядочивают по возрастанию: $p(1) \le p(2) \le ... \le p(m)$. Пусть $H(i)$ обозначает нулевую гипотезу, которой соответствует i-е значение в этом упорядоченном ряду – $p(i)$;
- находят максимальное значение k среди всех индексов $i = 1, 2, ..., m$, для которого выполняется неравенство $p(i) \le qi/m$;
- отклоняют все гипотезы $H(i)$ с индексами $i = 1, 2, ..., k$.

В качестве примера рассмотрим следующий ряд из 15 упорядоченных по возрастанию p-значений из оригинальной статьи Benjamini & Hochberg (1995):

```
0.0001, 0.0004, 0.0019, 0.0095, 0.0201, 0.0278, 0.0298,
0.0344, 0.0459, 0.3240, 0.4262, 0.5719, 0.6528, 0.7590, 1.000
```

Если бы мы осуществили контроль над групповой вероятностью ошибки, применив, например, поправку Бонферрони для уровня значимости 0.05 (то есть $0.05/15 = 0.0033$), то отклоненными оказались бы три гипотезы, которым соответствуют первые три p-значения.

При контроле над ожидаемой долей ложных отклонений на уровне 5% мы сравниваем каждое значение $p(i)$ с $0.05i/15$, начиная с самого высокого – $p(15)$. В итоге мы увидим, что первым p-значением, соответствующим указанному ограничению (5%), является $p(4)$:

$$p(4) = 0.0095 \le (4/15)0.05 = 0.013.$$

Теперь мы отклоняем четыре гипотезы, которым соответствуют первые четыре p-значения в приведенном выше ряду, поскольку все эти значения не превышают 0.013.

Функция **p.adjust**() реализует не только описанные выше метод Холма и поправку Бонферрони, но и процедуру Беньямини-Хохберга. Для этого на вход функции подается вектор с исходными p-значениями и аргументу `method` присваивается значение `"BH"` (от «*Benjamini-Hochberg*») или значение-синоним `"fdr"` («*false discovery rate*»).

Контроль над *FDR* при помощи этой функции выполняется способом, несколько отличным от описанного выше. В частности, вместо нахождения максимального индекса k исходные p-значения корректируются по формуле $q(i) = p(i)m/i$. Например, для первых двух p-значений из приведенного выше примера мы получили бы:

- $(0.0001 \times 15)/1 = 0.0015$;
- $(0.0004 \times 15/2) = 0.003$.

Воспользовавшись функцией **p.adjust**() для всех p-значений из рассмотренного примера, получим:

```
pvals <- c(0.0001, 0.0004, 0.0019, 0.0095,  0.0201,
           0.0278, 0.0298, 0.0344, 0.0459, 0.3240,
           0.4262, 0.5719, 0.6528, 0.7590, 1.000)

p.adjust(pvals, method = "BH")
```

```
 [1] 0.00150000 0.00300000 0.00950000 0.03562500 0.06030000
 [6] 0.06385714 0.06385714 0.06450000 0.07650000 0.48600000
[11] 0.58118182 0.71487500 0.75323077 0.81321429 1.00000000
```

Интерпретация этих скорректированных *p*-значений (в большинстве литературных источников их называют *q*-значениями) такова:

- допустим, что мы хотим контролировать долю ложно отклоненных гипотез на уровне $FDR = 0.05$;
- все гипотезы, *q*-значения которых ≤ 0.05, отклоняются;
- среди всех этих отклоненных гипотез доля ошибочно отклоненных не превышает 5%.

Как видим, в рассмотренном примере конечный результат после коррекции исходных *p*-значений при помощи функции **p.adjust**() идентичен тому, что был получен при использовании оригинальной процедуры Беньямини-Хохберга: на уровне 5% отклоняются первые четыре гипотезы.

Коррекция *p*-значений по методу Беньямини-Хохберга работает особенно хорошо в ситуациях, когда необходимо принять *общее решение* по какому-либо вопросу при наличии информации (то есть проверенных гипотез) *по многим параметрам* (Benjamini & Hochberg, 1995).

Типичным примером будет одновременный анализ многих биологических параметров (вес и температура тела, клеточные показатели крови и т. п.) в группе пациентов, которых лечили новым препаратом, в сравнении с группой, которой давали плацебо. Средние значения каждого параметра в этих группах можно было бы сравнить, например, при помощи *t*-теста или какого-либо из его непараметрических аналогов. В итоге в распоряжении исследователя оказалось бы большое число соответствующих *p*-значений.

Общий вывод, который исследователь хотел бы сделать, состоит в том, что новый препарат оказывает положительное влияние на исход лечения. Конечно, в такой ситуации исследователь был бы заинтересован в обнаружении максимально большого числа параметров, по которым экспериментальные группы различаются (контролируя при этом долю ложных заключений на определенном уровне). Контроль над групповой вероятностью ошибки в этом случае оказался бы слишком строгим, тогда как более мощный метод Беньямини-Хохберга допускает наличие определенной доли ложных отклонений среди всех отклоненных гипотез и тем самым способствует общему положительному выводу по поводу эффективности нового препарата.

Следует отметить, что описанный здесь метод контроля над ожидаемой долей ложных отклонений предполагает, что все тесты, при помощи которых получают *p*-значения, независимы (Benjamini & Hochberg, 1995). На практике, однако, в большинстве случаев это условие выполняться не будет: многие биологические параметры в вышеприведенном примере были измерены у одних и тех же испытуемых, что вносит определенный уровень корреляции между соответствующими тестами.

Метод Беньямини-Йекутили

Понимая важность и одновременно неосуществимость на практике предположения о независимости всех проверяемых гипотез, И. Беньямини и Д. Йекутили (Benjamini & Yekutieli, 2001; http://bit.ly/1xsGkZK) предложили усовершенствованный метод, учитывающий наличие корреляции между проверяемыми гипотезами (подробное описание метода и соответствующие доказательства можно найти в оригинальной статье).

Процедура Беньямини-Йекутили очень похожа на процедуру Беньямини-Хохберга. Основное отличие заключается во введении поправочной константы $c(m) = \sum_{i=1}^{m} \frac{1}{i}$:

- исходные p-значения упорядочивают по возрастанию: $p(1) \le p(2) \le ... \le \le p(m)$. Пусть $H(i)$ обозначает нулевую гипотезу, которой соответствует i-е значение в этом упорядоченном ряду, то есть $p(i)$;
- находят максимальное значение k среди всех индексов $i = 1, 2, ..., m$, для которого выполняется неравенство $p(i) \le (i/m) \times qc(m)$;
- отклоняют все гипотезы $H(i)$ с индексами $i = 1, 2, ... , k$.

В качестве примера используем приведенный ранее ряд из 15 упорядоченных по возрастанию p-значений (Benjamini & Hochberg, 1995):

```
0.0001, 0.0004, 0.0019, 0.0095, 0.0201, 0.0278, 0.0298,
0.0344, 0.0459, 0.3240, 0.4262, 0.5719, 0.6528, 0.7590, 1.000
```

При контроле над ожидаемой долей ложных отклонений на уровне 5% мы сравниваем каждое значение $p(i)$ с величиной $i/15 \times 0.05/c(m)$, начиная с самого высокого, то есть $p(15)$. В данном случае константа $c(m) = \sum_{i=1}^{15} 1/i = 3.31823$. В итоге мы увидим, что первым p-значением, соответствующим указанному ограничению (5%), является $p(3)$:

$$p(3) = 0.0019 \le (3/15) \times (0.05/3.31823) = 0.003014.$$

Таким образом, следует отклонить три гипотезы, которым соответствуют первые три p-значения в приведенном выше ряду, поскольку все эти значения не превышают 0.003014.

Контроль над ожидаемой долей ложных отклонений по методу Беньямини-Йекутили можно реализовать при помощи функции **`p.adjust()`**, подав на ее вход вектор с исходными p-значениями и присвоив аргументу `method` значение `"BY"` (от «*Benjamini-Yekutieli*»). При этом контроль над *FDR* с помощью этой функции выполняется способом, несколько отличным от описанной выше процедуры: вместо нахождения максимального индекса k выполняется корректировка исходных p-значений по формуле:

$$q(i) = p(i) \times m \times c(m)/i.$$

Так, для первых двух p-значений из приведенного выше примера мы получили бы:

- $(0.0001 \times 15 \times 3.31823)/1 = 0.004977$;
- $(0.0004 \times 15 \times 3.31823/2) = 0.009955$.

Воспользуемся функцией **p.adjust**() для преобразования всех *p*-значений из рассмотренного примера:

```
pvals <- c(0.0001, 0.0004, 0.0019, 0.0095,  0.0201,
           0.0278, 0.0298, 0.0344, 0.0459, 0.3240,
           0.4262, 0.5719, 0.6528, 0.7590, 1.000)

p.adjust(pvals, "BY")
 [1] 0.004977 0.009954 0.031523 0.118211 0.200089 0.211892
 [7] 0.211892 0.214025 0.253844 1.000000 1.000000 1.000000
[13] 1.000000 1.000000 1.000000
```

Интерпретация скорректированных *p*-значений выполняется тем же образом, что и в случае с методом Беньямини-Хохберга.

Кроме описанных четырех методов, функция **p.adjust**() предлагает также корректировку *p*-значений с использованием процедур Хохберга (не путать с методом Беньямини-Хохберга) и Хоммеля («*Hommel's method*»; подробнее см. обзор на сайте http://bit.ly/1bp2ORc). Контроль над ожидаемой долей ложных отклонений с использованием описанных процедур Беньямини-Хохберга-Йекутили можно выполнять также при помощи функций, входящих в состав специализированных пакетов, список которых можно найти, например, на сайте Лондонского колледжа http://bit.ly/1Ht4WBi.

6.9. Апостериорные сравнения групповых средних

Применяя однофакторный дисперсионный анализ, мы можем проверить нулевую гипотезу о том, что все сравниваемые группы происходят из одной генеральной совокупности и, следовательно, их средние значения не различаются, то есть H_0: $\mu_1 = \mu_2 = ... = \mu_m$. Если нулевую гипотезу не удается отвергнуть при заданном уровне значимости (например, $\alpha = 0.05$), то в дальнейшем анализе, в принципе, нет необходимости. Если же нулевая гипотеза отвергается, то *единственный вывод*, который можно сделать при помощи классического дисперсионного анализа, заключается в том, что изучаемый фактор оказывает существенное влияние на интересующую нас переменную.

После установления существенной разницы между группами в целом интересно пойти дальше и выполнить так называемый *апостериорный анализ*, то есть выяснить, какие именно группы статистически значимо отличаются друг от друга. Как мы узнали в предыдущем разделе, при выполнении таких попарных сравнений необходимо каким-то образом обеспечить контроль над групповой вероятностью ошибки 1-го рода. Метод Бонферрони в ряде случаев может оказаться вполне подходящим для этого. Однако при наличии большого числа сравниваемых групп он становится очень консервативным.

Специально для дисперсионного анализа разработан целый ряд апостериорных тестов (например, в пакете SPSS их насчитывается не менее 18). В список наиболее известных методов входят: обнаружение наименьшего значения значимой разности («*LSD test*»), критерий Дункана («*Dunkan's test*»), метод Стьюдента-Ньюмана-Кейлса («*Student-Newman-Keuls test*»), проверка достоверно значимой разности Тьюки («*Tukey HSD test*»), модифицированный метод LSD, критерий Шеффе («*Scheffe's test*»), и др. Здесь они перечислены в порядке снижения их мощности (или увеличения консервативности), хотя на этот счет в литературе существуют довольно противоречивые рекомендации.

Критерий Тьюки

Наиболее распространенным и рекомендуемым в литературе является *критерий достоверно значимой разности* HSD, названный в честь предложившего его американского математика и статистика Дж. Тьюки («*Tukey's honestly significant difference test*», или просто «*Tukey's HSD test*»). HSD-тест задает наименьшую величину разности математических ожиданий в группах, которую можно считать значимой, а также позволяет рассчитать ее доверительные интервалы с учетом числа выполняемых сравнений.

Критерий Тьюки используется для проверки нулевой гипотезы H_0: $\mu_B = \mu_A$ против альтернативной гипотезы H_1: $\mu_B \neq \mu_A$, где индексы A и B обозначают любые две сравниваемые группы. При наличии m групп всего возможно выполнить $m(m-1)/2$ попарных сравнений.

Первый шаг заключается в упорядочивании всех имеющихся групповых средних значений по возрастанию (от 1 до m). Далее выполняют попарные сравнения этих средних так, что сначала сравнивают наибольшее среднее с наименьшим, то есть m-ое с первым, затем m-ое со вторым, третьим и т. д. вплоть до $(m-1)$-го. Затем предпоследнее среднее, $(m-1)$-ое, тем же образом сравнивают с первым, вторым и т. д. до $(m-2)$-го. Эти сравнения продолжаются до тех пор, пока не будут перебраны все пары.

Указанные сравнения выполняются при помощи критерия Тьюки, который представляет собой несколько модифицированный критерий Стьюдента:

$$q = (\bar{x}_B - \bar{x}_A)/SE.$$

Отличие от критерия Стьюдента заключается в том, как рассчитывается стандартная ошибка SE:

$$SE = \sqrt{MS_w / n},$$

где MS_w – рассчитываемая в ходе дисперсионного анализа *внутригрупповая дисперсия*.

Приведенная формула для критерия Тьюки верна для случаев, когда все сравниваемые группы содержат одинаковое число наблюдений n. Если сравниваемые группы неодинаковы по размеру, стандартная ошибка будет рассчитываться следующим образом:

$$SE = \sqrt{\frac{MS_w}{2}\left(\frac{1}{n_A} + \frac{1}{n_B}\right)}.$$

Благодаря тому обстоятельству, что в приведенные выше формулы стандартной ошибки входит внутригрупповая дисперсия MS_w, критерий Тьюки становится подходящим критерием для выполнения большого числа парных сравнений групповых средних. Проверяемые нулевые гипотезы принимают или отвергают либо путем сравнения получаемых значений критерия q с определенным критическим значением для выбранного уровня значимости, либо рассчитывают соответствующие p-значения (подробнее см. примеры для критерия Стьюдента).

В среде R множественные сравнения групповых средних при помощи теста Тьюки можно выполнить с использованием функции `TukeyHSD()`, входящей в базовую версию системы. В качестве примера используем данные по содержанию стронция (мг/мл) в пяти водоемах США (Zar, 2010):

```
waterbodies <- data.frame(Water = rep(c("Grayson", "Beaver",
                                        "Angler", "Appletree",
                                        "Rock"), each = 6),
                          Sr = c(28.2, 33.2, 36.4, 34.6, 29.1, 31.0,
                                 39.6, 40.8, 37.9, 37.1, 43.6, 42.4,
                                 46.3, 42.1, 43.5, 48.8, 43.7, 40.1,
                                 41.0, 44.1, 46.4, 40.2, 38.6, 36.3,
                                 56.3, 54.1, 59.4, 62.7, 60.0, 57.3)
                                )
```

На рис. 81 ниже эти данные представлены графически.

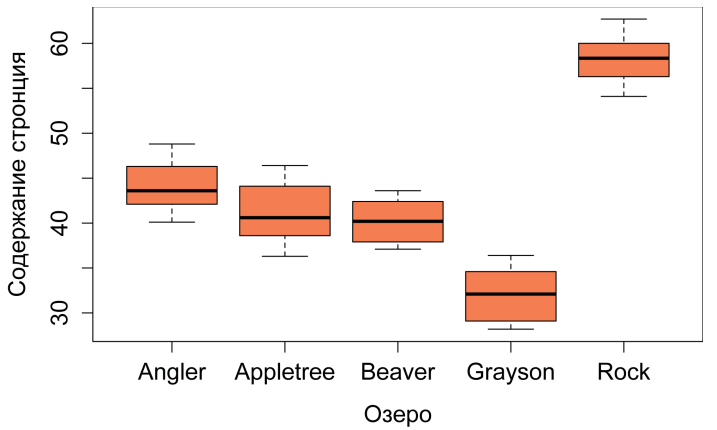


Рисунок 81

Необходимо выяснить: *а*) есть ли существенные различия между этими водоемами по содержанию стронция в целом и если есть, то *б*) какие именно во-

доемы отличаются друг от друга. Для ответа на первый вопрос выполним дисперсионный анализ при помощи функции **aov**():

```
M <- aov(Sr ~ Water, data = waterbodies)
summary(M)
            Df Sum Sq Mean Sq F value   Pr(>F)
Water        4 2193.4   548.4   56.16 3.95e-12 ***
Residuals   25  244.1     9.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Как видно из полученных результатов, обследованные водоемы статистически значимо различаются по содержанию стронция. Чтобы выяснить, где именно лежат эти различия, достаточно подать объект M на функцию **TukeyHSD**():

```
TukeyHSD(M)
  Tukey multiple comparisons of means
    95% family-wise confidence level
Fit: aov(formula = Sr ~ Water, data = waterbodies)
$Water
                         diff        lwr       upr     p adj
Appletree-Angler   -2.9833333  -8.281979  2.315312 0.4791100
Beaver-Angler      -3.8500000  -9.148645  1.448645 0.2376217
Grayson-Angler    -12.0000000 -17.298645 -6.701355 0.0000053
Rock-Angler        14.2166667   8.918021 19.515312 0.0000003
Beaver-Appletree   -0.8666667  -6.165312  4.431979 0.9884803
Grayson-Appletree  -9.0166667 -14.315312 -3.718021 0.0003339
Rock-Appletree     17.2000000  11.901355 22.498645 0.0000000
Grayson-Beaver     -8.1500000 -13.448645 -2.851355 0.0011293
Rock-Beaver        18.0666667  12.768021 23.365312 0.0000000
Rock-Grayson       26.2166667  20.918021 31.515312 0.0000000
```

В первом столбце полученной таблицы перечислены пары сравниваемых водоемов. Во втором столбце содержатся разности между соответствующими групповыми средними. Третий и четвертый столбцы содержат значения нижнего (lwr) и верхнего (upr) 95%-ных доверительных пределов для соответствующих разностей. Наконец, в пятом столбце представлены *p*-значения для каждой из сравниваемых пар водоемов.

Хорошо видно, что существенной разницы в парах «Appletree-Angler», «Beaver-Angler» и «Beaver-Appletree» нет ($p > 0.05$), тогда как во всех остальных случаях разница статистически значима. В целом полученные результаты хорошо согласуются с визуальной оценкой различий, которую можно сделать, глядя на приведенную выше диаграмму размахов (рис. 81).

Результаты парных сравнений групповых средних можно легко изобразить на графике (рис. 82):

```
par(mar = c(4.5, 8, 4.5, 1.5))
plot(TukeyHSD(M), las = 1)
```

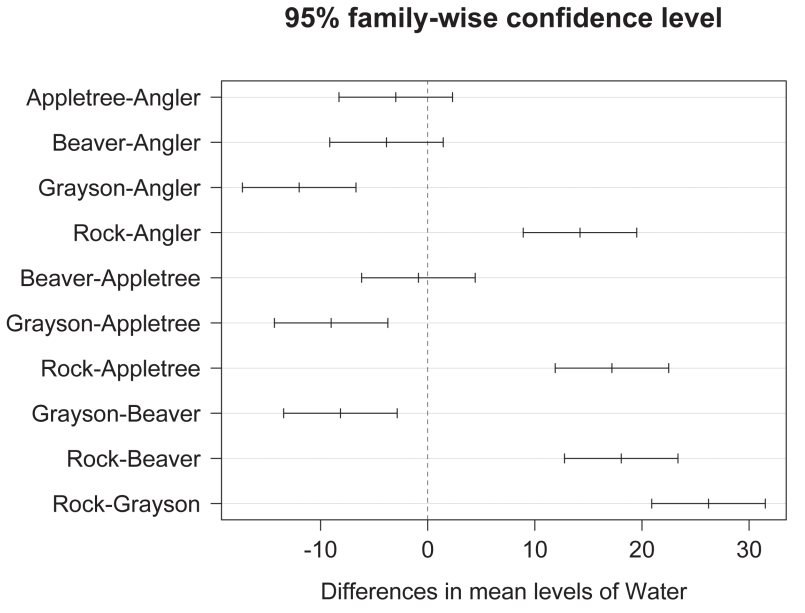


Рисунок 82

На рис. 82 приведены разности между групповыми средними (`Differences in mean levels of Water`) и их доверительные интервалы, рассчитанные с учетом контроля над групповой вероятностью ошибки (`95% family-wise confidence level`). В трех случаях доверительные интервалы включают 0, что указывает на отсутствие различий между соответствующими группами (сравните с *p*-значениями выше).

Критерий Тьюки имеет те же условия применимости, что и собственно дисперсионный анализ, то есть *нормальность распределения* данных и (особенно важно!) *однородность групповых дисперсий* (подробнее см. раздел 6.4). Устойчивость к отклонению от этих условий, равно как и статистическая мощность критерия Тьюки, возрастают при одинаковом числе наблюдений во всех сравниваемых группах (Zar, 2010).

Методы множественных проверок гипотез, реализованные в пакете `multcomp`

Многообразие описанных выше методов множественных проверок статистических гипотез может создать ощущение неразберихи и привести в замешательство даже опытных исследователей. Тем не менее между многими методами существует большое сходство. Более того, можно показать, что некоторые методы, известные и используемые под разными названиями и для разных целей, с математической точки зрения эквиваленты (например, тесты Тьюки и Даннета).

Используя теорию общих линейных моделей, профессор Ф. Бретц и соавторы (Bretz et al., 2010, http://bit.ly/19QCLC1) разработали общую методологическую

схему, объединяющую большинство классических критериев для множественной проверки гипотез. Как это часто происходит в наши дни, соответствующие методологические подходы были реализованы в дополнительном пакете для R – `multcomp` (от «*multiple comparisons*» – множественные сравнения). Ниже дается описание основных возможностей этого пакета. Заинтересованные читатели найдут подробные математические выкладки и множество примеров R-кода в упомянутой выше книге (Bretz et al., 2010).

В качестве примера используем данные `waterbodies` по содержанию стронция в воде пяти водоемов США, представленные в предыдущем разделе. Предположим, что наша задача заключается в выявлении различий между исследованными водоемами (фактор `Water`) по содержанию стронция. Решение поставленной задачи сводится к оценке параметров следующей линейной модели:

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon_i,$$

где y_i – это i-е значение содержания стронция в воде; β_0 – среднее значение содержания стронция для базового уровня изучаемого фактора (в данном случае выбор базового уровня совершенно произволен; по умолчанию R выберет в качестве базового тот уровень, название которого идет первым по алфавиту, – `Angler`); β_1, ..., β_4 – коэффициенты, отражающие разницу между средним значением содержания стронция в воде «базового водоема» и средними значениями в других водоемах; ε_i – нормально распределенные остатки. Параметры модели легко оценить с помощью функции **`lm`**():

```
M  <- lm(Sr ~ Water, data = waterbodies)
summary(M)
Call:
lm(formula = Sr ~ Water, data = waterbodies)
Residuals:
    Min      1Q  Median      3Q     Max
-4.8000 -2.2500 -0.4833  2.2042  5.3000
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)     44.083      1.276  34.555  < 2e-16 ***
WaterAppletree  -2.983      1.804  -1.654   0.1107
WaterBeaver     -3.850      1.804  -2.134   0.0428 *
WaterGrayson   -12.000      1.804  -6.651 5.72e-07 ***
WaterRock       14.217      1.804   7.880 3.09e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.125 on 25 degrees of freedom
Multiple R-squared:  0.8998, Adjusted R-squared:  0.8838
F-statistic: 56.15 on 4 and 25 DF,  p-value: 3.948e-12
```

Как видим, в целом средние уровни содержания стронция в воде исследованных водоемов значимо различаются (см. последнюю строку результатов: p-значение

для *F*-критерия оказалось существенно меньше 0.05, а именно `3.948e-12`). Кроме того, из полученных результатов мы можем сделать определенные выводы касательно того, какие из исследованных водоемов отличались от водоема `Angler`, автоматически выбранного программой в качестве «эталона» для сравнений. В частности, есть основания считать, что параметры β_2, β_3 и β_4 статистически значимо (на уровне 0.05) отличаются от 0. Другими словами, мы можем утверждать, что среднее содержание стронция в «эталонном» водоеме `Angler` значимо отличалось от таковых в `Beaver` (`p = 0.0428`), `Grayson` (`p = 5.72e-07`) и `Rock` (`p = 3.09e-08`).

К сожалению, такая интерпретация *p*-значений, полученных для параметров модели `M`, имеет один существенный недостаток: они являются *безусловными* («*marginal*»; сравните с «*условным распределением вероятностей*» – «*conditional probability distribution*»). Такое название связано с тем, что эти *p*-значения рассчитываются, исходя из допущения об отсутствии какой-либо связи между параметрами модели. Иными словами, параметры модели рассматриваются как некоррелирующие случайные величины, каждая из которых имеет свое собственное распределение вероятностей.

Проблема, однако, заключается в том, что мы не можем строго утверждать об отсутствии корреляции между оцениваемыми параметрами, поскольку при построении модели используется *один конкретный набор данных*. Как следствие, проверяя одновременно несколько статистических гипотез в отношении параметров модели на одних и тех же данных, мы не выполняем должного контроля над групповой вероятностью ошибки первого рода. Это, в свою очередь, может привести к повышенной вероятности того, что выводы, основанные на таком анализе, не подтвердятся в будущем при сборе дополнительных данных (например, при повторении эксперимента). В ряде областей слабая воспроизводимость результатов может иметь серьезные практические последствия (например, в фармацевтической промышленности, когда какая-либо компания объявляет о повышенной эффективности своего нового дорогого препарата, по сравнению с эффективностью более дешевого старого аналога).

Одним из возможных решений указанной проблемы является корректировка полученных *p*-значений при помощи того или иного метода множественных сравнений (например, с использованием поправки Бонферрони). Однако, как отмечают Ф. Бретц и соавторы (Bretz et al., 2010), более высокую статистическую мощность обеспечил бы расчет *p*-значений, основанный на непосредственном учете корреляции между параметрами модели.

Этот подход как раз и реализован в пакете `multcomp`. В частности, в зависимости от класса рассматриваемой модели принимается допущение о том, что наблюдаемые значения ее параметров являются случайными реализациями значений из определенного многомерного распределения (*t*-распределения или нормального распределения), ковариационная матрица которого отражает корреляцию между параметрами модели. Размерность распределения равна числу одновременно проверяемых статистических гипотез в отношении параметров модели. Соответствующие *p*-значения рассчитываются, исходя из свойств этого распределения.

Главной функцией пакета multcomp является **glht**() (от «*general linear hypothesis testing*»), которая предоставляет удобный интерфейс для выполнения одновременной проверки статистических гипотез в отношении параметров самых разнообразных статистических моделей, включая общие линейные модели, обобщенные линейные модели, модели со смешанными эффектами и т. д. Основное требование – объект с результатами расчета той или иной модели должен содержать оценки ее параметров и ковариационной матрицы, которые могут быть извлечены из этого объекта при помощи методов **coef**() и **vcov**() соответственно. Так, для рассчитанной нами выше модели M имеем:

```
coef(M)
(Intercept) WaterAppletree    WaterBeaver   WaterGrayson      WaterRock
      44.08          -2.98          -3.85         -12.00          14.22
```

```
vcov(M)
             (Intercept) WaterAppletree WaterBeaver WaterGrayson WaterRock
(Intercept)         1.63          -1.63       -1.63        -1.63     -1.63
WaterAppletree     -1.63           3.26        1.63         1.63      1.63
WaterBeaver        -1.63           1.63        3.26         1.63      1.63
WaterGrayson       -1.63           1.63        1.63         3.26      1.63
WaterRock          -1.63           1.63        1.63         1.63      3.26
```

Синтаксис команд с использованием функции **glht**() в большинстве случаев имеет следующую структуру:

```
glht(model, linfct, alternative = c("two.sided", "less", "greater"), ...)
```

Здесь аргумент model – это линейная модель, подогнанная с использованием, например, таких стандартных функций, как **aov**(), **lm**() или **glm**(). На вход аргумента linfct (от «*linear function*» – линейная функция) подается матрица контрастов, при помощи которой задаются подлежащие проверке гипотезы (см. раздел 6.7). Имеющиеся способы спецификации аргумента linfct будут рассмотрены ниже. Аргумент alternative служит для указания типа альтернативных гипотез: "two.sided" (двухсторонняя), "less" («меньше, чем») и "greater" («больше, чем»). Многоточие «...» означает, что функция **glht**() может принимать и другие дополнительные аргументы (см. справочный файл, доступный по команде ?glht).

Способы спецификации линейных контрастов, используемых для кодирования сочетаний экспериментальных групп и подаваемых на аргумент linfct, рассмотрим на примере с содержанием стронция в водоемах. Следует подчеркнуть, что с подлежащими сравнению группами следует определиться *до выполнения анализа данных*. Иными словами, проверяемая научная гипотеза (или гипотезы) должна быть сформулирована до сбора данных. Действительно, любые гипотезы, формулируемые во время анализа уже имеющихся данных, будут диктоваться свойствами именно этих данных, и нет никакой гарантии, что при повторении эксперимента исследователь обнаружит те же закономерности.

В случае с дисперсионным анализом самый простой способ задать матрицу контрастов состоит в использовании служебной функции **mcp**() из пакета multcomp, которая сделает всю работу автоматически. От нас требуется лишь указать фактор, уровни которого соответствуют сравниваемым группам, и название одного из «встроенных» критериев для выполнения сравнений. Так, для выполнения всех парных сравнений средних значений концентрации стронция в исследованных водоемах при помощи критерия Тьюки соответствующая команда будет иметь вид:

```
glht(M, linfct = mcp(Water = "Tukey"))

  General Linear Hypotheses
Multiple Comparisons of Means: Tukey Contrasts
Linear Hypotheses:
                           Estimate
Appletree - Angler == 0   -2.9833
Beaver - Angler == 0      -3.8500
Grayson - Angler == 0    -12.0000
Rock - Angler == 0        14.2167
Beaver - Appletree == 0   -0.8667
Grayson - Appletree == 0  -9.0167
Rock - Appletree == 0     17.2000
Grayson - Beaver == 0     -8.1500
Rock - Beaver == 0        18.0667
Rock - Grayson == 0       26.2167
```

Результатом выполнения приведенной команды является таблица с перечисленными проверяемыми гипотезами (например, первая гипотеза состоит в том, что средние значения концентрации стронция в водоемах Appletree и Angler не различаются: Appletree - Angler == 0) и наблюдаемыми уровнями различий между группами (так, в водоеме Appletree содержание стронция в среднем оказалось на 2.983 мг/мл меньше, чем в Angler). Позднее мы увидим, как можно получить также *p*-значения для каждой из этих гипотез.

Помимо критерия Тьюки, реализованы и другие широко используемые критерии, такие как, например, критерий Даннетта ("Dunnett"), который применяется для сравнения всех групп с контрольной, или критерий Уильямса ("Williams"), который широко используется в токсикологии при анализе зависимости величины биологического эффекта от дозы исследуемого вещества. Критерий Уильямса подобен критерию Даннетта, но применяется для сравнения всех групп с контрольной только в случае, если групповые средние демонстрируют монотонный тренд на убывание или возрастание. С полным списком «встроенных» критериев можно ознакомиться в справочном файле, доступном по команде ?contrMat.

При необходимости выполнить сравнения только для определенных групп мы можем воспользоваться другим способом спецификации матрицы контрастов, который состоит в подаче на вспомогательную функцию **mcp**() списка из проверяемых гипотез, используя символьные выражения. Предположим, нас интересуют

сравнения только следующих водоемов друг с другом: «Rock vs. Angler», «Grayson vs. Appletree» и «Grayson vs. Beaver». Соответствующая команда будет иметь вид:

```
glht(M, linfct = mcp(Water = c(
  "Rock - Angler = 0",
  "Grayson - Appletree = 0",
  "Grayson - Beaver = 0")
  )
)

  General Linear Hypotheses
Multiple Comparisons of Means: User-defined Contrasts
Linear Hypotheses:
                          Estimate
Rock - Angler == 0          14.217
Grayson - Appletree == 0    -9.017
Grayson - Beaver == 0       -8.150
```

Наконец, пользователь может создать матрицу с весовыми коэффициентами линейных контрастов самостоятельно. Предположим, мы хотим выполнить те же три сравнения, что и в предыдущем примере. Создадим соответствующую матрицу с коэффициентами контрастов:

```
contr <- rbind("Rock - Angler" = c(-1, 0, 0, 0, 1),
               "Grayson - Appletree" = c(0, -1, 0, 1, 0),
               "Grayson - Beaver" = c(0, 0, -1, 1, 0)
               )

contr
                    [,1] [,2] [,3] [,4] [,5]
Rock - Angler         -1    0    0    0    1
Grayson - Appletree    0   -1    0    1    0
Grayson - Beaver       0    0   -1    1    0
```

Теперь подадим матрицу `contr` на функцию **`mcp`**():

```
glht(M, linfct = mcp(Water = contr))
  General Linear Hypotheses
Multiple Comparisons of Means: User-defined Contrasts
Linear Hypotheses:
                          Estimate
Rock - Angler == 0          14.217
Grayson - Appletree == 0    -9.017
Grayson - Beaver == 0       -8.150
```

Как видим, результат оказался идентичным предыдущему.

Метод **`summary`**() позволяет получить дополнительные результаты расчетов, выполняемых функцией **`glht`**(). В частности, мы можем вывести *p*-значения для соответствующих сравнений:

```
  summary(glht(M, linfct = mcp(Water = "Tukey")))
  Simultaneous Tests for General Linear Hypotheses
Multiple Comparisons of Means: Tukey Contrasts
Fit: lm(formula = Sr ~ Water, data = waterbodies)
Linear Hypotheses:
                         Estimate Std. Error t value Pr(>|t|)
Appletree - Angler == 0   -2.9833     1.8042  -1.654  0.47913
Beaver - Angler == 0      -3.8500     1.8042  -2.134  0.23757
Grayson - Angler == 0    -12.0000     1.8042  -6.651  < 1e-04 ***
Rock - Angler == 0        14.2167     1.8042   7.880  < 1e-04 ***
Beaver - Appletree == 0   -0.8667     1.8042  -0.480  0.98848
Grayson - Appletree == 0  -9.0167     1.8042  -4.998  0.00035 ***
Rock - Appletree == 0     17.2000     1.8042   9.533  < 1e-04 ***
Grayson - Beaver == 0     -8.1500     1.8042  -4.517  0.00112 **
Rock - Beaver == 0        18.0667     1.8042  10.014  < 1e-04 ***
Rock - Grayson == 0       26.2167     1.8042  14.531  < 1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```

Кроме получения подобной сводной информации, мы можем выполнить и некоторые другие виды анализа, воспользовавшись аргументом `test` функции **`summary`**() (подробнее см. справочный файл – `?summary.glht`).

Помимо расчета скорректированных *p*-значений, имеется также возможность рассчитать доверительные интервалы, которые будут учитывать корреляцию между параметрами модели. Для этого следует воспользоваться функцией **`confint`**(), подав на нее объект, полученный при помощи **`glht`**(), и указав требуемый доверительный уровень:

```
mult <- glht(M, linfct = mcp(Water = contr))
confint(mult, level = 0.95)
  Simultaneous Confidence Intervals
Multiple Comparisons of Means: User-defined Contrasts
Fit: lm(formula = Sr ~ Water, data = waterbodies)
Quantile = 2.5324
95% family-wise confidence level
Linear Hypotheses:
                         Estimate lwr      upr
Rock - Angler == 0        14.2167   9.6478  18.7855
Grayson - Appletree == 0  -9.0167 -13.5855  -4.4478
Grayson - Beaver == 0     -8.1500 -12.7188  -3.5812
```

В выведенных результатах столбцы `lwr` и `upr` содержат нижний и верхний доверительные пределы соответствующих оценок разницы средних значений (`Estimate`). Для удобства интерпретации рассчитанных доверительных интервалов, а также для презентации этих результатов в сжатом виде мы можем легко построить график (рис. 83), подобный приведенному на рис. 82:

```
plot(confint(mult, level = 0.95))
```

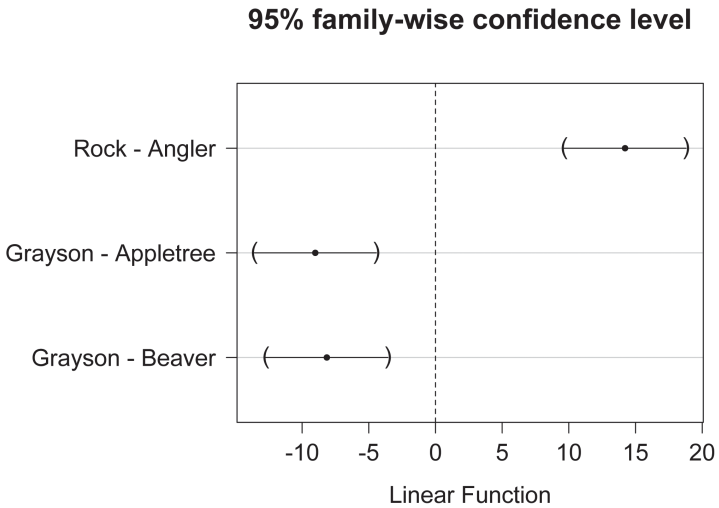

95% family-wise confidence level
Rock - Angler
Grayson - Appletree
Grayson - Beaver
-10
-5
0
5
10
15
20
Linear Function

Рисунок 83

Глава 7

Регрессионные модели зависимостей между количественными переменными

7.1. О понятии «статистическая модель»

Термин «модель» используется во многих сферах человеческой деятельности и имеет множество смысловых значений. Тем не менее в общем виде модель можно определить как некоторую искусственную конструкцию, являющуюся *упрощенным представлением* реальной системы. Более простое устройство модели выражается в том, что при ее создании принимают во внимание только определенные свойства реальной системы, полагаемые существенными для изучаемого процесса или явления (представьте себе, например, масштабную модель дома, созданную архитектором). Изменяя эти свойства у модели, мы можем лучше понять устройство реальной системы и, что особенно важно, с определенной вероятностью предсказать ее поведение в разных ситуациях.

Построение моделей является одной из центральных тем статистики как науки. Существует большое число классов статистических моделей, различающихся как по лежащим в их основе математическим принципам, так и по преимущественным областям применения. Однако, несмотря на все свое разнообразие, статистические модели сходны в том, что они описывают *взаимосвязь между случайными переменными*. Как именно это происходит? Постараемся разобраться.

Пример простейшей статистической модели

В мире реальных вещей и явлений практически всегда присутствует некоторый уровень *неопределенности,* вызванный естественной изменчивостью, влиянием внешних факторов или погрешностью измерений. Предположим, что мы имеем дело с некоторой *случайной* количественной переменной Y. Отдельные выборочные значения этой переменной будем обозначать как y_i. Индекс i изменяется от 1 до n, где n – это объем выборки.

В качестве примера рассмотрим систолическое кровяное давление у людей (выражается в мм ртутного столба). Очевидно, что уровень кровяного давления не может быть одинаковым у всех людей – при обследовании *случайно сформированной выборки* мы почти всегда будем наблюдать определенный *разброс значений* этой переменной, хотя некоторые значения будут встречаться чаще других. Ниже представлены 100 возможных значений давления крови (не будем пока уточнять механизм получения этих данных и предположим, что это – реальные измерения, выполненные у случайно отобранных людей, которые различались по возрасту, полу, массе тела и, возможно, каким-то другим характеристикам):

```
y <- c(
109.14, 117.55, 106.76, 115.26, 117.13, 125.39, 121.03,
114.03, 124.83, 113.92, 122.04, 109.41, 131.61, 103.93,
116.64, 117.06, 111.73, 120.41, 112.98, 101.20, 120.19,
128.53, 120.14, 108.70, 130.77, 110.16, 129.07, 123.46,
130.02, 130.31, 135.06, 129.17, 137.08, 107.62, 139.77,
121.47, 130.95, 138.15, 114.31, 134.58, 135.86, 138.49,
110.01, 127.80, 122.57, 136.99, 139.53, 127.34, 132.26,
120.85, 124.99, 133.36, 142.46, 123.58, 145.05, 127.83,
140.42, 149.64, 151.01, 135.69, 138.25, 127.24, 135.55,
142.76, 146.67, 146.33, 137.00, 145.00, 143.98, 143.81,
159.92, 160.97, 157.45, 145.68, 129.98, 137.45, 151.22,
136.10, 150.60, 148.79, 167.93, 160.85, 146.28, 145.97,
135.59, 156.62, 153.12, 165.96, 160.94, 168.87, 167.64,
154.64, 152.46, 149.03, 159.56, 149.31, 153.56, 170.87,
163.52, 150.97)
c(mean(y), sd(y)) # среднее значение и станд. отклонение
[1] 135.15730  16.96017
shapiro.test(y) # тест на нормальность распределения
        Shapiro-Wilk normality test
data:  y
W = 0.9826, p-value = 0.2121
library(ggplot2) # графическое изображение распределения данных
ggplot(data = data.frame(y), aes(x = y)) + geom_histogram() +
          ylab("Частота") + xlab("Давление, мм рт. ст.")
```

Приведенная на рис. 84 гистограмма по форме напоминает колокол, указывая на то, что мы, возможно, имеем дело с нормально распределенной переменной. Об этом же свидетельствуют и формальные тесты на нормальность, в частности тест Шапиро-Уилка.

Как известно, произвольное нормальное распределение полностью задается двумя *параметрами:* μ – математическим ожиданием (а именно средним значением) и σ –стандартным отклонением. То обстоятельство, что случайная величина Y распределена согласно нормальному закону, принято обозначать выражением $y_i \sim N(\mu, \sigma)$, которое представляет собой пример простейшей статистической модели.

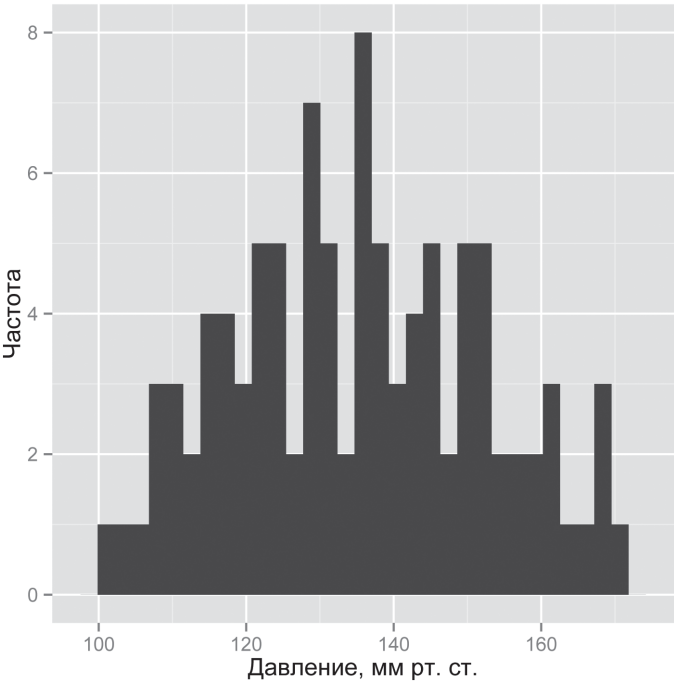


Рисунок 84

Подобно другим *параметрическим моделям*, эта модель включает две составные части: *систематическую и случайную*. Систематическая часть представлена средним значением μ, которое отражает *центральную тенденцию* в данных. В свою очередь, стандартное отклонение σ характеризует степень *вариации* значений *Y*, то есть их разброс относительно среднего.

Греческие буквы μ и σ обозначают истинные, или генеральные, параметры модели, которые, как правило, нам неизвестны. Тем не менее мы можем *оценить* значения параметров по соответствующим выборочным статистикам. Так, в случае представленных выше 100 значений систолического кровяного давления выборочные среднее значение и стандартное отклонение составляют 135.16 мм рт. ст. и 16.96 мм рт. ст. соответственно. Допуская, что данные действительно происходят из нормально распределенной *генеральной совокупности*, мы можем записать нашу модель в виде $y_i \sim N(135.16, 16.96)$. Эту модель можно использовать для предсказания давления крови, однако для всех людей предсказанное значение окажется одинаковым и будет равно μ. Обычным (и эквивалентным) способом записи такой модели является также следующий:

$$y_i = 135.16 + \varepsilon_i,$$

где ε_i – это *остатки* модели, имеющие нормальное распределение со средним значением 0 и стандартным отклонением 16.96, то есть $\varepsilon_i \sim N(0, 16.96)$. Остатки

рассчитываются как разница между реально наблюдаемыми значениями переменной Y и значениями, предсказанными моделью (в рассматриваемом примере $\varepsilon_i = y_i - 135.16$).

С другой стороны, эта запись представляет собой не что иное, как *линейную регрессионную модель*, у которой нет ни одного предиктора и которую часто называют «нуль-моделью» или «нулевой моделью» («*null model*»).

Исследование свойств статистических моделей имитационными методами

В сущности, статистическая модель – *это упрощенное математическое представление процесса, который, как мы полагаем, привел к генерации наблюдаемых значений изучаемой переменной*. Это значит, что мы можем использовать модель для *имитации* («*simulation*») – то есть процедуры, воспроизводящей моделируемый процесс и позволяющей тем самым искусственно генерировать новые значения изучаемой переменной, которые, как мы надеемся, будут обладать свойствами реальных данных. Причин, по которым мы хотели бы получить такие «искусственные данные», можно назвать много, но основные из них обычно сводятся к решению следующих практических задач:

- нахождение границ доверительных интервалов для параметров модели;
- оценка уровня неопределенности в отношении предсказываемых моделью значений зависимой переменной;
- расчет статистической мощности (сопутствующая задача – нахождение минимального объема наблюдений, необходимого для ответа на стоящий перед исследователем вопрос);
- сравнение искусственно сгенерированных значений зависимой переменной с реально наблюдаемыми значениями с целью выяснить, насколько хорошо модель подходит для описания изучаемого явления.

С подробными примерами, посвященными использованию *имитационных методов* при построении обычных регрессионных и многоуровневых иерархических моделей, можно ознакомиться в великолепной книге Э. Гелмана и Дж. Хилл (Gelman & Hill, 2007). Там же приведены многочисленные примера кода R для решения различных задач регрессионного анализа. Здесь мы обратимся к последней из перечисленных выше задач и рассмотрим, насколько хорошо значения кровяного давления, генерируемые нашей нулевой моделью, согласуются с экспериментальными данными.

Новые данные на основе этой простой модели можно легко сгенерировать в R при помощи функции **rnorm**() (подробнее см. раздел 4.6):

```
set.seed(101) # для воспроизводимости результата
y.new.1 <- rnorm(n = 100, mean = 135.16, sd = 16.96)
```

Выше было показано, что эквивалентным способом получения этих же значений будет генерация нормально распределенных остатков модели ε_i:

```
set.seed(101)
y.new.2 <- 135.16 + rnorm(n = 100, mean = 0, sd = 16.96)
```

```
# проверим, идентичны ли оба вектора
all(y.new.1 == y.new.2)
[1] TRUE
```

Теперь необходимо вспомнить о том, что параметры нашей нулевой модели являются лишь *точечными* оценками истинных параметров и что всегда будет присутствовать некоторая *неопределенность* в отношении того, *насколько точны* эти точечные оценки. В приведенных выше командах эта неопределенность не была учтена: при создании векторов `y.new.1` и `y.new.2` выборочные оценки среднего значения и стандартного отклонения кровяного давления рассматривались как параметры генеральной совокупности. В зависимости от поставленной задачи такой подход может оказаться достаточным. Однако мы сделаем еще один шаг и постараемся учесть неопределенность в отношении точечных оценок параметров модели.

При проведении имитаций воспользуемся функцией **`lm`**`()`, которая, как мы уже хорошо знаем, предназначена для подгонки линейных регрессионных моделей. Ничего удивительного здесь нет – ведь мы уже выяснили, что нашу простую модель кровяного давления можно рассматривать как линейную регрессионную модель, у которой нет ни одного предиктора:

```
y.lm <- lm(y ~ 1) # формула для оценки только свободного члена

summary(y.lm)
Call:
lm(formula = y ~ 1)
Residuals:
   Min     1Q Median     3Q    Max
-33.96 -13.26   0.41  12.04  35.71
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  135.157      1.696   79.69   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 16.96 on 99 degrees of freedom
```

Как следует из приведенных результатов, свободный член подогнанной модели (`Intercept`) в точности совпадает со средним значением данных (135.16 мм рт. ст.), а стандартное отклонение остатков модели (`Residual standard error`) совпадает со стандартным отклонением этих данных (16.96 мм рт. ст.). Важно, однако, что при этом мы параллельно вычислили оценку стандартной ошибки среднего значения, равную 1.696 (см. столбец `Std. Error` на пересечении со строкой `(Intercept)`).

По определению, стандартная ошибка некоторого параметра – это стандартное отклонение (нормального) распределения значений этого параметра, рассчитанных по выборкам одинакового размера из той же генеральной совокупности. Мы можем использовать это обстоятельство для учета неопределенности в отношении точечных оценок параметров модели при порождении новых данных в ходе процедуры имитации. Так, зная выборочные оценки параметров и их стандартные ошибки, мы можем: *а*) сгенерировать несколько возможных значений этих

параметров (то есть составить несколько реализаций той же модели, варьируя значения параметров) и *б*) сгенерировать новые данные на основе каждой из этих альтернативных реализаций модели.

Решение указанных задач возможно с использованием функции **sim**() из пакета arm, который является приложением к упомянутой выше книге Gelman & Hill (2007). Эта функция принимает на входе модельные объекты классов lm, glm и др., извлекает оценки параметров и их соответствующие стандартные ошибки и возвращает заданное пользователем число альтернативных реализаций модельных параметров. Например, следующие команды позволят сгенерировать 5 альтернативных реализаций среднего кровяного давления и его стандартного отклонения для нашей простой модели:

```
install.packages("arm")
library(arm)
set.seed(102) # для воспроизводимости результата
y.sim <- sim(y.lm, 5)

# y.sim - объект класса S4, который содержит
# слоты coef (коэффициенты модели) и sigma
# (станд. отклонения остатков модели):
str(y.sim)
Formal class 'sim' [package "arm"] with 2 slots
  ..@ coef : num [1:5, 1] 136 134 137 136 137
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : NULL
  ..@ sigma: num [1:5] 16.8 18.9 17.3 16.7 15

# Извлекаем альтернативные реализации среднего из y.sim:
y.sim@coef
         [,1]
[1,] 136.4775
[2,] 134.3283
[3,] 136.7074
[4,] 136.0771
[5,] 137.3246

# Извлекаем альтернативные реализации ст.отклонений остатков:
y.sim@sigma
[1] 16.829, 18.870, 17.303, 16.743, 15.006
```

Конечно, пяти реализаций модели недостаточно, чтобы сделать какие-либо убедительные выводы. Увеличим это число до 1000:

```
set.seed(102)
y.sim <- sim(y.lm, 1000)

# Инициализация пустой матрицы, в которой мы будем сохранять
# данные, сгенерированные на основе 1000 альтернативных реализаций модели:
```

```
y.rep <- array(NA, c(1000, 100))

# Заполняем матрицу y.rep имитированными данными:
for(s in 1:1000){
  y.rep[s, ] <- rnorm(100, y.sim@coef[s], y.sim@sigma[s])
      }
```

Чтобы лучше понять, что мы только что сделали, изобразим гистограммы выборочных распределений значений кровяного давления, сгенерированных на основе, например, первых 12 реализаций нулевой модели (рис. 85):

```
par(mfrow = c(3, 4), mar = c(2, 2, 1, 1))
for(s in 1:12){ hist(y.rep[s, ], xlab = "", ylab = "", breaks = 20, main = "")}
```

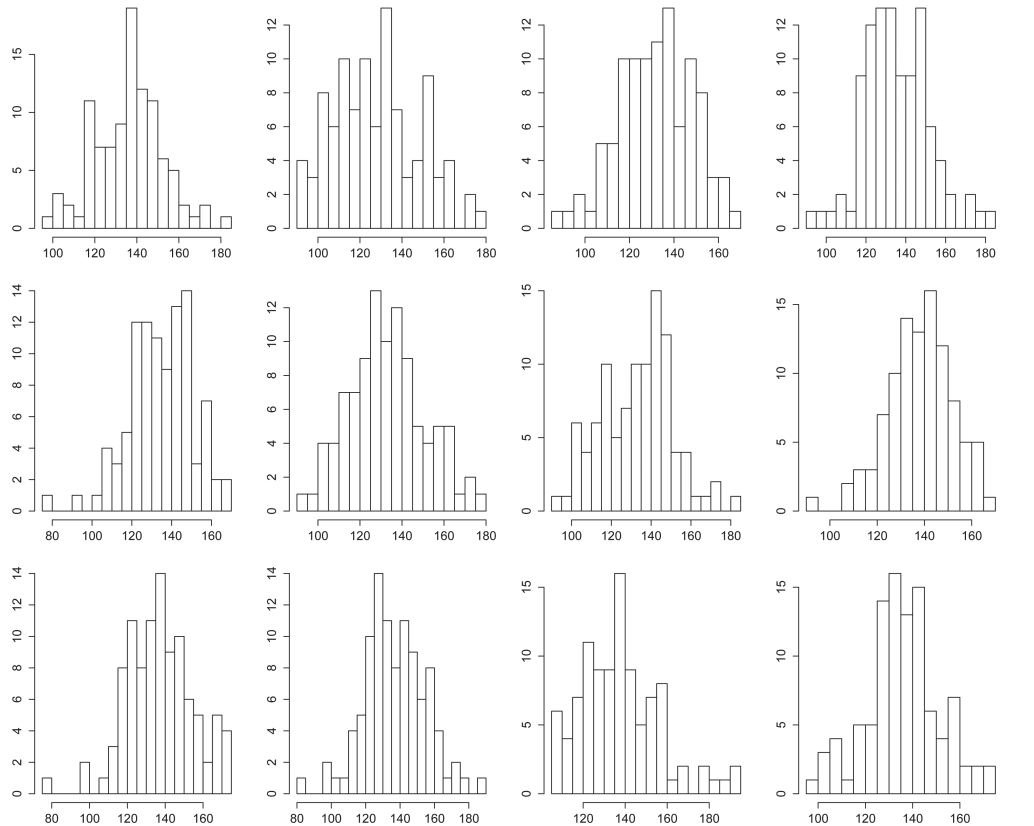


Рисунок 85

Приведенный рисунок дает некоторое представление о том, насколько варьируют между собой и по отношению к реально наблюдаемым данным распределения

значений кровяного давления, полученные имитацией на основе альтернативных реализаций нашей нуль-модели.

Однако больше всего нас интересует наличие *систематических расхождений* между свойствами этих имитированных значений и свойствами реально наблюдаемых данных. Если такие расхождения имеются, значит, нулевая модель недостаточно хороша для описания вариации реально наблюдаемых значений. Имеется несколько подходов для обнаружения этих расхождений. Один из них – расчет каких-либо подходящих описательных статистик для каждого из имитированных распределений и сравнение их с теми же статистиками у реальных данных.

Рассчитаем *интерквартильный размах* (ИКР) для каждого имитированного набора данных и сравним полученное распределение из 1000 таких значений с ИКР реальных данных. Для расчета ИКР в R служит базовая функция **IQR**():

```
# Расчет ИКР для каждого из 1000 имитированных
# распределений значений кровяного давления
test.IQR <- apply(y.rep, MARGIN = 1, FUN = IQR)
```

Выведем гистограмму 1000 значений ИКР, рассчитанных для каждого из имитированных распределений кровяного давления (рис. 86). Вертикальной синей линией покажем ИКР для реально наблюдаемых значений кровяного давления:

```
hist(test.IQR, xlim = range(IQR(y), test.IQR),
     main = "ИКР", xlab = "", ylab = "Частота", breaks = 20)
lines(rep(IQR(y), 2), c(0, 100), col = "blue", lwd = 4)
```

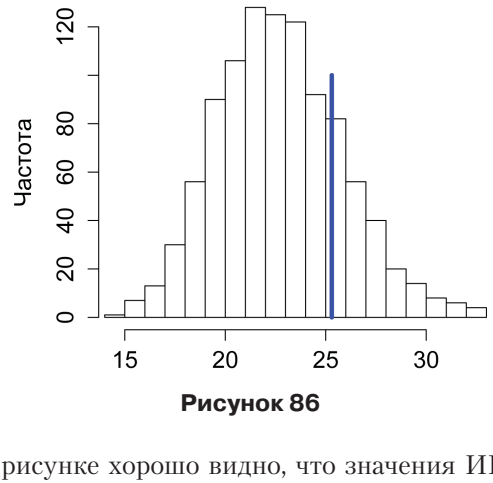


Рисунок 86

На приведенном рисунке хорошо видно, что значения ИКР для имитированных данных систематически занижены, по сравнению с реальными данными. Это свидетельствует о том, что нулевая модель в целом недооценивает уровень вариации реальных значений кровяного давления. Причиной этому может быть то, что

мы не учитываем воздействие на кровяное давление каких-то важных факторов (например, возраст, пол, диета, состояние здоровья и т. п.). Рассмотрим, как можно расширить нашу нулевую модель, добавив в нее один из таких факторов.

Пример модели с одним количественным предиктором

Предположим, что помимо кровяного давления мы также измеряли у каждого испытуемого его/ее возраст (в годах):

```
# Значения возраста:
x <- rep(seq(16, 65, 1), each = 2)
# Объединяем значения возраста и давления крови в одну таблицу
Data <- data.frame(Age = x, BP = y)
```

Покажем графически связь между возрастом и систолическим кровяным давлением (рис. 87). Для визуализации тренда в данных добавим линию регрессии синего цвета:

```
ggplot(data = Data, aes(x = Age, BP)) + geom_point() +
       geom_smooth(method = "lm", se = FALSE) +
       geom_rug(color = "gray70", sides = "tr") +
       ylab("Частота") + xlab("Возраст, лет")
```

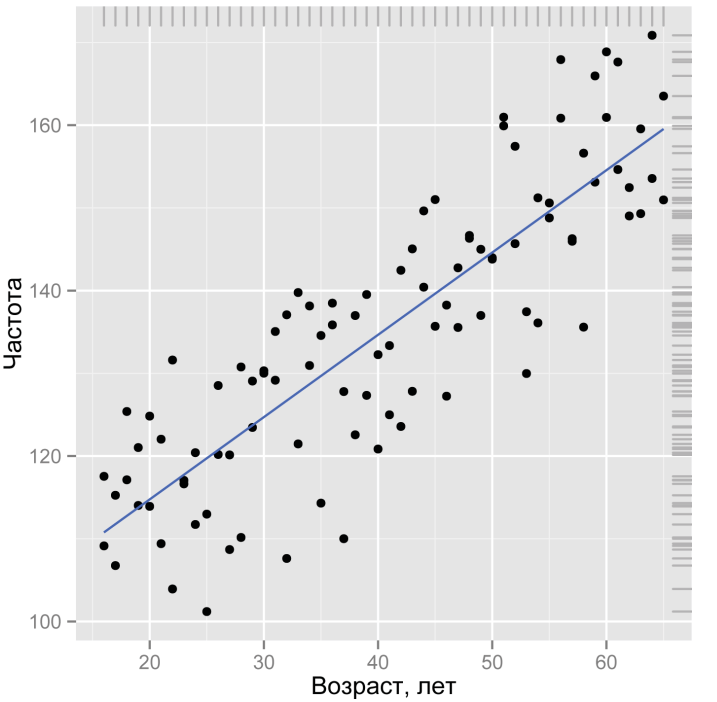


Рисунок 87

Небольшие отрезки светло-серого цвета по краям графика соответствуют наблюдаемым значениям возраста (вверху) и кровяного давления (справа).

Из графика видно, что между давлением крови и возрастом существует выраженная линейная зависимость: несмотря на определенную вариацию наблюдений, по мере увеличения возраста давление *в среднем* также возрастает. Мы можем учесть это систематическое изменение среднего кровяного давления, добавив возраст (`Age`) в нашу нулевую модель:

$$y_i \sim N(\beta_0 + \beta_1 \times Age_i, \sigma),$$

или в эквивалентной форме записи:

$$y_i = \beta_0 + \beta_1 \times Age_i + \varepsilon_i, \text{ где } \varepsilon_i \sim N(0, \sigma).$$

Параметры β_0 и β_1 можно без труда оценить методом наименьших квадратов при помощи функции **`lm`**():

```
summary(lm(BP ~ Age, data = Data))
Call:
lm(formula = BP ~ Age, data = Data)
Residuals:
     Min       1Q   Median       3Q      Max
-21.6626  -6.2509   0.0075   6.3125  17.3525
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 94.85346    2.66731   35.56   <2e-16 ***
Age          0.99515    0.06204   16.04   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 8.953 on 98 degrees of freedom
Multiple R-squared:  0.7242, Adjusted R-squared:  0.7214
F-statistic: 257.3 on 1 and 98 DF,  p-value: < 2.2e-16
```

Согласно полученным результатам, модель кровяного давления можно записать как

$$y_i \sim N(94.853 + 0.995 \times Age_i, 8.953)$$

или

$$y_i = 94.853 + 0.995 \times Age_i + \varepsilon_i,$$

где $\varepsilon_i \sim N(0, 8.953)$.

Графически эта модель изображена на рис. 87 в виде голубой линии. Обратите внимание: помимо высокой значимости параметров подогнанной модели ($p << 0.001$ в обоих случаях), стандартное отклонение остатков составляет 8.853, что почти в 2 раза меньше, чем у нулевой модели (16.96). Это указывает на то, что модель, включающая возраст в качестве предиктора, гораздо лучше описывает вариацию значений кровяного давления у 100 обследованных испытуемых, чем наша исходная модель.

Данное заключение подтверждается тем, что в результате выполнения имитаций, аналогичных описанным выше, значение интерквартильного размаха для исходных данных располагается в центре распределения имитированных значений ИКР, указывая на отсутствие систематических различий между имитированными и наблюдаемыми данными. Код на языке R и полученную гистограмму мы здесь не приводим, рекомендуя читателю проделать вычисления самостоятельно.

Теперь настало время раскрыть небольшой секрет: тот факт, что модель, включающая возраст, гораздо лучше описывает исходные данные, неудивителен, поскольку эти наблюдения были... сгенерированы на основе модели

$$y_i = 97.078 + 0.949Age_i + \varepsilon_i, \text{ где } \varepsilon_i \sim N(0, 9.563)$$

следующим образом:

```
set.seed(101)
y <- rnorm(100, mean = 97.078 + 0.949*x, 9.563)
```

Последняя модель, *придуманная в целях демонстрации* обсуждаемых принципов, была использована в качестве «истинной» («*true model*») в том смысле, что она описывала генеральную совокупность. Другими словами, мы предположили, что у нас была возможность одномоментно измерить давление у всех существующих людей и полученные данные описывались бы именно этой «истинной» моделью.

В реальной ситуации ни структура (систематическая часть и остатки), ни значения параметров истинной модели исследователю, как правило, неизвестны. Все, чем он располагает, – это наборы экспериментальных данных, часто недостаточно репрезентативных и сильно «зашумленных». Имея эти данные и хорошее понимание изучаемого явления (в смысле того, какие факторы считать достаточно важными для рассмотрения), исследователь может только надеяться приблизиться к структуре истинной модели и оценить ее параметры с определенной точностью. К сожалению, такой успех гарантирован далеко не всегда.

Назначение регрессионных моделей

В общем виде под *регрессией* случайной величины Y на X понимается зависимость, задающая траекторию движения точки (y_x, x), которая определяется условным математическим ожиданием $y_x = \mathrm{E}[Y|X = x]$ для каждого текущего значения x. Эта траектория моделируется с использованием некоторой *функции регрессии* $f(x, \beta)$ с параметрами β, которая позволяет оценить средние значения $\hat{y}$ реализаций зависимой переменной Y для каждого фиксированного значения x независимой переменной (предиктора) X. Поскольку для точного описания $f(x, \beta)$ необходимо знать закон условного распределения $\mathrm{E}[Y|X = x]$, то на практике ограничиваются ее наиболее подходящей аппроксимацией $\hat{f}_a(x, \beta)$, обеспечивающей минимум средней ошибки ε восстановления значений $Y(x)$. Эти рассуждения легко распространить на многомерный случай, где X – набор из нескольких предикторов. В сущности, статистическое моделирование, или «*статистическое обучение*» («*statistical learning*»; James et al., 2013), представляет собой набор подходов, позволяющих оценить функцию $f(x, \beta)$ на основе ограниченной совокупности наблюдений.

Существует несколько основных причин для нахождения функции регрессии:

1. *Статистические выводы и объяснение научных феноменов*: оценка функции $f(x, \beta)$ облегчает *понимание* характера взаимоотношений между переменной-откликом и предикторами (например, насколько силен отклик при изменении того или иного предиктора, какой из предикторов наиболее важен и т. п.). Так, в примере выше мы выяснили, что возраст тесно связан с уровнем систолического давления – информация, которая может оказаться очень важной для практикующего врача.
2. *Компактное описание*, позволяющее представить наиболее объяснимые с предметной точки зрения функциональные отношения между переменными в рамках формальной модели и найти оценку ее параметров, основанную на эмпирических данных.
3. *Обобщение* результатов многих выборочных наблюдений и распространение статистических выводов на всю генеральную совокупность (в форме, например, доверительных интервалов модели).
4. *Прогноз*, или *предсказание*, заключающееся в вычислении ожидаемых значений $\hat{y}$ переменной-отклика с использованием уравнения модели. Например, банк хочет спрогнозировать риск невозврата кредита в зависимости от набора предикторов (уровень доходов и образования, опыт работы, пол, возраст, кредитная история клиента). Точность предсказания будет зависеть от двух составляющих – *устранимой* и *неустранимой погрешности* («*reducible error*» и «*irreducible error*» соответственно). Первая связана с тем, что наша модель никогда не является идеальной, хотя при использовании дополнительной информации об изучаемом явлении и применении более точных статистических методов имеющуюся неопределенность можно снизить. В свою очередь, неустранимая погрешность обусловлена неточностью измерений и влиянием скрытых факторов, которые мы не можем учесть, потому что просто о них не знаем.

7.2. Простая линейная регрессия: каков возраст Вселенной?

Как было показано выше, регрессионная модель представляет собой упрощенное математическое представление изучаемого процесса, с помощью которого, по нашему предположению, можно будет с минимальной ошибкой предсказывать вероятные значения наблюдаемой величины. Во многих случаях первым естественным приближением функции регрессии некоторого отклика y на предиктор x является *модель простой линейной регрессии*, которая может быть записаная следующим образом:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i,$$

где β_0 и β_1 – генеральные параметры, или коэффициенты, модели; ε_i – остатки, или ошибки, в отношении которых делаются следующие предположения (подробнее см. раздел 7.3):

- остатки статистически независимы (то есть в данных нет пространственно-временной корреляции);
- остатки нормально распределены с нулевым средним, то есть их математическое ожидание $E[\varepsilon_i] = 0$;
- дисперсия остатков ε_i одинакова на всем диапазоне наблюдаемых значений предиктора x, то есть $E[\varepsilon_i^2] = \sigma^2$ (см. аналогичные исходные предпосылки для линейной модели дисперсионного анализа ANOVA, представленные в разделах 6.2 и 6.4).

Общий подход для нахождения оптимальных оценок $\hat{\beta}_0$ и $\hat{\beta}_1$ неизвестных параметров модели β_0 и β_1 заключается в минимизации некоторой *функции потерь* $\rho(\hat{\varepsilon})$, учитывающей разности между прогнозируемыми и фактическими значениями отклика:

$$\sum_{i=1}^{n} \rho\left[y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i)\right] = \sum_{i=1}^{n} \rho\left[(y_i - \hat{y}_i)\right] = \sum_{i=1}^{n} \rho(\varepsilon) \to \min.$$

Если указанные выше требования в отношении остатков выполняются, то так называемый *метод наименьших квадратов* (МНК; англ. «*ordinary least squares*», OLS) позволит получить оптимальные (в смысле их несмещенности и эффективности) оценки параметров модели. МНК, являющийся частным случаем *метода максимального правдоподобия*, основан на использовании функции потерь $\rho(\hat{\varepsilon}) = \hat{\varepsilon}^2$, то есть минимизируется *сумма квадратов остатков* («*residual sum of squares*»):

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y})^2.$$

Можно показать, что минимальное значение *RSS* будет достигаться при следующих значениях коэффициентов модели:

$$\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2},$$

что соответствует тангенсу угла наклона линии регрессии в координатах (x; y), и

$$\hat{\beta}_0 = \bar{y} - \beta_1 \bar{x},$$

что соответствует длине отрезка, отсекаемого этой линией на оси ординат.

Наконец, *стандартное отклонение остатков* («*residual standard error*»), являющееся важной характеристикой качества модели, в случае простой линейной регрессии оценивается как $\sigma = \sqrt{RSS / (n-2)}$.

Модель для оценки постоянной Хаббла

Как известно, связь между расстоянием до внегалактического объекта (например, галактики, квазара) и скоростью его удаления, обусловленного расширением Вселенной после Большого взрыва, описывается законом Хаббла (http://bit.ly/1GvvHHe). Этот закон выражается простой линейной зависимостью: $y = \beta x$,

где y – относительная скорость движения любых двух галактик, разграниченных в данный момент времени расстоянием x, а β – постоянная Хаббла, выраженная в км/с на мегапарсек. Величина, обратная постоянной Хаббла, дает приблизительный возраст Вселенной (так называемый «хаббловский возраст»).

В рамках проекта *Key Project*, целью которого являлось уточнение значения β, с помощью космического телескопа «Хаббл» (http://bit.ly/1OBkybv) были измерены расстояния до 24 галактик и установлены скорости их удаления. Пользуясь этими опубликованными данными (Freedman et al., 2001, http://bit.ly/1EYQZv2), мы можем оценить значение постоянной Хаббла при помощи простой линейной регрессии. Обратим внимание на то, что свободный член уравнения здесь приравнен к нулю, поскольку в момент, когда Вселенная находилась в состоянии сингулярности, галактик не существовало и они, конечно же, не могли удаляться друг от друга.

Готовые для построения модели данные из статьи Friedman et al. (2001) можно найти в составе таблицы `hubble` из пакета `gamair`:

```
install.packages("gamair")
library(gamair)
data(hubble)
str(hubble)
'data.frame': 24 obs. of  3 variables:
 $ Galaxy: Factor w/ 24 levels "IC4182","NGC0300",..: 2 3 4 5 6 8 9 7 10 11
 $ y     : int  133 664 1794 1594 1473 278 714 882 80 772 ...
 $ x     : num  2 9.16 16.14 17.95 21.88 ...
```

Как было показано ранее, параметры линейной модели в R можно оценить при помощи стандартной функции **`lm`**`()`:

```
M <- lm(y ~ x - 1, data = hubble)
```

Обратите внимание на `-1` в формуле модели – такое обозначение используется для того, чтобы исключить свободный член регрессионной модели. Посмотрим на результат подгонки модели:

```
summary(M)
Call:
lm(formula = y ~ x - 1, data = hubble)
Residuals:
   Min     1Q Median     3Q    Max
-736.5 -132.5  -19.0  172.2  558.0
Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x   76.581      3.965   19.32 1.03e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 258.9 on 23 degrees of freedom
Multiple R-squared:  0.9419, Adjusted R-squared:  0.9394
F-statistic: 373.1 on 1 and 23 DF,  p-value: 1.032e-15
```

Как видим, оцененное значение постоянной Хаббла составило 76.581 км/с на мегапарсек. Это значение существенно отличается от нуля (см. *p*-значение соответствующего *t*-теста в столбце `Pr(>|t|)`).

С использованием полученного нами уравнения $y = 76.581x$ расчет возраста Вселенной не представляет труда. Один мегапарсек – это 3.09×1019 км. Разделим полученную выше постоянную Хаббла на это значение, чтобы выразить ее в секундах:

```
(hub.const <- 76.581/3.09e19)
 [1] 2.47835e-18
```

Тогда возраст Вселенной, выраженный в секундах, составит:

```
(age <- 1/hub.const)
 [1] 4.034943e+17
```

Выполнив простое преобразование, получим возраст, выраженный в годах:

```
age/(60^2*24*365)
[1] 12794721567
```

Таким образом, данные Freedman et al. (2001) указывают на то, что возраст Вселенной составляет ~12.8 миллиарда лет. Отметим, что по последним оценкам возраст Вселенной составляет 13.798 млрд лет (http://bit.ly/1EYRoO2).

Доверительные интервалы

Обычно истинные величины коэффициентов регрессии β_0 и β_1 неизвестны, и мы находим лишь их оценки $\hat{\beta}_0$ и $\hat{\beta}_1$ (например, при помощи функции `lm()`). Иначе говоря, истинная линия регрессии может проходить как-то иначе, чем построенная по выборочным данным. Оценить неопределенность в отношении построенной модели и/или получаемых с ее помощью предсказаний можно, вычислив соответствующую доверительную область. В общем случае различают три способа выражения интервальных оценок точности и воспроизводимости модели: *а*) доверительная область регрессионной линии, *б*) доверительная область значений прогнозируемой переменной и *в*) толерантные интервалы регрессии (Кобзарь, 2006, с. 665).

Как было отмечено выше, предполагается, что при любом текущем значении $x = x_0$ соответствующие ему значения y распределены нормально, а их средним является значение $\hat{y}$, полученное по уравнению регрессии. Неопределенность оценки $\hat{y}$ в точке x_0 (обозначим эту оценку как $\hat{\mu}_{Y|x_0}$) характеризуется *стандартной ошибкой регрессии*:

$$S_{\hat{y}} = \sigma\sqrt{\frac{1}{n} + \frac{(\bar{x} - x_0)^2}{\sum(x_i - \bar{x})^2}},$$

где σ – рассмотренное ранее стандартное отклонение остатков.

Зная $S_{\hat{y}}$, 100(1 – α)-процентный *доверительный интервал модели регрессии* (*CI*, от «*confidence interval*») в точке x_0 можно рассчитать как $CI = \hat{\mu}_{Y|x_0} \pm t_{\alpha/2,\, n-2} S_{\hat{y}}$, где

$t_{(1-\alpha/2,\, n-2)}$ – это соответствующий квантиль t-распределения Стьюдента. Доверительная область, рассчитанная по приведенной формуле, является геометрическим местом доверительных интервалов на всем наблюдаемом диапазоне значений предиктора x. Обычно это довольно узкая полоса, которая несколько расширяется при крайних значениях x (что неудивительно, поскольку при возрастании x_0 возрастает и $S_{\hat{y}}$ – см. формулу выше).

Доверительный интервал значений зависимой переменной, или интервал предсказания (*PI*, от «*prediction interval*»), складывается из разброса значений вокруг линии регресии и неопределенности положения самой этой линии. Его ширина определяется ошибкой $S_e(\hat{y}|x_0) = S_e(\hat{y}|x_0) = S_e(\hat{y}_{x_0}) + \varepsilon_{\hat{y}|x_0}$, где $\varepsilon_{\hat{y}|x_0}$ – случайная условная ошибка при прогнозе $\hat{y}$. Для простой линейной регрессии эту ошибку можно рассчитать по формуле

$$S_Y = S_e\sqrt{1+\frac{1}{n}+\frac{(\bar{x}-x_0)^2}{\sum(x_i-\bar{x})^2}}.$$

Доверительная область $PI = \hat{\mu}_{Y|x_0} \pm t_{\alpha/2,\, n-2}S_Y$, рассчитанная для всех возможных значений предиктора x_0, является геометрическим местом доверительных интервалов для величин $\hat{y}$, экстраполированных на «новые» прогнозируемые значения отклика.

В статистической среде R интервалы *CI* и *PI* для линейной модели легко оценить с использованием функции **predict**(). Эта функция предназначена для расчета значений $\hat{y}$ по подогнанной модели и имеет два важных параметра: newdata, который определяет таблицу с набором предикторов (если он не определен, то используется исходная выборка), и interval = **c**("none", "confidence", "prediction"), обозначающий тип оцениваемой доверительной области. Для модели, оценивающей постоянную Хаббла, имеем:

```
CPI.df <- cbind(predict(M, interval = "conf"), predict(M, interval = "pred"))
CPI.df <- CPI.df[,-4];
colnames(CPI.df) <- c("Y_fit", "CI_l", "CI_u", "PI_l", "PI_u")
head(CPI.df)
      Y_fit      CI_l      CI_u      PI_l      PI_u
1  153.1623  136.7587  169.5659 -382.7326  689.0573
2  701.4835  626.3550  776.6120  160.5966 1242.3704
3 1236.0201 1103.6430 1368.3972  684.2611 1787.7791
4 1374.6320 1227.4097 1521.8544  819.1244 1930.1397
5 1675.5960 1496.1406 1855.0515 1110.6902 2240.5019
6  246.5914  220.1816  273.0012 -289.7031  782.8859
```

Построим график, изображающий линию регрессии и доверительные области (рис. 88):

```
matplot(hubble$x, CPI.df,
type = "l", lwd = c(2, 1, 1, 1, 1),
col = c(1, 2, 2, 4, 4),
ylab = "Скорость, км/с", xlab = "Расстояние, Мпс")
with(hubble, matpoints(x, y, pch = 20))
```

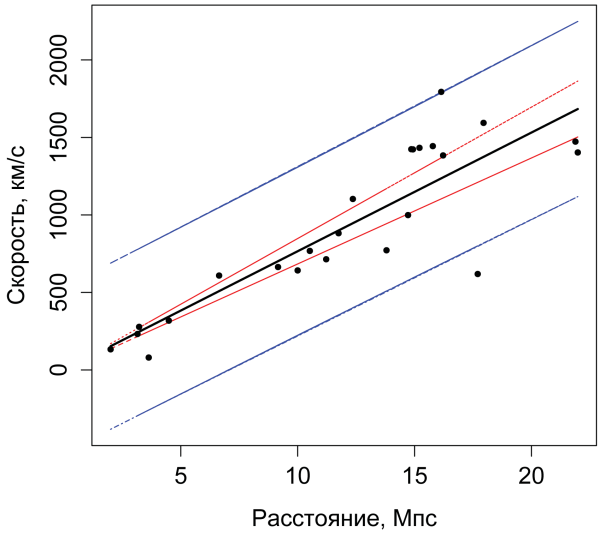


Рисунок 88

На рис. 88 черным цветом показана линия регрессии $y = 76.581x$, построенная по 24 точкам, а также 95%-ная доверительная область линии регрессии, которая ограничена пунктирными линиями красного цвета. Можно утверждать, что если мы соберем аналогичные наборы данных по другим галактикам и построим по ним линии регрессии, то в 95 случаях из 100 эти прямые не покинут пределов доверительной области, ограниченной красными линиями.

Обратите внимание, что некоторые точки оказались вне доверительной области, ограниченной красными линиями. Это совершенно естественно, поскольку речь идет о доверительной области *CI* линии регрессии, а не самих значений. В то же время в 95%-ную доверительную область прогнозирования *PI*, ограниченную синими линиями, попадет 95% всех возможных значений величины y в исследованном диапазоне значений x.

Оценка неопределенности в отношении параметров линейной регрессии

Как и в случае с любыми другими выборочными оценками, всегда существует неопределенность в отношении того, насколько выборочные оценки параметров регрессионной модели близки к соответствующим истинным значениям (то есть параметрам генеральной совокупности). Рассмотрим несколько способов описания этой неопределенности, которую количественно выражают при помощи стандартных ошибок и доверительных интервалов.

1. *Параметрический подход.* Если объем выборки «достаточно велик» и выполняются определенные условия в отношении свойств остатков модели, можно воспользоваться *центральной предельной теоремой* (ЦПТ; http://bit.ly/19pbzJy),

которая предсказывает, что выборочные оценки любого параметра линейной модели имеют приближенно нормальное распределение.

Как видно из приведенных ранее результатов расчетов, полученных с помощью функции **lm**(), постоянная Хаббла была оценена (Estimate) в 76.581 км/с на мегапарсек со стандартной ошибкой (Std. Error) 3.965 км/с на мегапарсек. По определению, стандартная ошибка параметра – это стандартное отклонение (нормального) распределения значений этого параметра, полученного по случайным выборкам фиксированного размера из той же генеральной совокупности. Соответственно, чем меньше значение стандартной ошибки параметра, тем его оценка точнее.

Согласно ЦПТ, мы предполагаем, что истинное значение постоянной Хаббла принадлежит нормально распределенной совокупности, математическое ожидание и стандартное отклонение которой составляют около 76.581 и 3.965 км/с на мегапарсек соответственно. Заметим, однако, что найденное нами наиболее вероятное точечное значение постоянной при повторных измерениях может оказаться иным. Чтобы охарактеризовать эту неопределенность, необходимо рассчитать *доверительный интервал* – не противоречащий имеющимся данным диапазон значений, в котором истинное значение постоянной Хаббла находится с определенной вероятностью (например, 95%).

Ориентировочно оценить границы доверительного интервала можно, предположив, что примерно 95% всех значений распределения оценок постоянной Хаббла лежат в диапазоне $\pm 2SE_\beta$ относительно его среднего значения, где SE_β – стандартная ошибка параметра β, то есть:

$$76.581 - 2\times3.965 = 68.651 \quad \text{и} \quad 76.581 + 2\times3.965 = 84.511 \text{ км/с на мегапарсек.}$$

Однако неопределенность имеется в отношении не только оценки самого параметра β, но и оценки его стандартного отклонения для соответствующего нормального распределения. Не углубляясь в детали, отметим, что в связи с этим обстоятельством более точные значения границ доверительного интервала дадут вычисления, основанные на свойствах *t*-распределения Стьюдента. Тогда границы 95%-го доверительного интервала для параметра β составят: $\beta \pm t_{0.975} SE_\beta$, где $t_{0.975}$ – 0.975-квантиль *t*-распределения с $(n - p)$ числом степеней свободы, где n – объем выборки, а p – число параметров модели. Для нашего примера получаем:

```
beta <- summary(M)$coefficients[1]
SE <- summary(M)$coefficients[2]
ci.lower <- beta - qt(0.975, df = 23)*SE
ci.upper <- beta + qt(0.975, df = 23)*SE

c(ci.lower, ci.upper)
[1] 68.379  84.783
```

Используя эти оценки нижней и верхней границ 95%-го доверительного интервала, мы можем рассчитать диапазон значений, в котором с вероятностью 95% находится истинный возраст Вселенной:

```
Uni.upper <- 1/(ci.lower*60^2*24*365.25/3.09e19)
Uni.lower <- 1/(ci.upper*60^2*24*365.25/3.09e19)

c(Uni.lower, Uni.upper)
[1] 11549039573   14319551383
# то есть примерно от 11.5 до 14.3 млрд лет
```

Заметьте, что полученный таким способом доверительный интервал включает значение возраста Вселенной, рассчитанное на основе самых последних космологических изысканий (13.798 млрд лет).

2. *Бутстреп.* В разделах 2.7 и 5.3 нами уже рассматривались методы «размножения выборок» (рандомизация и бутстреп) для получения эмпирических распределений анализируемых статистик. Используем бутстреп-метод для оценки стандартной ошибки и 95%-го доверительного интервала для постоянной Хаббла.

На рис. 89 приведены примеры 6 случайных выборок из 24 наблюдений, извлеченных из исходной совокупности данных по скорости удаления галактик. Эти псевдовыборки формируются «с возвратом» – это значит, что, будучи однажды случайно выбранным, то или иное наблюдение вновь помещается в исходную со-

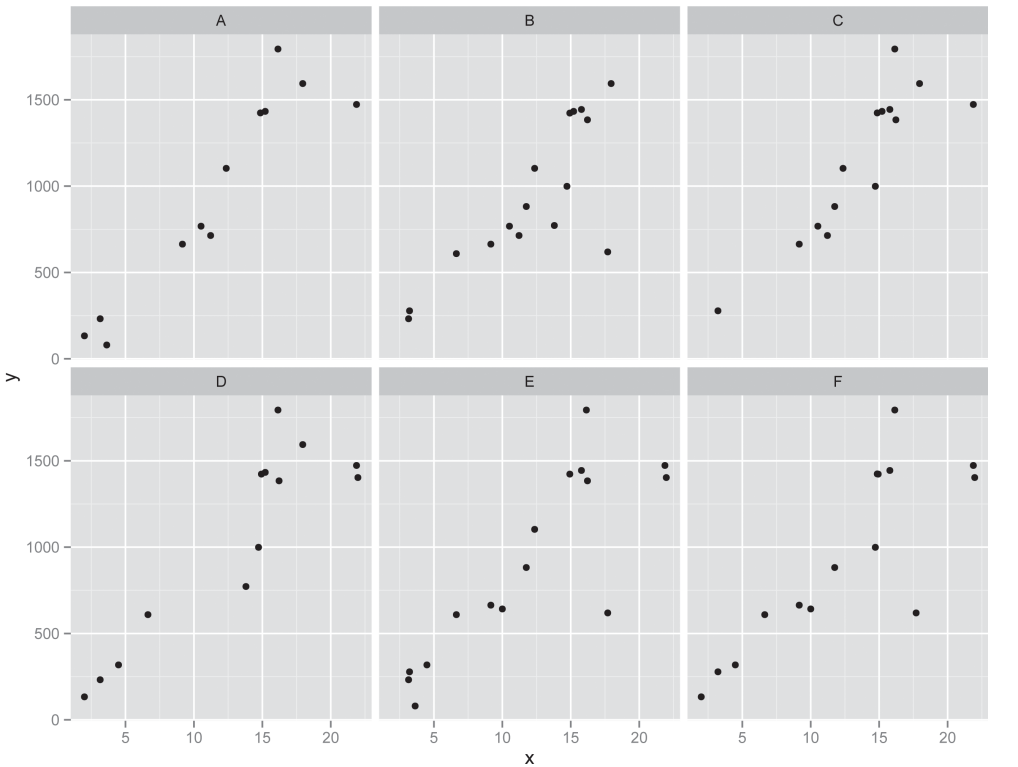


Рисунок 89

вокупность, откуда оно может быть извлечено снова (следовательно, некоторые наблюдения могут повторяться в генерируемой выборке несколько раз).

Обратите внимание: на представленном рисунке общий тренд сохраняется («чем больше *x*, тем больше *y*»), однако наблюдения, входящие в состав каждой выборки A–F, несколько различаются, что в итоге приведет и к несколько различающимся оценкам постоянной Хаббла при подгонке линейной модели при помощи функции **lm**().

Как отмечено выше, при выполнении бутстреп-процедуры формируется «большое» число повторных выборок из исходной совокупности. То, насколько «большим» должно быть это число, зависит от объема исходной совокупности и ее свойств (подробнее см. Efron & Tibshirani, 1994, http://bit.ly/1ByfOdE). Для нашего примера мы сформируем 1000 случайных повторных выборок. Хотя формирование бутстреп-распределений параметров регрессии можно реализовать в R, «вручную» записав необходимые команды, проще будет воспользоваться специально предназначенной для этого функцией **boot**() из одноименного стандартного пакета. Эта функция имеет следующие обязательные аргументы:

- data: таблица с исходными данными;
- statistic: функция, выполняющая вычисление интересующего нас параметра(ов);
- R: число повторных выборок, по которым рассчитывается этот параметр.

Создадим небольшую функцию, которая будет оценивать параметры регрессионной модели и возвращать значения постоянной Хаббла:

```
regr <- function(data, indices) {
# вектор indices будет формироваться функцией boot()
  dat <- data[indices, ]
  fit <- lm(y ~ -1 + x, data = dat)
  return(summary(fit)$coefficients[1])
}
```

Теперь подадим **regr**() на функцию **boot**():

```
library(boot)
results <- boot(data = hubble, statistic = regr, R = 1000)
```

При выводе содержимого объекта results на экран получим:

```
results
ORDINARY NONPARAMETRIC BOOTSTRAP
Call:
boot(data = hubble, statistic = regr, R = 1000)
Bootstrap Statistics :
    original     bias    std. error
t1* 76.58117 0.03528876    4.812987
```

В представленных результатах original – это значение постоянной Хаббла, оцененное по эмпирическим данным (см. выше); bias (смещение) – разница между средним значением 1000 бутстреп-оценок постоянной Хаббла и исходной оцен-

кой; std. error – бутстреп-оценка стандартной ошибки постоянной Хаббла. Обратите внимание на то, что полученная бутстреп-оценка стандартной ошибки выше, чем рассчитанная параметрическим методом.

Для объектов класса boot, к которому принадлежит results, имеется метод plot, который позволяет изобразить графически полученное распределение бутстреп-оценок постоянной Хаббла и одновременно проверить нормальность их распределения при помощи графика квантилей (рис. 90).

```
plot(results)
```

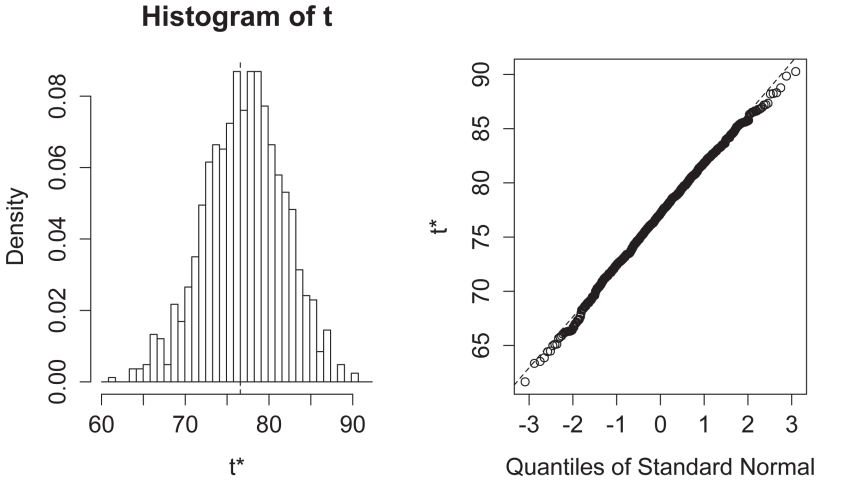


Рисунок 90

Как было отмечено ранее, мы можем оценить нижнюю и верхнюю границы 95%-го доверительного интервала постоянной Хаббла, найдя 0.025- и 0.975-квантили изображенного выше распределения:

```
quantile(results$t, c(0.025, 0.975))
    2.5%    97.5%
67.07360 85.73249
```

Используя эти значения, рассчитаем соответствующие границы значений возраста Вселенной:

```
U.lower <- 1/(85.73249*60^2*24*365.25/3.09e19)
U.upper <- 1/(67.07360*60^2*24*365.25/3.09e19)

U.lower
[1] 11421129999 # то есть примерно 11.4 млрд лет

U.upper
[1] 14598320553 # то есть примерно 14.6 млрд лет
```

Следует подчеркнуть, что границы доверительного интервала, рассчитанные «методом процентилей», могут оказаться смещенными. В состав пакета boot входит функция **boot.ci**(), которая позволяет рассчитать несколько других типов доверительных интервалов, включая интервалы с поправкой на смещение (аргумент type = "bca"; подробнее см. ?boot.ci):

```
boot.ci(results, type = "bca")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates
CALL :
boot.ci(boot.out = results, type = "bca")
Intervals :
Level       BCa
95%   (66.44, 85.36 )
Calculations and Intervals on Original Scale
```

3. *Имитационные процедуры.* Как было показано в разделе 7.1, можно также использовать принципы *байесовской статистики* и оценить неопределенность в отношении параметров той или иной модели методом имитаций. В общем виде эта процедура включает следующие шаги:

- подгоняется определенная модель к имеющимся данным;
- полученные оценки параметров модели используются для генерации большого числа альтернативных, но в то же время правдоподобных (то есть не противоречащих данным) реализаций этих параметров.

Этот метод легко реализовать с помощью функции **sim**() из пакета arm (Gelman & Hill, 2007), работа которой построена на выполнении следующих шагов:

1. При помощи обычного регрессионного анализа оцениваются вектор регрессионных коэффициентов $\hat{\beta}$, ковариационная матрица $V_{\hat{\beta}}$ и стандартное отклонение остатков $\hat{\sigma}$.
2. Создается большое число реализаций оценок вектора параметров β^* и стандартного отклонения остатков σ^*. Для каждой реализации:
   - генерируется значение $\sigma^* = \hat{\sigma}\sqrt{(n-k)/\chi^2}$, где статистика χ^2 случайным образом извлекается из соответствующего распределения с $n - k$ степенями свободы;
   - с учетом полученного значения σ^* формируется вектор β^* путем случайного извлечения значений из многомерного нормального распределения со средним значением $\hat{\beta}$ и ковариационной матрицей $V_{\hat{\beta}}$.

Для нашего примера получаем:

```
library(arm)
simulations <- sim(M, 1000)
hist(simulations@coef, breaks = 30)
```

Гистограмма распределения 1000 имитированных значений постоянной Хаббла по форме практически не отличается от приведенной выше гистограммы, полученной бутстреп-методом.

Рассчитав стандартное отклонение этого распределения, мы можем оценить стандартную ошибку постоянной Хаббла и увидеть, что она, как и в случае с бутстреп-анализом, также оказалась выше исходной оценки (см. **summary**(M)):

```
sd(simulations@coef)
[1] 4.051294
```

95%-ный доверительный интервал легко можно найти известным нам способом:

```
quantile(simulations@coef, c(0.025, 0.975))
    2.5%    97.5%
68.11832 84.31069
```

Конечно, может возникнуть вопрос: какой из описанных выше трех методов следует применять на практике? Ответ на этот вопрос определяется несколькими факторами. Так, если модель адекватно описывает данные (с точки зрения ее структуры и выполнения требований к ее остаткам), то рассмотренный в самом начале классический подход даст хороший результат. Бутстреп-анализ и имитации, возможно, окажутся слишком «тяжелой артиллерией» для характеристики неопределенности в отношении одного-единственного коэффициента простой регрессии. В то же время эти методы незаменимы при оценке неопределенности в отношении величин, которые являются функцией (особенно нелинейной функцией) от каких-либо параметров модели. Например, при выполнении бутстреп-анализа мы могли бы рассчитывать для каждой «псевдовыборки» не просто постоянную Хаббла, а сразу возраст Вселенной, изменив соответствующим образом функцию **regr**().

Оценка «качества» регрессионной модели

В нашем примере с постоянной Хаббла оценка коэффициента β оказалась статистически значимой, что является косвенным подтверждением адекватности модели имеющимся данным. Однако при необходимости сравнить между собой несколько альтернативных моделей, отличающихся по своей структурной части и/или составу исходных данных, желательно получить более точный ответ на вопрос, какая из возможных спецификаций модели является более предпочтительной.

Многие руководства по статистике предлагают ориентироваться на формальные критерии качества аппроксимации, такие как *среднеквадратичная ошибка регрессии*, *коэффициент детерминации* и *дисперсионное отношение Фишера*. Эти показатели, вычисляемые при помощи функции **lm**(), образуют завершающий блок результатов, выводимых при помощи **summary**():

```
...
Residual standard error: 258.9 on 23 degrees of freedom
Multiple R-squared:  0.9419, Adjusted R-squared:  0.9394
F-statistic: 373.1 on 1 and 23 DF,  p-value: 1.032e-15
```

В третьей строке приведенного блока результатов фигурируют значение *F*-критерия и соответствующее ему *p*-значение. В разделе 5.5 мы рассматривали дисперсионный анализ, где *F*-критерий использовался для сравнения меж- и внутригрупповой дисперсий. Поскольку дисперсионный анализ и линейная регрессия с математической точки зрения являются частными случаями одного и того же класса общих линейных моделей (см. раздел 6.2), здесь *F*-критерий также представляет собой отношение дисперсий с похожей интерпретацией:

$$F = \frac{(TSS - RSS)/k}{RSS/(n-k-1)},$$

где $TSS = \sum_{i=1}^{n}(y_i - \overline{y})^2$ – общая сумма квадратов («*total sum of squares*»), $RSS = \sum_{i=1}^{n}(y_i - \hat{y})^2$ – сумма квадратов остатков, $k = 2$ – число параметров модели (коэффициент регрессии и стандартное отклонение остатков), n – объем выборки.

Ниже приведена геометрическая интерпретация понятий «общая сумма квадратов» (*TSS*) и «сумма квадратов остатков» (*RSS*). Предположим, что мы не строим никакой модели и пытаемся предсказать зависимую переменную *y*, просто рассчитав ее среднее значение. Вертикальные отрезки на диаграмме, представленной на рис. 91 слева, отражают расстояние от каждого выборочного наблюдения *y* до этого предсказанного среднего значения (представлено в виде синей горизонтальной линии). При расчете *TSS* все эти расстояния возводятся в квадрат и суммируются. Вертикальные отрезки на рис. 91 справа отражают расстояние от каждого выборочного значения зависимой переменной *y* до значения, предсказанного регрессионной моделью ($\hat{y}$). При расчете *RSS* все эти расстояния также возводятся в квадрат и суммируются.

Выполнить разложение дисперсии отклика *y* на описанные составляющие и получить стандартную таблицу дисперсионного анализа можно с использованием функции **anova**():

```
anova(M)
Analysis of Variance Table
Response: y
          Df   Sum Sq  Mean Sq F value    Pr(>F)
x          1 25013691 25013691  373.08 1.032e-15 ***
Residuals 23  1542066    67046
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Очевидно, что чем меньше значение *RSS* (то есть чем ближе регрессионная линия ко всем наблюдениям одновременно), тем больше будет значение *F*-критерия и тем лучше построенная регрессионная модель будет *в целом* описывать экспериментальные данные. Другими словами, рассчитывая *F*-критерий, мы проверяем нулевую гипотезу о равенстве всех регрессионных коэффициентов построенной модели нулю (так называемый «*omnibus test*»):

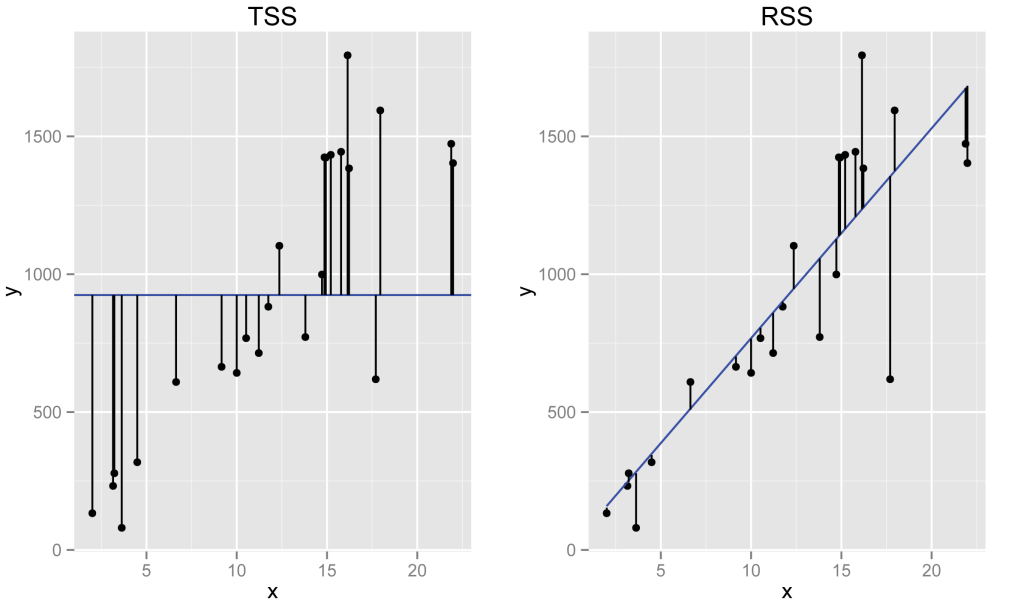


Рисунок 91

H_0: $\beta_1 = \beta_2 = ... = \beta_p = 0$.

В приведенном виде эта нулевая гипотеза соответствует случаю множественной регрессии (то есть когда есть несколько предикторов). В рассматриваемом нами примере мы оценивали лишь один параметр (постоянную Хаббла), и *F*-критерий составил `373.1`, что гораздо больше 1. Этот результат означает, что вероятность получить столь же высокое значение *F* при отсутствии связи между *x* и *y* очень мала (`p = 1.032e-15`). Поскольку у нас есть лишь один предиктор, то мы одновременно проверили гипотезы о статистической *значимости модели в целом* и о равенстве нулю постоянной Хаббла β (обратите внимание на то, что *p*-значения в обоих этих тестах совпадают).

Во второй снизу строке результатов расчета модели приведены значения `Multiple R-squared` и `Adjusted R-squared`. В первом случае речь идет о так называемом *коэффициенте детерминации*, который обозначается как R^2 и рассчитывается как $R^2 = 1 - RSS/TSS$. Напомним, что *TSS* отражает общий разброс значений зависимой переменной без учета предикторов, а *RSS* – остаточную дисперсию значений отклика, которую нам не удалось «объяснить» при помощи построенной модели.

Таким образом, коэффициент детерминации R^2 измеряет *долю общей дисперсии зависимой переменной, объясненную моделью*, и изменяется от 0 до 1. Чем ближе значение R^2 к 1, тем точнее модель описывает данные: для случая простой регрессии с одним предиктором R^2 будет представлять собой просто возведенный в квадрат коэффициент корреляции между *y* и *x*. Достигнутое нами значение коэффици-

ента детерминации составило R² = 0.9419, что указывает на весьма высокое качество рассматриваемой модели.

Скорректированный коэффициент детерминации («*adjusted R-squared*») применяется для сравнения моделей-претендентов с разным количеством предикторов и будет обсуждаться нами в последующих разделах. Здесь мы лишь приведем формулу для его расчета:

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-k-1},$$

где k – это число параметров модели. Отметим, что $R^2 \geq R^2_{adj}$.

В первой строке приведенных выше результатов регрессионного анализа представлено значение Residual standard error – стандартное отклонение остатков модели, которое в общем виде рассчитывается как $RSE = \sqrt{RSS / (n-k-1)}$. Показатель *RSE* является оценкой разброса наблюдаемых значений зависимой переменной относительно истинной линии регрессии. Так, в нашем примере *RSE* = 258.9, из чего следует, что в среднем наблюдаемые значения скорости галактик отличаются от истинных значений на 258.9 км/сек. Очевидно, что чем меньше значение *RSE*, тем точнее модель описывает анализируемые данные.

Считается, что представленные количественные показатели отражают разные аспекты «качества» регрессионной модели, и поэтому их стоит использовать в совокупности. На самом деле они лишь по-разному комбинируют суммы квадратов отклонений *RSS* и *TSS*. В дальнейшем мы рассмотрим дополнительные критерии качества модели, основанные на иных принципах.

7.3. Стандартные методы диагностики линейных моделей

Оценка параметров линейных регрессионных моделей, равно как и выводы в отношении статистической значимости этих параметров, базируется на выполнении ряда математических допущений. Диагностика выполнения этих допущений является составной частью процесса построения регрессионной модели и сводится к следующим составляющим (Faraway, 2004):

- проверка допущений в отношении остатков модели;
- проверка адекватности структуры систематической части модели;
- обнаружение необычных наблюдений.

Существуют как графические, так и формальные методы диагностики линейных моделей. Хотя формальные методы используются реже, они доступны в нескольких пакетах для R (например, car и lmtest; см. также раздел 7.8). Этот раздел посвящен более распространенным графическим методам.

Проверка допущений в отношении остатков модели

Одно из наиболее важных допущений при работе с линейными моделями, параметры которых оцениваются методом наименьших квадратов, состоит в том, что

остатки модели независимы (то есть не коррелируют) и имеют нормальное распределение со средним значением 0 и некоторым фиксированным стандартным отклонением σ, то есть $\varepsilon_i \sim N(0, \sigma)$ (см. предыдущий раздел). Мы уже сталкивались с этими условиями в разделе 6.4 при рассмотрении моделей дисперсионного анализа (что неудивительно, поскольку дисперсионный анализ является частным случаем линейной модели) и поэтому детально останавливаться на них не будем. Вспомним лишь, что для проверки нормальности распределения количественных переменных можно использовать визуальный анализ гистограмм и квантильных графиков, а также применять формальные методы вроде теста Шапиро-Уилка. Для проверки наличия корреляции в остатках можно использовать графики автокорреляционной функции, а также соответствующие формальные тесты (в частности, тест Дарбина-Уотсона, реализованный, например, в пакете `lmtest`).

Остановимся несколько подробнее лишь на условии однородности дисперсии остатков модели. Это условие предполагает, что остатки имеют постоянную дисперсию во всем диапазоне предсказанных моделью значений зависимой переменной, а также во всем пространстве значений предикторов. Чаще всего данное условие проверяется графическим способом. Прежде всего строится график, на котором по оси абсцисс откладывают предсказанные моделью значения зависимой переменной, а по оси ординат – соответствующие значения остатков. Если условие однородности дисперсии выполняется, то на таком графике точки будут располагаться совершенно случайно относительно горизонтальной линии $y = 0$, без проявления каких-либо закономерностей (см. пример для дисперсионного анализа в разделе 6.4, где это условие не выполнялось).

К сожалению, иногда интерпретация подобных диаграмм может вызывать затруднения, особенно при небольших объемах наблюдений. Поэтому для «тренировки глаза» рекомендуется построить несколько калибровочных графиков со случайно сгенерированными данными, свойства которых заведомо известны (Faraway, 2004). Примеры таких графиков (по 9 для каждого случая) и код для их построения приведены ниже.

Случай однородной дисперсии остатков (рис. 92):

```
par(mfrow = c(3, 3))
set.seed(101)
for (i in 1:9)  plot(1:50, rnorm(50),
                     xlab = "Предсказанные значения",
                     ylab = "Остатки")
```

Случай явно выраженного возрастания дисперсии остатков при увеличении предсказанных значений зависимой переменной (рис. 93):

```
par(mfrow = c(3, 3), mar = c(4.5, 4.4, 2, 2))
set.seed(101) # для воспроизводимости кода
for (i in 1:9)  plot(1:50, (1:50)*rnorm(50),
                     xlab = "Предсказанные значения",
                     ylab = "Остатки")
```

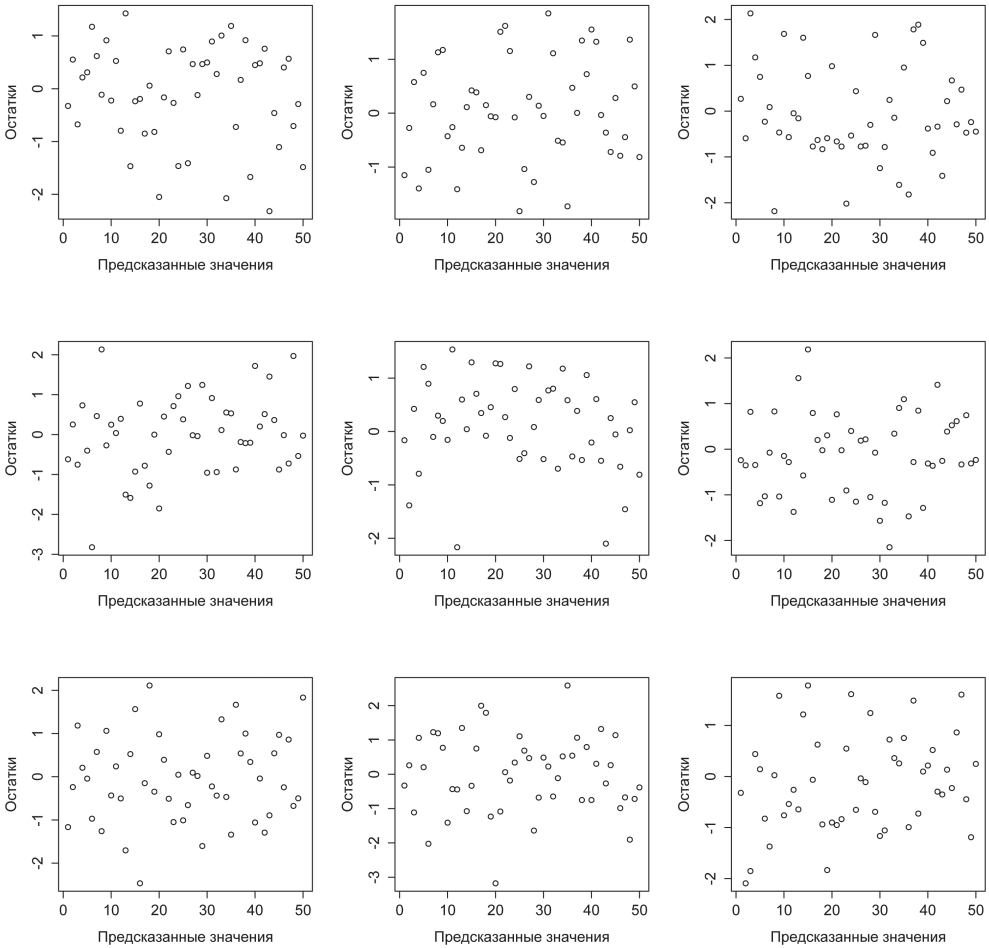


Рисунок 92

Случай умеренно выраженного возрастания дисперсии остатков при увеличении предсказанных значений зависимой переменной (рис. 94):

```
par(mfrow = c(3, 3), mar = c(4.5, 4.4, 2, 2))
set.seed(101) # для воспроизводимости кода
for (i in 1:9)  plot(1:50, sqrt((1:50))*rnorm(50),
                     xlab = "Предсказанные значения",
                     ylab = "Остатки")
```

Помимо построения графиков с предсказанными моделью значениями зависимой переменной, рекомендуется также анализировать графики, где по оси X откладывают значения каждого из *включенных* и *не включенных* в модель предикторов. При наличии проблем с неоднородностью дисперсии остатков этот подход

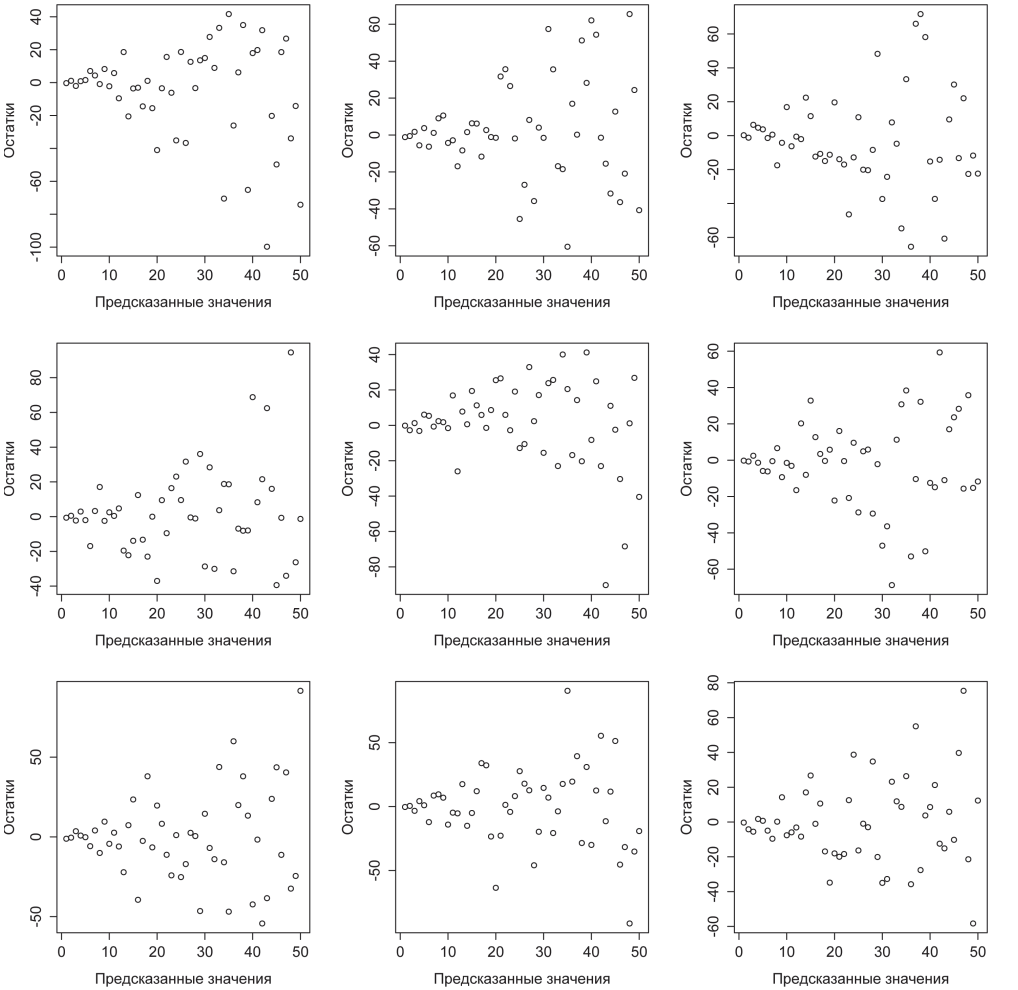


Рисунок 93

позволит выявить «проблемные» предикторы. В зависимости от свойств данных и стоящей перед исследователем задачи способы устранения неоднородности дисперсии остатков могут включать преобразование исходных значений зависимой переменной и/или предикторов, добавление в модель недостающих предикторов, а также выбор другого, более подходящего ситуации, типа модели (Faraway, 2004; Harrell, 2002).

Как было отмечено выше, существуют не только графические, но и формальные методы проверки условия однородности дисперсии остатков. В частности, в пакете `car` имеется функция **`ncvTest`**`()` (от «*non-constant variance test*» – тест на непостоянную дисперсию), которая позволяет проверить нулевую гипотезу о том,

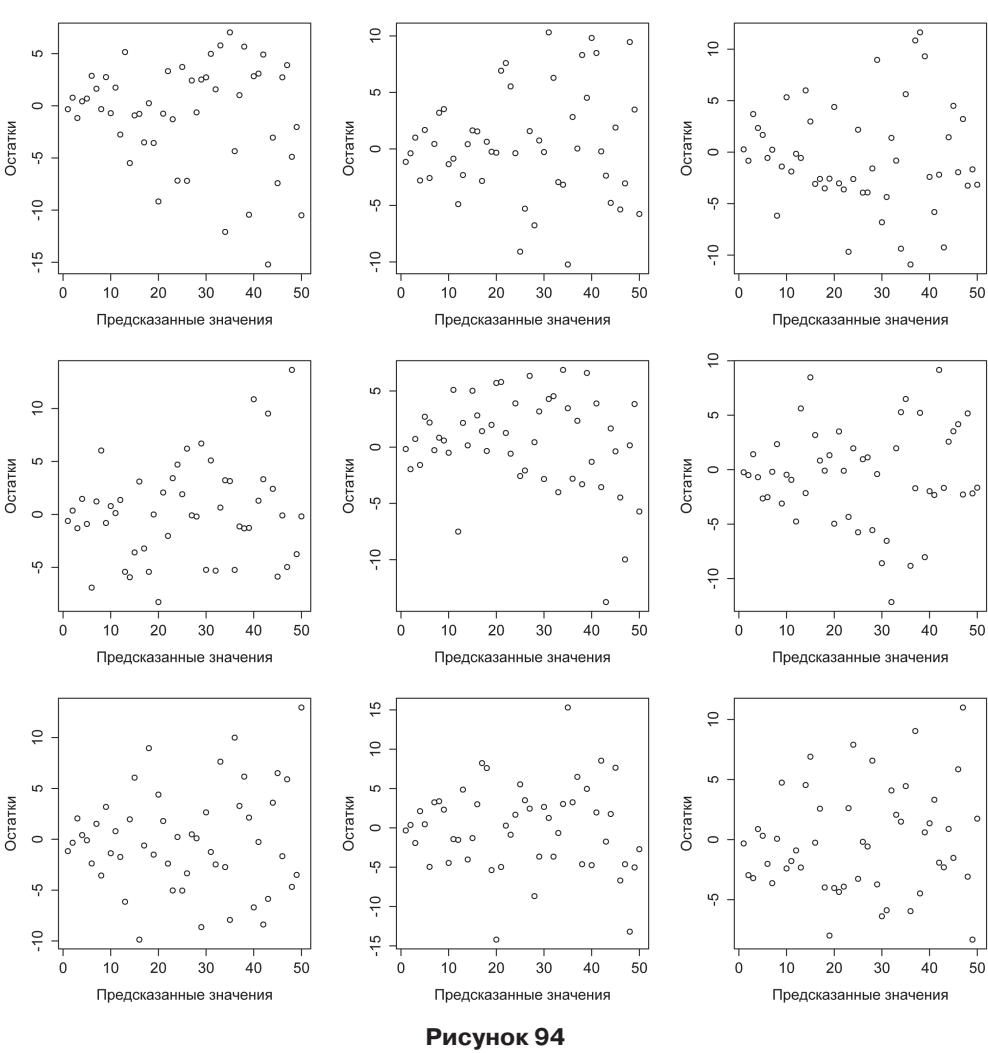


Рисунок 94

что дисперсия остатков никак не связана с предсказанными моделью значениями. Пример использования этой функции здесь не приводится – синтаксис соответствующей команды очень прост (например, `ncvTest(model)`), равно как и интерпретация результатов теста.

Проверка адекватности структуры систематической части модели

Как было отмечено в разделе 7.1, все параметрические статистические модели имеют две части: систематическую и случайную. Так, например, в модели $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$ систематическая часть представлена выражением $[\beta_0 + \beta_1 x_i]$, которое

определяет математическое ожидание (среднее значение) зависимой переменной. В свою очередь, остатки ε_i отражают наличие случайной вариации наблюдаемых значений зависимой переменной относительно ожидаемого среднего значения.

Предположим, что истинная модель, описывающая процесс генерации значений некоторой переменной y, выглядит следующим образом: $y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \varepsilon_i$, где $\beta_0 = 0.1$, $\beta_1 = 0.2$, $\beta_2 = 0.035$ и $\varepsilon_i \sim N(0, 10)$. Ниже приведен код, позволяющий сгенерировать 50 значений y на основе этой модели для $x = 1, 2, ..., 50$ и изобразить соответствующую зависимость графически (рис. 95).

```
x = 1:50
set.seed(200)
y = rnorm(50, 0.1 + 0.2*x + 0.035*(x^2), 10)
plot(x, y)
```

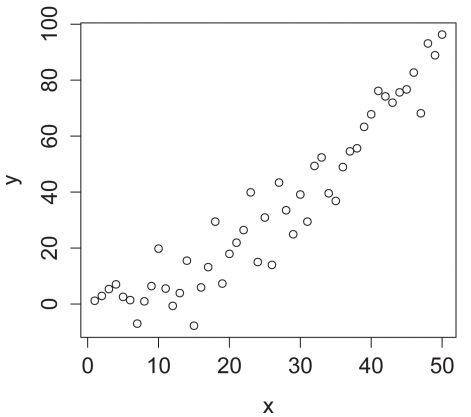


Рисунок 95

К сожалению, на практике исследователю никогда не известны ни структура истинной модели, ни значения ее параметров. Все, что он видит, – это данные вроде тех, что изображены на рис. 95. По имеющимся данным бывает очень сложно (а порой и просто невозможно) сделать верное предположение касательно структуры систематической части модели. Так, в рассматриваемом примере мы могли бы начать с подгонки простой регрессионной модели $y_i = \beta_0 + \beta_1 x_{i1} + \varepsilon_i$:

```
M1 <- lm(y ~ x)
summary(M1)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-21.927  -5.853   1.328   7.870  15.264
 Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -14.754      2.959  -4.986 8.45e-06 ***
x              1.929      0.101  19.103  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.31 on 48 degrees of freedom
Multiple R-squared:  0.8838, Adjusted R-squared:  0.8813
F-statistic: 364.9 on 1 and 48 DF,  p-value: < 2.2e-16
```

Казалось бы, все указывает на то, что модель `M1` хорошо описывает данные: сама модель в целом ($p << 0.05$, F-тест) и все ее коэффициенты статистически значимы ($p << 0.05$, t-тесты), а скорректированный коэффициент детерминации очень высок (0.881). Графически эта модель представлена на рис. 96 в виде прямой линии.

```
plot(x, y)
abline(coef(M1)[1], coef(M1)[2], col = "blue")
```

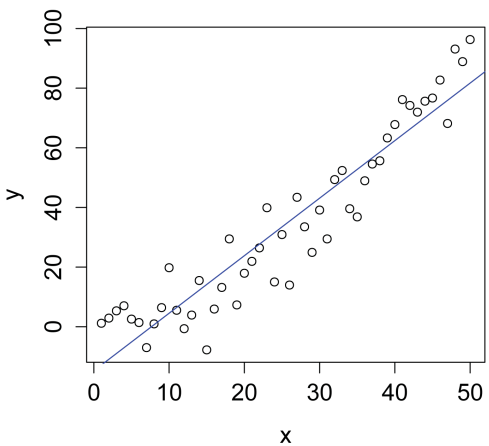


Рисунок 96

Глядя на графическое представление модели `M1`, можно заметить, что она все же не так хорошо подходит для описания имеющихся данных: видно, что в области низких и высоких значений x модель несколько «недооценивает» значения y, тогда как в середине диапазона значений x модель имеет тенденцию к «переоцениванию» значений y (рис. 96). Другими словами, есть основания полагать, что связь между этими двумя переменными не является прямолинейной. Это становится особенно очевидным при анализе графика зависимости остатков модели от подогнанных значений – на этом графике остатки образуют U-образную фигуру, что прямо указывает на нелинейный характер зависимости y от x (рис. 97).

```
plot(fitted(M1), resid(M1),
     xlab = "Предсказанные значения",
     ylab = "Остатки")
```

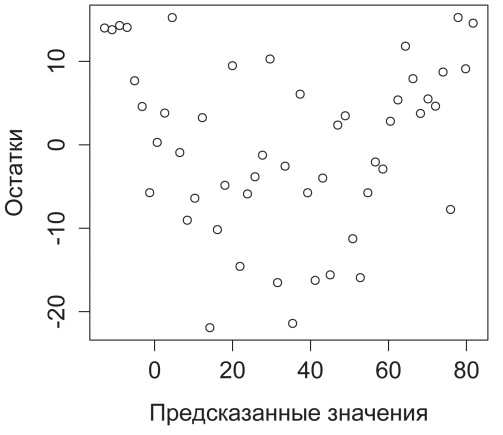


Рисунок 97

Таким образом, анализ остатков приводит нас к заключению о необходимости изменения структуры систематической части модели. Учесть нелинейный характер зависимости y от x можно несколькими способами. Например, мы можем добавить в модель `M1` еще один предиктор, значения которого получены путем возведения в квадрат значений исходного предиктора x:

```
M2 <- lm(y ~ x + I(x^2))
summary(M2)

Call:
lm(formula = y ~ x + I(x^2))

Residuals:
     Min       1Q   Median       3Q      Max
-18.5271  -4.2275   0.4925   4.8585  17.2978

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.579515   3.537067   0.164    0.871
x           0.159959   0.319949   0.500    0.619
I(x^2)      0.034692   0.006082   5.704 7.53e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.006 on 47 degrees of freedom
Multiple R-squared:  0.9313, Adjusted R-squared:  0.9284
F-statistic: 318.6 on 2 and 47 DF,  p-value: < 2.2e-16
```

Модель `M2` воспроизводит структуру истинной модели, на основе которой были сгенерированы данные. Как следствие полученные точечные оценки пара-

метров модели M2 в целом близки к соответствующим истинным значениям: 0.58 против $\beta_0 = 0.1$, 0.16 против $\beta_1 = 0.2$, 0.035 против $\beta_2 = 0.035$ и 8.0 против $\sigma = 10$. Интересно, что оценки коэффициентов β_0 и β_1 оказались незначимы ($p = 0.871$ и $p = 0.619$ соответственно), тогда как оценка параметра β_3 значима. Означает ли этот результат, что предиктор x можно исключить из модели, оставив только x^2? Нет. Наблюдаемая ситуация (то есть незначимый регрессионный коэффициент предиктора x при значимом коэффициенте производного предиктора – x^2) довольно обычна и вполне может быть обусловлена недостаточно большим числом наблюдений и свойствами данной случайно сгенерированной выборки. В таких ситуациях рекомендуется оставлять в модели оба предиктора (Harrell, 2002). Кроме того, в целом модель M2 статистически значима ($p << 0.001$, F-тест) и имеет более высокий скорректированный коэффициент детерминации, чем M1 (0.928 против 0.881). Наконец, анализ остатков модели M2 показывает (рис. 98), что структура ее систематической части адекватна имеющимся данным (сравните с рис. 97).

```
plot(fitted(M2), resid(M2),
     xlab = "Предсказанные значения",
     ylab = "Остатки")
```

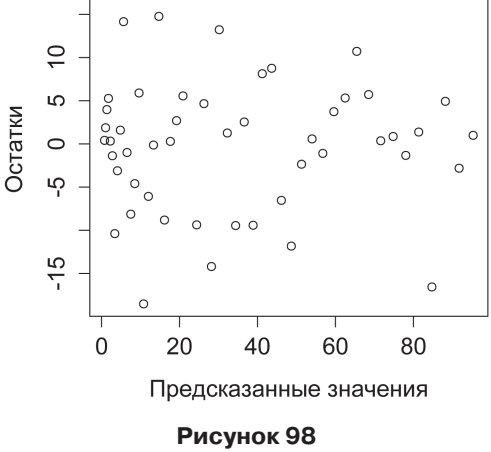


Рисунок 98

Подчеркнем еще раз, что структура модели M2 совпадает со структурой истинной модели, на основе которой были сгенерированы анализируемые данные. Однако в этом примере мы знали, с чем сравнивать. В реальной жизни мы бы так и не узнали о том, что нам повезло подобрать правильную структуру. Например, для описания этих данных вполне можно было попробовать и другие типы моделей, которые тоже показали бы хорошие результаты (например, модель экспоненциального роста или сплайн-модель). На практике выбор окончательной модели должен подкрепляться хорошим пониманием изучаемого явления, а также использованием соответствующих статистических критериев (в частности, информационным критерием Акаике).

Встроенные диагностические графики

В рассмотренных выше примерах диагностические графики были построены «вручную». Однако в R имеется и встроенный метод, позволяющий создавать соответствующие графики автоматически. Для реализации этого метода достаточно просто подать модельный объект на функцию **plot**():

```
par(mfrow = c(2, 2))
plot(M2)
```

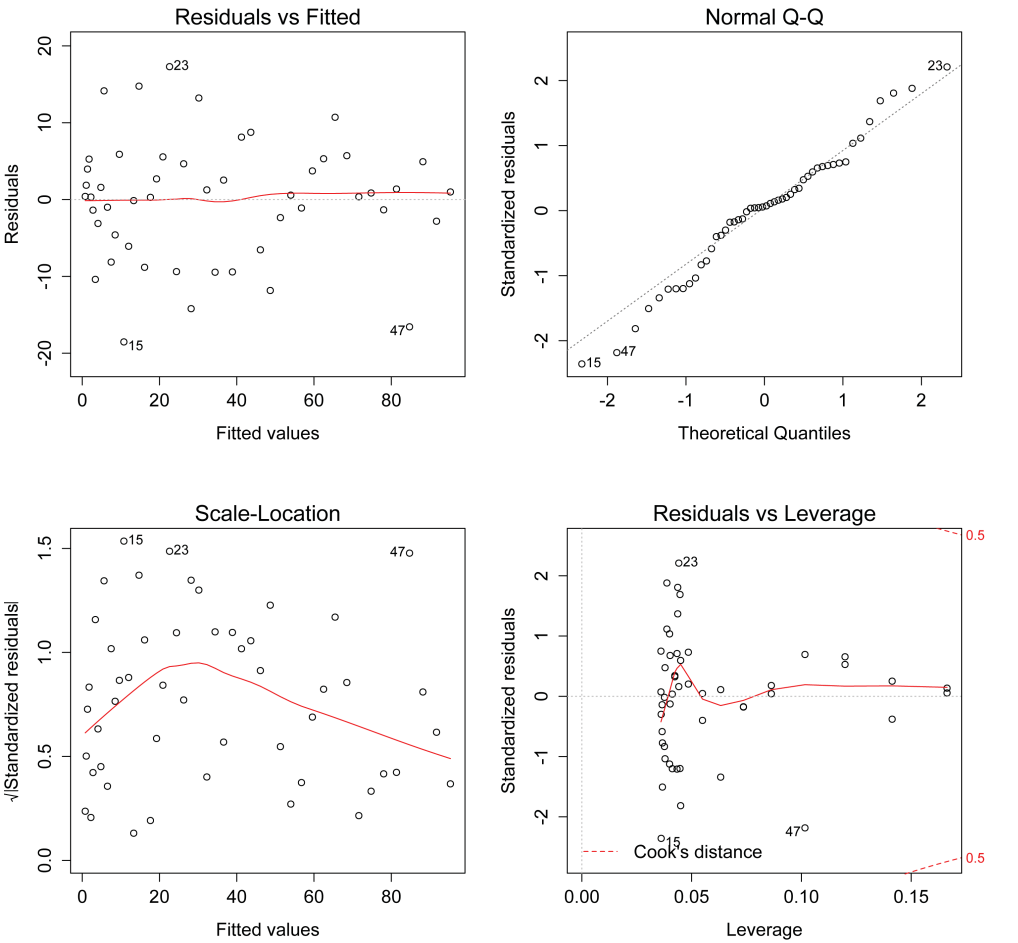


Рисунок 99

Слева вверху на рис. 99 представлен уже знакомый нам тип графиков: он отражает разброс остатков в зависимости от предсказанных моделью значений и служит для диагностики условия однородности дисперсии. К графику добавлена

сглаживающая линия, облегчающая выявление закономерностей в расположении остатков. Примерно горизонтальное расположение этой линии (как в приведенном примере) указывает на отсутствие каких-либо проблем.

Слева внизу на рис. 99 находится очень похожий график – единственное отличие здесь состоит в том, что вместо исходных остатков по оси ординат откладываются значения квадратного корня из абсолютных значений стандартизованных остатков (стандартизация выполняется путем деления каждого остатка на стандартное отклонение, рассчитанное по всей совокупности остатков). Стандартизация остатков облегчает выявление неоднородности их дисперии. Сглаживающая линия в данном случае не является горизонтальной. Однако это скорее обусловлено неодинаковой плотностью точек в разных участках диаграммы, нежели нарушением условия однородности дисперсии.

Справа вверху на рис. 99 представлен график квантилей, позволяющий проверить условие нормальности распределения остатков.

Наконец, *справа внизу* на рис. 99 находится график, предназначенный для обнаружения наблюдений, оказывающих существенное влияние на оценки параметров модели. Пунктирными изолиниями на этом графике показаны пороговые значения так называемого расстояния Кука («*Cook's distance*»), о котором мы поговорим подробнее ниже. Точки, выходящие за пределы этих изолиний, следует считать влиятельными.

На всех четырех графиках на рис. 99 есть несколько точек, отмеченных их порядковыми номерами. Этот прием используется для того, чтобы сообщить исследователю о «необычных» свойствах этих конкретных точек. Интерпретация выражения «необычный» завист от контекста, то есть от того, о каком из приведенных графиков идет речь. Отмеченные точки не обязательно являются «проблемными» – просто исследователю следует обратить на них внимание. В следующем подразделе мы поговорим о таких необычных наблюдениях подробнее.

Выявление необычных и влиятельных наблюдений

Говоря о необычных наблюдениях в контексте регрессионного анализа, можно выделить следующие три ситуации:

- наблюдение представлено *необычным сочетанием* значений предикторов. В англоязычной литературе такие наблюдения, которые *потенциально* могут оказывать заметное влияние на оценки параметров модели, носят название «*leverage points*»;
- наблюдение не согласуется с рассматриваемой моделью, то есть является *выбросом* («*outlier*»);
- наблюдение оказывает существенное влияние на оценки параметров модели («*influential point*», или «*influential observation*»). Другими словами, удаление такого влиятельного наблюдения из выборки приведет к значительному изменению предсказываемых моделью значений. Влиятельные наблюдения обладают как минимум одним из двух указанных выше свойств (то есть являются либо «*leverage points*», либо «*outliers*»), но чаще всего сочетают их.

Необычные наблюдения могут оказывать существенное влияние на качество модели (как с точки зрения статистической значимости ее параметров, так и с точки зрения ее предсказательной силы), в связи с чем выявление таких наблюдений является важной частью диагностики регрессионных моделей.

Наблюдения, представленные необычным сочетанием предикторов и, как следствие, расположенные далеко от центра (многомерного) распределения наблюдаемых значений предикторов, *потенциально* могут оказывать значительное влияние на оценки параметров модели (рис. 100). О таких наблюдениях говорят, что они обладают «*высоким потенциалом воздействия*» («*high leverage*») на параметры модели. На рис. 100 приведен пример ядерной оценки плотности вероятности двумерного распределения. Примерные координаты центра этого распределения – (10, 2). Видно, что некоторые точки находятся достаточно далеко от центра – в частности, точка, выделенная желтым цветом. Это необычное для данного распределения наблюдение потенциально (но необязательно) может оказать существенное влияние на оценки параметров линейной модели, включающей предикторы x_1 и x_2.

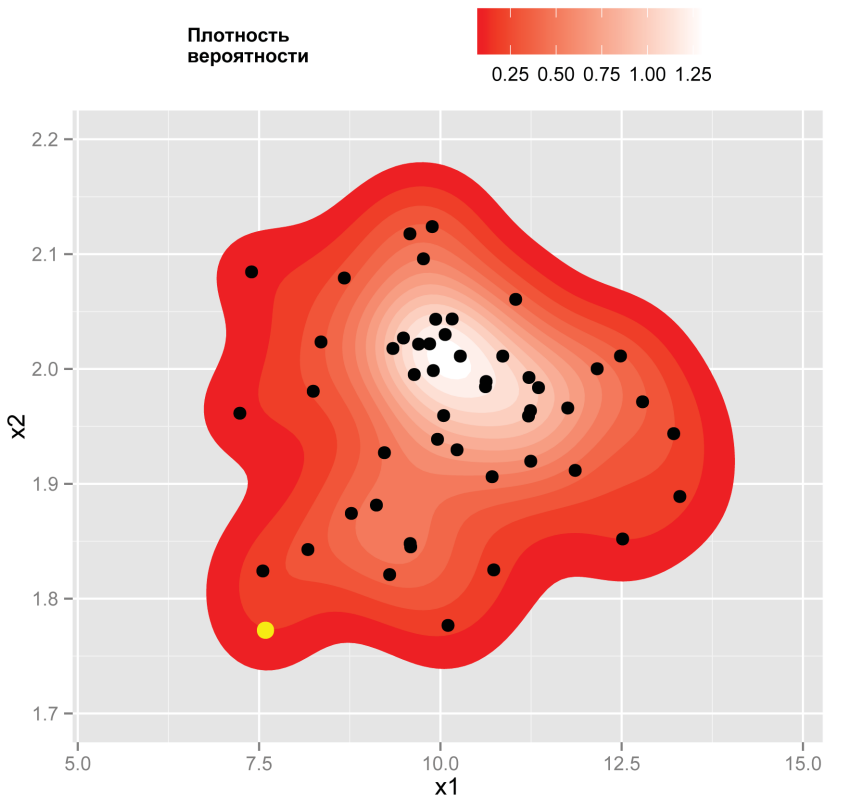


Рисунок 100

Имеется возможность выразить этот потенциал воздействия количественно. Распространенным подходом является расчет диагональных элементов так называемой *матрицы проекции на пространство предикторов* («*hat matrix*»; употребляются также названия «матрица влияния» и «матрица воздействия» («*influence matrix*»)), которые обозначаются как h_i. Английский термин «*hat matrix*» происходит от названия значка ^ («*hat*» – шапка), который обычно используется в обозначении предсказываемых моделью значений зависимой переменной – $\hat{y}$.

Чтобы понять, откуда берется матрица проекции и что она собой представляет, необходимо прибегнуть к записи линейных моделей с использованием нотации линейной алгебры. В общем виде мы можем представить линейную модель следующим образом:

$$\mathbf{y} = \mathbf{X\beta} + \boldsymbol{\varepsilon},$$

где $\mathbf{X}$ – матрица модели (размером $n \times p$, где p – число регрессионных коэффициентов + свободный член уравнения; см. также раздел 6.2), $\boldsymbol{\beta}$ – вектор подлежащих оценке регрессионных коэффициентов, а $\boldsymbol{\varepsilon}$ – вектор остатков с нулевым математическим ожиданием.

Вектор с оценками регрессионных коэффициентов $\hat{\boldsymbol{\beta}}$ получают следующим образом:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{y},$$

откуда предсказываемые моделью значения зависимой переменной можно записать как

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{y},$$

или в более простой форме как

$$\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}.$$

Матрица $\mathbf{H}$ (размером $n \times n$) как раз и является матрицей проекции, поскольку она позволяет выполнить линейное преобразование наблюдаемых значений y так, чтобы получить предсказываемые значения (подробнее см., например, Hoaglin & Welsch (1978, http://bit.ly/1BQ8Kbc).

Рассмотрим на примере некоторые полезные свойства матрицы проекции. Предположим, у нас имеются два предиктора x_1 и x_2 (см. также рис. 100):

```
set.seed(604)
df = data.frame(x1 = rnorm(50, 10, 1.5),
                x2 = rnorm(50, 2, 0.1))
head(df)
        x1       x2
1  7.235169 1.961452
2 12.788041 1.971398
3  9.640045 1.995138
4 11.353070 1.983749
5 11.242469 1.963742
6 11.038517 2.060605
```

Предположим далее, что переменные x_1 и x_2 связаны с некоторой переменной y следующей зависимостью:

```
set.seed(101)
df$y = 2*df$x1 - 25*df$x2 + rnorm(50, 0, 2)
```

По имеющимся данным мы можем восстановить регрессионную модель при помощи функции **lm**():

```
M <- lm(y ~ x1 + x2, data = df)
summary(M)

Call:
lm(formula = y ~ x1 + x2, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-4.3855 -1.2553  0.3463  1.2528  2.8974

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -12.1994     6.0746  -2.008   0.0504 .
x1            2.2655     0.1766  12.827  < 2e-16 ***
x2          -20.2806     2.9861  -6.792 1.69e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.811 on 47 degrees of freedom
Multiple R-squared:  0.8137, Adjusted R-squared:  0.8057
F-statistic: 102.6 on 2 and 47 DF,  p-value: < 2.2e-16
```

Для расчета матрицы проекции сначала сформируем матрицу модели, для чего воспользуемся уже известной нам функцией **model.matrix**():

```
X = model.matrix(y ~ x1 + x2, data = df)

head(X)
  (Intercept)        x1       x2
1           1  7.235169 1.961452
2           1 12.788041 1.971398
3           1  9.640045 1.995138
4           1 11.353070 1.983749
5           1 11.242469 1.963742
6           1 11.038517 2.060605
```

Получить **H** теперь очень просто (в приведенной ниже команде %*% – это оператор умножения матриц, **solve**() – функция, выполняющая обращение матриц, а **t**() – функция, выполняющая транспонирование матриц):

```
H <- X %*% solve( t(X) %*% X ) %*% t(X)
```

Для расчета предсказываемых моделью значений воспользуемся приведенной выше формулой:

```
Yhat <- H %*% df$y
```

Легко проверить, что полученный таким образом вектор предсказанных значений $\hat{\mathbf{y}}$ идентичен вектору, который можно было бы получить более удобным способом при помощи стандартной функции **fitted**():

```
identical(round(fitted(M), 2), round(H %*% df$y, 2)[, 1])
[1] TRUE
```

Диагональные элементы матрицы проекции изменяются от 0 до 1 и отражают *потенциал воздействия* отдельных наблюдений на оценки регрессионных коэффициентов. Чем дальше то или иное наблюдение находится от центра многомерного распределения значений предикторов, тем выше будет соответствующий диагональный элемент. Эти элементы можно получить несколькими эквивалентными способами:

```
hat(X) # Способ 1
diag(H) # Способ 2
Minf <- influence(M) # Способ 3
Minf$hat
```

Оказывается, что сумма диагональных элементов матрицы проекции равна числу коэффициентов регрессионного уравнения, включая свободный член:

```
sum(hat(X))
[1] 3
```

Соответственно, среднее значение h_i можно рассчитать как p/n. Отсюда вытекает эмпирическое правило, позволяющее судить о том, оказывает ли некоторое наблюдение существенное влияние на параметры модели, – значения $h_i > 2p/n$ являются достаточно большими, чтобы считать соответствующие наблюдения стоящими внимания (Fox & Weisberg, 2010).

Значения h_i удобнее всего анализировать графически. Так, для нашего примера имеем (рис. 101):

```
plot(hat(X), type = "h")
abline(h = 2*3/50, lty = 2, col = "blue")
```

На рис. 101 изображены показатели потенциала воздействия для каждого из 50 наблюдений, полученные на основе приведенной выше модели M. Согласно пороговому значению $2p/n$ (показано горизонтальной пунктирной линией), четыре из 50 имеющихся наблюдений (под номерами 24, 25, 40 и 44) имеют высокий потенциал воздействия на коэффициенты этой модели и требуют дополнительного изучения.

В контексте регрессионного анализа *выбросом* называют наблюдение, которое не согласуется с рассматриваемой моделью. Другими словами, выброс – это наблюдение с большим остатком, возникающим из-за того, что соответствующее

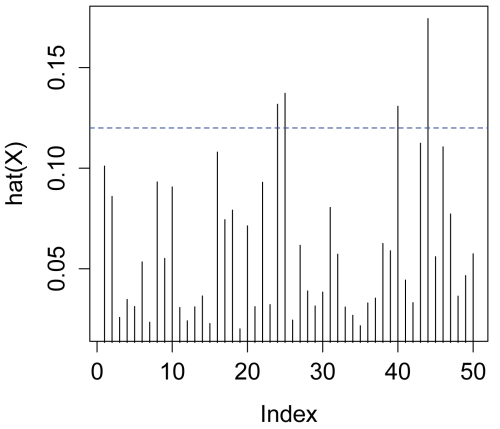


Рисунок 101

выборочное значение зависимой переменной y_i значительно отличается от предсказанного значения $\hat{y}_i$. Для выявления выбросов можно использовать уже рассмотренные нами ранее диагностические графики, на которых по оси абсцисс откладывают предсказываемые моделью значения, а по оси ординат – остатки (см., например, рис. 98). На практике, однако, при работе с такими «сырыми» значениями остатков (*«raw residuals»*) бывает трудно сказать, какие из них достаточно велики. Кроме того, значения «сырых» остатков зависят от шкалы, по которой измеряется зависимая переменная. Для устранения этих недостатков применяют разного рода стандартизацию. Следует отметить, что терминология, используемая в разных источниках для обозначения таких преобразованных остатков, крайне запутанна. В R для диагностики линейных моделей, чьи параметры оцениваются по методу наименьших квадратов, используются следующие два типа остатков.

Standardized residuals – собственно *стандартизованные остатки*:

$$r_i = \frac{\varepsilon_i}{\sigma\sqrt{1-h_i}},$$

где ε_i – остаток i-го наблюдения, σ – стандартное отклонение («сырых») остатков модели, а h_i – уже знакомый нам показатель потенциала воздействия i-го наблюдения на коэффициенты модели.

Такие остатки имеют приближенно стандартное нормальное распределение. Соответственно, наблюдения, чьи стандартизованные остатки выходят за пределы диапазона от –2 до 2, считают выбросами. Для нашего примера получаем (заметьте, что для расчета `r` ниже мы могли бы применить также базовую функцию **`rstandard()`**):

```
r <- residuals(M)/(summary(M)$sig*sqrt(1 - hat(X)))
plot(fitted(M), r, ylim = c(-3, 3))
abline(h = 0, lty = 2)
abline(h = c(-2, 2), lty = 2, col = "blue")
```

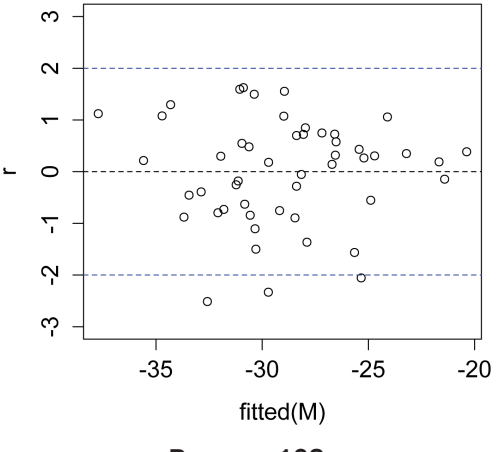


Рисунок 102

Как видно по рисунку, наблюдения 20, 34 и (с натяжкой) 43 можно было бы считать выбросами.

Одним из важных недостатков описанных выше стандартизованных остатков является тот факт, что при расчете i-го значения r используется оценка σ, которая включает значение соответствующего i-го «сырого» остатка. Другими словами, любые значения r_i и σ, строго говоря, не являются независимыми – обстоятельство, которое затрудняет формальную проверку статистической гипотезы о том, что некоторое i-е наблюдение не является выбросом. Для устранения указанного недостатка используют так называемые *стьюдентизированные остатки* («*studentized residuals*»):

$$t_i = \frac{\varepsilon_i}{\sigma_{(-i)}\sqrt{1-h_i}}.$$

Главным отличием этой формулы от предыдущей является наличие индекса $(-i)$ в обозначении стандартного отклонения остатков: $\sigma_{(-i)}$. Этот индекс указывает на то, что стандартное отклонение рассчитывается по остаткам модели, подогнанной *после исключения из данных i-го наблюдения* (то есть, по сути, используется так называемый «метод перочинного ножа» – «*jackknife*»). Поскольку мы рассчитываем как бы «внешнюю» по отношению к i-му наблюдению оценку стандартного отклонения (i-е наблюдение исключается из расчета и не влияет на эту оценку), такие остатки известны также под названием «*externally studentized residuals*» (*внешне стьюдентизированные остатки*). Соответственно, то, что мы выше определили как стандартизованные остатки, в ряде руководств называют «*internally studentized residuals*» (*внутренне стьюдентизированные остатки*) (Hoaglin & Welsch, 1978; Fox & Weisberg, 2010).

Как следует из их названия, стьюдентизированные остатки имеют t-распределение с $n - p - 1$ степенями свободы. Следовательно, мы можем использовать

квантили этого распределения для проверки того, насколько статистически значимо определенное наблюдение является выбросом. Так, в случае с нашим примером мы можем проверить, является ли статистически значимым выбросом наблюдение с наибольшим (абсолютным) значением стьюдентизированного остатка (в R для расчета стьюдентизированных остатков служит функция **rstudent**()):

```
max(abs(rstudent(M)))
[1] 2.671676

abs(qt(.05/(50*2), 50-3-1)) # с поправкой Бонферрони
[1] 3.514957
```

Как видим, максимальное наблюдаемое значение стьюдентизированного остатка (2.672) не превышает критического *t*-значения = 3.515 (рассчитанного с применением поправки Бонферрони). Из этого следует вывод, что данное наблюдение (№ 20) не является статистически значимым выбросом. Аналогичный вывод справедлив и для других наблюдений, чьи стьюдентизированные остатки еще меньше, чем 3.515.

Таким образом, применяя стьюдентизированные остатки, мы не обнаружили каких-либо наблюдений, явно не согласующихся с моделью M. Для сравнения отметим, что в выполненном ранее анализе мы выявили три наблюдения с высоким потенциалом воздействия на параметры модели. Такая ситуация обычна: наблюдения с высоким потенциалом воздействия необязательно являются выбросами (и наоборот).

Выполнив проверку на наличие наблюдений с необычными сочетаниями значений предикторов и наблюдений, которые не согласуются с рассматриваемой моделью, обычно переходят к обобщению полученной информации в виде количественных показателей, отражающих собственно степень влияния каждого наблюдения на параметры модели.

Говоря, что некоторое наблюдение является *влиятельным*, мы имеем в виду, что оно вносит существенный вклад в оценки параметров модели. Поэтому наиболее простой и очевидный способ измерения степени влиятельности заключается в *удалении конкретного наблюдения* из выборки и последующем расчете оценок параметров модели без него. Если удаление приводит к значительному «скачку» в оценках тех или иных параметров модели, значит, это наблюдение является влиятельным. Подобные изменения оценок параметров модели можно записать следующим образом:

$$d_{ij} = b_j - b_{j(-i)},$$

где $b_{j(-i)}$ обозначает оценку j-го параметра модели (то есть β_j), полученную по методу наименьших квадратов после удаления i-го наблюдения ($i = 1, ..., n$, а $j = 1, ..., k$).

Для облегчения интерпретации полезно стандартизовать каждое значение d_{ij} путем его деления на стандартную ошибку соответствующего коэффициента:

$$d_{ij}^* = \frac{d_{ij}}{SE_{(-i)}(b_j)}.$$

Во многих источниках по регрессионному анализу величину d_{ij} называют еще $DFBETA_{ij}$, а d_{ij}^{*} – $DFBETAS_{ij}$. Для расчета эти двух показателей в R имеются одноименные функции – **dfbeta**() и **dfbetas**(). Удобнее всего анализировать эти показатели графически. Например, следующим образом можно изобразить стандартизованные показатели влиятельности каждого наблюдения на параметры нашей модели M (рис. 103):

```
par(mfrow = c(1, 3))
plot(dfbetas(M)[, 1], ylab = "Изменение свободного члена",
     xlab = "Порядковый номер")
plot(dfbetas(M)[, 2], ylab = "Изменение коэффициента x1",
     xlab = "Порядковый номер")
plot(dfbetas(M)[, 3], ylab = "Изменение коэффициента x2",
     xlab = "Порядковый номер")
```

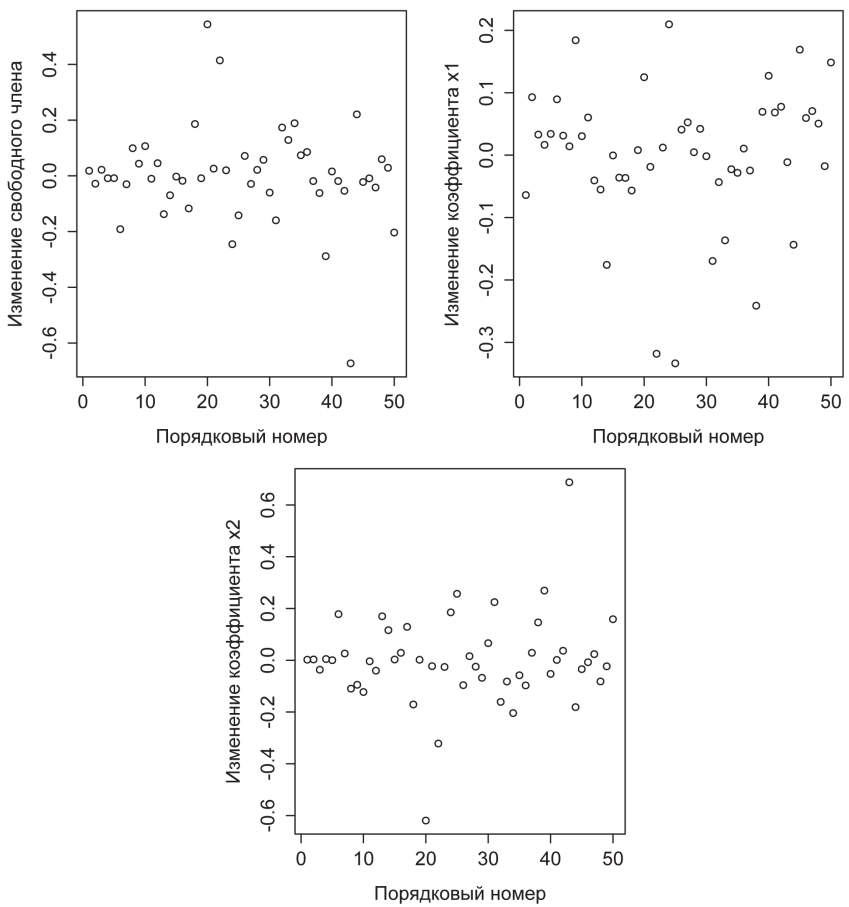


Рисунок 103

По рис. 103 видно, что в большинстве случаев значения показателя $DFBETAS_{ij}$ близки к нулю. Тем не менее несколько наблюдений демонстрируют повышенную влиятельность – например, два хорошо выделяющихся наблюдения на графике для оценок коэффициента предиктора x2.

Cook (1977; http://bit.ly/1IHQir8) предложил выражать влиятельность при помощи следующего показателя:

$$D_i = \frac{r_i^2}{k+1} \times \frac{h_i}{1-h_i},$$

где r_i – стандартизованный остаток i-го наблюдения, а h_i – соответствующий диагональный элемент матрицы влияния. Как можно видеть из приведенной формулы, первая ее часть отражает степень того, насколько i-е наблюдение не согласуется с моделью (то есть в какой мере его можно рассматривать в качестве выброса), тогда как вторая часть отражает потенциал воздействия этого наблюдения на параметры модели. Как и в случае с $DFBETA_{ij}$ и $DFBETAS_{ij}$, большое значение расстояния Кука указывает на высокую влиятельность соответствующего наблюдения. Значения расстояния Кука также принято анализировать графически.

Родственным расстоянию Кука показателем является показатель $DFFITS_i$ (Belsley et al. 1980, http://bit.ly/1G5ocX0):

$$DFFITS_i = t_i \sqrt{\frac{h_i}{1-h_i}},$$

где t_i – стьюдентизированный остаток i-го наблюдения.

Для расчета расстояний Кука в R служит функция **cooks.distance()**. Ниже приведены значения расстояний Кука для первых трех наблюдений, использованных при подгонке модели M:

```
head(cooks.distance(M), n = 3)
          1           2           3
0.001735057 0.003831049 0.003505790
```

Помимо точечных оценок коэффициентов регрессионного уравнения, в ряде случаев интерес может представлять также влияние отдельных наблюдений на стандартные ошибки этих коэффициентов, которые, по определению, отражают точность оценок. В случае с множественной регрессией, включающей несколько предикторов, речь будет идти о выяснении влияния отдельных наблюдений на определитель ковариационной матрицы параметров модели. В упомянутой выше работе Belsley et al. (1980) для измерения подобного влияния было предложено так называемое *ковариационное отношение* («*covariance ratio*»):

$$COVRATIO_i = \frac{1}{(1-h_i)\left(\frac{n-k-z-t_i^2}{n-k-1}\right)^{k+1}}.$$

Как несложно догадаться, в R для расчета ковариационных отношений служит функция **covratio**():

```
head(covratio(M), n = 3)
       1        2        3
1.183041 1.157888 1.067450
```

Влиятельными обычно считаются наблюдения, чьи значения ковариационного отношения значительно превышают 1.

Базовая функция **influence.measures**() позволяет одновременно рассчитать все перечисленные выше показатели влиятельности. Так, для нашего примера с моделью M получаем (для экономии места показана только часть таблицы с результатами):

```
influence.measures(M)

Influence measures of
  lm(formula = y ~ x1 + x2, data = df) :

    dfb.1_   dfb.x1   dfb.x2   dffit cov.r   cook.d    hat  inf
1   0.0181 -0.06395  0.00217  0.0714 1.183 1.74e-03 0.1010
2  -0.0282  0.09283  0.00311  0.1062 1.158 3.83e-03 0.0860
3   0.0217  0.03294 -0.03654 -0.1019 1.067 3.51e-03 0.0258
...
```

В первых трех столбцах таблицы с результатами расчетов приведены *DFBETAS* для свободного члена и двух коэффициентов регрессионного уравнения. Далее следуют столбцы с *DFFIT*, *COVRATIO*, расстояниями Кука и показателями потенциала воздействия. Замыкает таблицу столбец inf – он содержит звездочки * напротив наблюдений, которые *по совокупности всех показателей* следует считать влиятельными.

В заключение отметим, что в литературе встречаются разные мнения касательно того, какие значения описанных выше показателей считать достаточно большими, чтобы назвать какое-то наблюдение влиятельным. В целом каких-то оптимальных пороговых значений, подходящих для всех возможных случаев, не существует. Fox (1991; http://bit.ly/1ONpxWw) рекомендует сосредоточить усилия на визуальном исследовании распределений соответствующих показателей (например, используя графики вроде приведенных на рис. 103). Тем не менее в некоторых ситуациях использование заданных по умолчанию пороговых значений может оказаться полезным (например, при автоматизации определенных вычислений). В этой связи используются следующие распространенные значения (Fox, 1991):

- $|d_{ij}^*| \geq 1$ или $|d_{ij}^*| \geq 2$;
- $D_i > 4/(n-k-1)$;
- $DFFITS_i > 2\sqrt{(k+1)(n-k-1)}$;
- $COVRATIO_i > 1$ или $|COVRATIO_i - 1| > 3(k+1)/n$.

7.4. Модели регрессии при разных видах функции потерь

Два типа регрессионных моделей

Рассмотрим предварительно геометрическую интерпретацию двух основных типов моделей регрессии (Sokal & Rohlf, 1995; Legendre & Legendre, 1998).

Модель I типа, подробно рассмотренная в предыдущем разделе, применяется в условиях, когда объясняющая переменная X является «управляемой» (то есть известной априори) или случайная составляющая ее вариации существенно меньше ошибки Y. Помимо гипотезы о том, что переменные в эксперименте линейно связаны, для модели I типа существенны все предположения классической модели.

Концептуально модель регрессии I типа ориентирована на *оптимальный прогноз* значений отклика Y. С геометрической точки зрения, при этом выполняется поиск минимальной суммы квадратов остатков, то есть отсчет ведется строго по вертикали (см. сплошные линии на рис. 104 слева). Можно также показать, что найденная прямая $Y = f(X)$ проходит через «центр тяжести» облака точек-наблюдений, образованный средними значениями переменных $(\bar{x}, \bar{y})$.

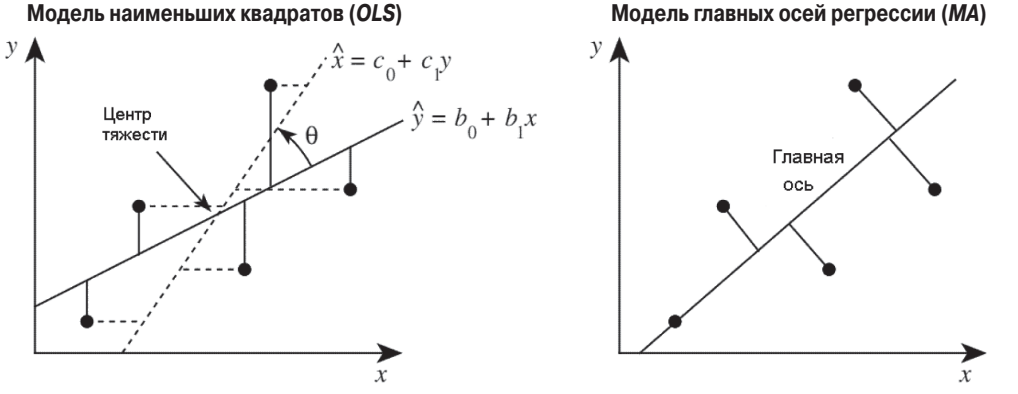


Рисунок 104

«*В рамках модели I типа (то есть когда независимая переменная X является управляемой) гипотеза о взаимозависимости случайных величин X и Y не имеет смысла*» (Legendre & Legendre, 1998, p. 499). Это подтверждается тем фактом, что для простой регрессии коэффициент детерминации R^2 просто сводится к квадрату коэффициента линейной корреляции Пирсона r_{xy}, рассмотренному в разделе 5.6. Однако в действительности через центральную точку $(\bar{x}, \bar{y})$ проходит и линия обратной регрессии $x = c_0 + c_1 y + \varepsilon$, которая будет совпадать с основной линией лишь при $r_{xy} = 1$ (на рис. 104 слева). В противном случае коэффициент корреляции равен геометрическому среднему из оценок параметров регрессии каждой переменной

на другую: $c_{(X\cdot Y)} = r_{xy}^2 / b_{(Y\cdot X)}$ или $r_{xy} = \mathrm{tg}(45^\circ - \theta/2)$, а величина угла θ между двумя линиями регрессии может быть использована для оценки нелинейных нарушений функции $Y = f(X)$.

Если случайные вариации X и Y равновелики и пропорциональны, то на оценку параметра угла наклона регрессионной прямой, найденную обычным МНК, оказывает влияние наличие ошибки измерения предиктора.

Модели *регрессии II типа* используются в случаях, когда требуется оценить параметры уравнения, описывающего функциональные отношения между парой двух случайных переменных (то есть X уже не является управляемой переменной, поскольку неконтролируемая ошибка ее измерения достаточно велика). Предполагается, что модели II типа более корректным образом оценивают *степень отношения* между двумя исходными переменными (Sokal & Rohlf, 1995; Legendre & Legendre, 1998).

При аппроксимации данных моделями II типа имеет значение направление, в котором измеряются отклонения от линии регрессии:

- в методе *главных осей* («*major axis regression*», *MA*) расстояния от эмпирических точек отсчитываются в перпендикулярном направлении к аппроксимирующей прямой, после чего квадраты этих расстояний суммируют и минимизируют (рис. 104, справа);
- в *стандартном* методе главных осей («*standard major axis*», *SMA*) находят и суммируют разности вдоль направления, которое совпадает с перпендикуляром к линии регрессии I типа;
- *ранжированный* метод главных осей («*ranged major axis*», *RMA*) осуществляет предварительное преобразование исходных переменных для достижения одинакового размаха их значений (здесь имеется некоторая неразбериха в обозначениях, поскольку метод *SMA* в некоторых источниках известен также под названием *редуцированных главных осей*, или *RMA*, от «*reduced major axis regression*»).

Коэффициенты этих регрессионных моделей рассчитываются по простым формулам. Например, для модели *SMA* имеем:

$$\hat{\beta}_{1(SMA)} = \sqrt{\frac{\sum y_i^2 - (\sum y_i)^2 / n}{\sum x_i^2 - (\sum x_i)^2 / n}} = \hat{\beta}_{1(OLS)} / r_{xy},$$

$$\hat{\beta}_{0(SMA)} = \overline{y} - \hat{\beta}_{1(SMA)} \overline{x}.$$

Предположим, что мы хотим проанализировать изменчивость видового состава донных организмов, для чего в изучаемой реке на всем ее протяжении от истоков до устья возьмем 13 гидробиологических проб. Построим модели регрессии доли численности (%) организмов из двух подсемейств – Diamesinae и Orthocladiinae (личинки хирономид, или комаров-звонцов) – в общей численности зообентоса от температуры воды t, которая обусловлена продольным градиентом реки:

```
#   Определяем исходные данные:  NDO - доля Diamesinae+Orthocladiinae
#    T  -  температура воды при взятии гидробиологической пробы
NDO <- c(88.9, 94.9, 70.8, 46.4, 31.0, 66.5, 83.6, 71.9,
         59.6, 22.5, 29.2, 6.5, 17.5)
```

```
T <- c(14.8, 14.5, 11.5, 12.6, 12, 14.5, 13.3,
       16.7, 16.9, 20.6, 21.2, 22.5, 22.4)

Ex1 <- lm(NDO ~ T); Ex0 <- lm(NDO ~ 1); summary(Ex1)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  134.779     27.464   4.908 0.000466 ***
T             -4.978      1.628  -3.057 0.010909 *
---
Residual standard error: 22.53 on 11 degrees of freedom
Multiple R-squared: 0.4593,     Adjusted R-squared: 0.4102
F-statistic: 9.346 on 1 and 11 DF,  p-value: 0.01091
```

(Здесь и далее результаты моделирования приводятся в сокращенном виде.)

Комплексный регрессионный анализ с использованием различных моделей I и II типов может быть одновременно выполнен при помощи функции **lmodel2**() из одноименного пакета:

```
library(lmodel2)

# Для RMA-модели вводим нормировку осей.
# Задаем 999 перестановок
Ex2.res <- lmodel2(NDO ~ T, range.y = "relative",
                   range.x = "relative", nperm = 999)
Ex2.res

Model II regression
n = 13   r = -0.6777465   r-square = 0.4593404
Parametric P-values: 2-tailed = 0.01091  1-tailed = 0.0054547
Angle between the two OLS regression lines = 6.086494 degrees
Regression results
  Method Intercept      Slope  Angle (degrees)  P-perm (1-tailed)
1    OLS  134.7791  -4.978119        -78.64165              0.008
2     MA  229.2404 -10.729858        -84.67554              0.008
3    SMA  173.6523  -7.345105        -82.24713                 NA
4    RMA  204.0832  -9.198041        -83.79524              0.008

Confidence intervals
  Method  2.5%-Intercept 97.5%-Intercept  2.5%-Slope 97.5%-Slope
1    OLS        74.33168        195.2265   -8.562224   -1.394014
2     MA       155.11402        675.9505  -37.930007   -6.216310
3    SMA       128.38529        246.1094  -11.757010   -4.588800
4    RMA       134.22977        500.2764  -27.233223   -4.944670
Eigenvalues: 868.0162 8.551026
H statistic used for computing C.I. of MA: 0.004425181
```

По итогам расчетов мы сразу получили объемную и ценную информацию для последующего анализа: *а*) подсчитан коэффициент корреляции и проверена его

статистическая значимость; *б*) для четырех версий моделей получены оценки параметров, углы расхождения между линиями регрессии, вычислена бутстреп-методом значимость коэффициента угла наклона b_1; *в*) приведены границы доверительных интервалов для всех оцениваемых параметров.

Теперь изобразим все четыре модели на одном графике (рис. 105, слева) и отдельно модель *SMA* с доверительными интервалами для линии регрессии (рис. 105, справа):

```
op <- par(mfrow = c(1, 2))
plot(Ex2.res, method = "OLS", conf = FALSE, centroid = TRUE, main = "Все модели",
     xlab = "T", ylab = "NDO", col = 1, pch = 22, bg = "blue")
lines(Ex2.res, "SMA", col = 2, conf = FALSE)
lines(Ex2.res, "RMA", col = 3, conf = FALSE)
lines(Ex2.res, "MA", col = 4, conf = FALSE)
legend("topright", c("OLS", "SMA", "RMA", "MA"),
       col = 1:4, lty = 1)
plot(Ex2.res, "SMA", lwd = 2, main = "SMA",
     xlab = "T", ylab = "NDO")
```

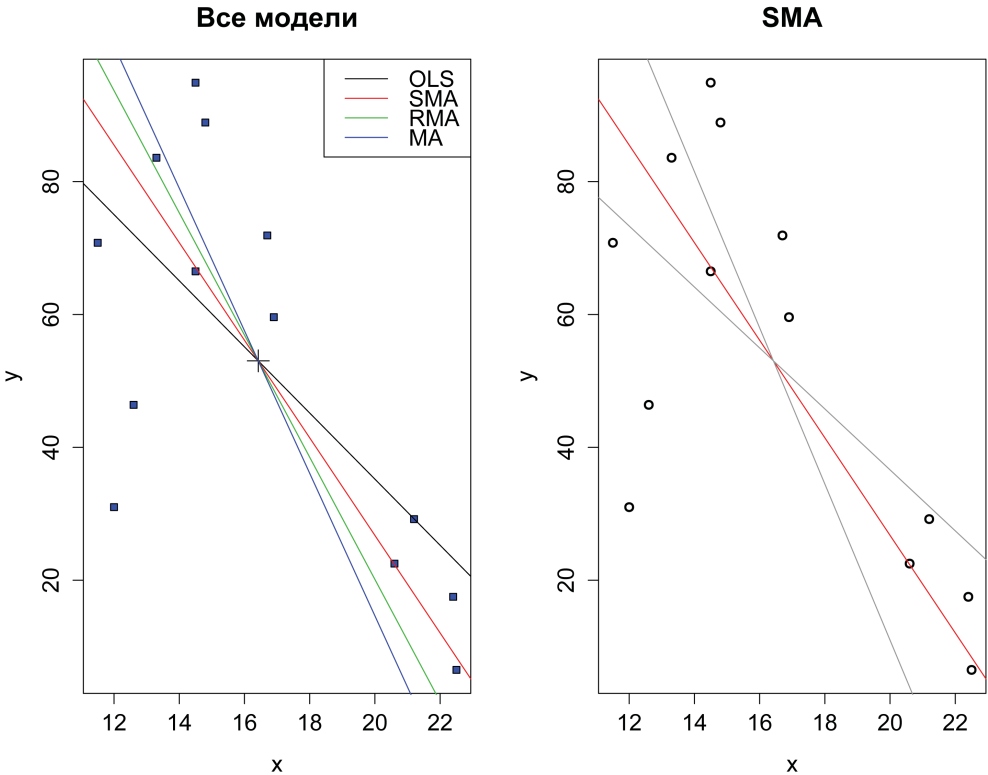


Рисунок 105

Подробные рекомендации касательно того, в каких условиях та или иная модель может оказаться предпочтительнее, обосновываются П. Лежандром в руководстве пользователя к пакету `lmodel2` (доступно по команде **`vignette`**`("mod2user")`). Мы же только обратим внимание читателя на то, что при использовании моделей II типа оценка коэффициента $\hat{\beta}_1$, угол наклона линии регрессии к оси X, а также угол θ между линиями прямой и обратной регрессии увеличиваются.

Робастные процедуры

Если распределение выборочных остатков модели регрессии $\hat{\varepsilon} = \hat{\varepsilon}_1, \hat{\varepsilon}_2, ..., \hat{\varepsilon}_n$ имеет асимметрию и «длинные хвосты», то это обычно вызывает обоснованное беспокойство при использовании метода наименьших квадратов. Действительно, функция потерь $\rho(\hat{\varepsilon}) = \hat{\varepsilon}^2$ при возведении отклонений в квадрат придает слишком большой вес значениям, аномально отклоняющимся от общей тенденции, – даже один выброс может существенно изменить оценки искомых параметров. Возможный и далеко не самый лучший метод борьбы с этим явлением – удалить наибольшие остатки, рассматривая их как выбросы, и повторить расчеты. Альтернативой является использование функции потерь $\rho(\hat{\varepsilon}) = |\hat{\varepsilon}|$, то есть минимизация суммы абсолютных разностей $|y_i - \hat{y}|$ («*least absolute deviations*», *LAD*). В русскоязычной литературе соответствующую регрессию называют медианной, а процедуру подгонки моделей – *методом наименьших модулей* (см. http://bit.ly/1CmWHG0).

Еще более предпочтительным вариантом является использование различных методов *надежной* («*robust*») и *устойчивой* («*resistant*») регрессии. Рассмотрим три из них.

Регрессия *Хубера* («*Huber regression*») пытается найти компромисс между двумя крайними случаями – *OLS* и *LAD*:

- небольшие остатки (при $|x| < c$) возводятся в квадрат: $\rho(\hat{\varepsilon}) = \hat{\varepsilon}^2$;
- для больших отклонений (при $|x| > c$) используются простые разности $\rho(\hat{\varepsilon}) = 2c|\hat{\varepsilon}| - c^2$.

Пороговое значение c («*tuning constant*») находят с использованием эмпирической зависимости $c = 1.345\sigma$ (где σ – стандартное отклонение остатков) либо путем кросс-проверки. Эта версия модели легко реализуется при помощи функции **`rlm`**`()` из пакета `MASS`:

```
# Робастная регрессия Хубера
library(MASS); summary(rlm(NDO ~ T))

Coefficients:
            Value    Std. Error t value
(Intercept) 147.2908  32.1618     4.5797
T            -5.6236   1.9070    -2.9490
Residual standard error: 17.52 on 11 degrees of freedom
```

Обратите внимание на существенное снижение стандартного отклонения остатков, по сравнению с методом OLS.

Другой известный подход основан на методе *урезанных квадратов* (*«least trimmed squares»*, *LTS*), в котором минимизируется $\sum_{i=1}^{q} \hat{\varepsilon}^2_{(i)}$, где $q < n$, а (i) – комбинации наилучших индексов x, которые подбираются так называемым «генетическим алгоритмом» (см. http://bit.ly/19wXIRk). Этот вариант модели можно реализовать при помощи функции **lmrob**() из пакета robustbase.

```
# Метод урезанных квадратов LTS
library(robustbase); summary(lmrob(NDO ~ T))

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  146.948     44.743   3.284  0.00728 **
T             -5.616      2.234  -2.513  0.02881 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Robust residual standard error: 17.95
Convergence in 18 IRWLS iterations
```

Наконец, робастная линейная регрессия *Кендалла-Тейла* (*«Kendall-Theil robust line»*, *KTR*; см. Helsel & Hirsch, 2002, http://on.doi.gov/1Ib4hVC) имеет вид $y = \hat{\beta}_0 + \hat{\beta}_1 x + \varepsilon$. Знак ˆ над символами β означает, что коэффициенты регрессии являются не параметрами некой теоретической модели, а заданными статистиками на множестве возможных значений случайных величин X и Y. Регрессия Кендалла-Тейла тесно связана с коэффициентом ранговой корреляции τ Кендалла, то есть проверка гипотезы $H_0 : \hat{\beta}_1 = 0$ эквивалентна оценке значимости тау $H_0 : \tau = 0$.

Основные шаги процедуры подбора уравнения регрессии *KTR* сводятся к следующему:

- вычисляется множество $n(n-1)/2$ коэффициентов угла наклона $b_{ij} = (y_i - y_j)/(x_i - x_j)$ для всех возможных пар точек i и j исходной выборки;
- расчетные коэффициенты модели оцениваются как медианы

  $\hat{\beta}_1 = \text{med}(b_{ij})$; $\hat{\beta}_0 = \text{med}(y) - \hat{\beta}_1 \times \text{med}(x)$;

- доверительные интервалы коэффициентов рассчитываются по интерполяционным формулам (см. прилагаемый скрипт Kendall_Theil_Regr.r, разработанный М. Вейнкауфом), а статистическая значимость модели в целом оценивается по формулам для коэффициента ранговой корреляции τ Кендалла.

```
# Расчеты характеристик регрессии Кендалла-Тейла:
source("Kendall_Theil_Regr.r")
df1 <- data.frame(T, NDO)

Kendall(df1)
                  Results
Dip           -5.56716418
Upper CI Dip   0.21153846
Lower CI Dip  -9.48214286
```

```
Intercept    141.99402985
Tau           -0.42307692
p              0.05047365
```

Интересно, что это – единственная из рассмотренных моделей, оказавшаяся статистически незначимой.

Итак, мы получили семь различных вариантов модели линейной регрессии для нашего примера:

$Y = 135 - 4.98X$ (модель I, метод LSQ);
$Y = 229 - 10.7X$ (модель II, метод MA);
$Y = 174 - 7.34X$ (модель II, метод SMA);
$Y = 204 - 9.20X$ (модель II, метод RMA);
$Y = 147.3 - 5.62X$ (регрессия Хубера);
$Y = 146.9 - 5.61X$ (метод LTS);
$Y = 146.9 - 5.57X$ (метод KTR).

Обращает на себя внимание почти полное единодушие в оценках параметров робастными методами (дискуссию о робастных оценках и о том, следует ли их называть по-русски «устойчивыми» или «надежными», можно почитать на сайте http://bit.ly/1a7QVig).

Педантичный читатель, вероятно, будет требовать от нас прямого указания на то, какой из перечисленных вариантов наиболее адекватный. Ответа у нас нет хотя бы потому, что разные модели служат разным целям исследований. Современные подходы к формальному тестированию моделей реализуются с использованием бутстрепа, кросс-проверки и имитационных методов (Шитиков, Розенберг, 2014). В целом можно сказать, что поиск моделей регрессии, которые «наилучшим образом» описывают двумерное облако экспериментальных точек, является фундаментальной проблемой анализа данных, не имеющей конечного решения.

7.5. Критерии выбора моделей оптимальной сложности

Как было отмечено в разделе 7.1, регрессионные модели в общем виде можно представить как $y = E(Y|X = x) + \varepsilon = f(x, \beta) + \varepsilon$, то есть условное математическое ожидание отклика $E(Y|X = x)$ моделируется некоторой аппроксимирующей функцией $f(x, \beta)$ с параметрами β, которая с ошибкой ε восстанавливает среднюю величину реализаций y случайной величины Y для произвольного значения x независимой переменной X. Задачей регрессионного анализа является нахождение такой аппроксимирующей функции $f(x, \beta)$, которая позволяет максимизировать определенные заданные исследователем критерии оптимальности.

Необходимо отметить, что большинство процессов в реальном мире нелинейны (более того, можно пойти еще дальше и утверждать, что линейных зависимостей в природе вообще не существует). Например, закон Ома нелинеен, если включить в него зависимость сопротивления проводника от температуры. Поэтому в учебни-

ках физики основные законы мироздания представлены экспонентами, гиперболами, параболами, показательными, степенными и полиномиальными функциями, тригонометрическими рядами Фурье, а также их различными комбинациями.

Иногда статистике поручается *верификация* теоретической модели (то есть проверка ее истинности, адекватности по отношению к новым полученным данным; англ. «*model verification*»). При этом предметная (или дескриптивная) модель жестко обусловливает модель статистическую, предписывая ей определенные свойства, включающие в себя функциональную структуру и требуемые переменные. Привлечение априорной информации о механизме изучаемых процессов часто является решающим фактором при выборе формы зависимости. Например, принято считать, что зависимость «доза–эффект» в токсикологии описывается пробит-функцией (см. примеры на сайтах http://bit.ly/1NuyCC9 и http://bit.ly/19iHBHi), а большинство аллометрических зависимостей между размерами двух структурных частей организма подчиняется степенному уравнению, предложенному Т. Гекслиеще в 1932 г. Однако в большинстве иных случаев выбор алгебраической формы функции $f(x, \beta)$ далеко не очевиден и обычно осуществляется с учетом следующих обстоятельств:

- возможность непротиворечивой интерпретации регрессионной модели на основе теоретических представлений в конкретной предметной области;
- точность аппроксимации моделью имеющихся выборочных данных;
- устойчивость регрессионной модели при экстраполяции (то есть при $x \to |\infty|$) и при наличии «выбросов»;
- минимальная сложность («экономность» модели), определяемая числом подбираемых параметров.

Подбор, или «селекция», оптимальной модели («*model selection*») из ограниченного числа моделей-претендентов является частным случаем более общей задачи определения структуры модели («*structure selection*»). Естественным является желание выбрать модель, обеспечивающую минимальную среднеквадратичную ошибку ε, которая по мере увеличения сложности функции $f(x, \beta)$, как правило, монотонно убывает. Однако включаемые в модель дополнительные члены на определенном этапе могут уже не нести полезную информацию или начинают дублировать друг друга (что происходит, например, при бесконтрольном увеличении степени полинома). При этом средняя ошибка на независимых контрольных данных сначала уменьшается, затем проходит через точку минимума и далее только возрастает. Модели, содержащие избыточное число параметров и, как следствие, генерирующие предсказания на новых данных с высокой ошибкой, называют *переобученными* («*overfitted*»), или «переусложненными». Создание переобученных моделей часто вызвано использованием «внутренних» критериев качества, основанных на среднеквадратичном отклонении остатков, таких как дисперсионное отношение Фишера F и коэффициент детерминации R^2. Эти показатели будут *всегда возрастать* при увеличении числа переменных модели, даже если включаемые в модель дополнительные предикторы не имеют тесной связи с зависимой переменной.

Выход заключается в использовании таких критериев, которые налагают «штраф» на число параметров модели k, – чем больше параметров, тем больше этот «штраф» и, как следствие, тем меньше значение соответствующего критерия. Одним из таких критериев является упомянутый ранее скорректированный коэффициент детерминации

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-k-1}.$$

К сожалению, на практике величина штрафа, налагаемого на число параметров модели в формуле R^2_{adj}, часто оказывается недостаточной. В связи с этим современные подходы к выбору хорошо интерпретируемых моделей оптимальной сложности ориентируются на применение информационных критериев, основанных на функции максимального правдоподобия и более чувствительных к увеличению числа параметров модели, а также на использование методов перекрестной проверки.

Информационный критерий Акаике («*Akaike information criterion*», AIC) в явном виде включает «штраф» за увеличение сложности модели:

$$\mathrm{AIC} = G^2 + 2k,$$

где G^2 – так называемый *девианс* («*deviance*») – показатель адекватности модели, вычисляемый с использованием функции правдоподобия (подробнее см. раздел 8.2). При нормальном законе распределения остатков

$$G^2 = -n\ln\left(\sum_{i=1}^{n}(\hat{y}_i - y_i)^2/n\right),$$

где n – количество наблюдений (Burnham & Anderson, 2002, http://bit.ly/1NrW9BE).

В случаях, когда переобучение модели особенно нежелательно, используют различные «клоны» AIC, где штраф на число параметров увеличен еще больше:

- байесовский информационный критерий: $\mathrm{BIC} = G^2 + 2\ln(n)\,k$;
- скорректированный критерий: $\mathrm{AIC}_c = \mathrm{AIC} + 2k(k+1)/(n-k-1)$.

В среде R для расчета информационных критериев используются функции **AIC**(fit), **BIC**(fit), где fit – объект соответствующей модели, а также функция **extractAIC**(), возвращающая эквивалентное число степеней свободы, использованное при расчете критерия.

Все перечисленные выше показатели качества подгонки модели («*goodness of fit*») относятся к *внутренним* критериям, то есть они оценивают модель с использованием тех же данных, по которым эта модель была построена. Однако минимизация ошибок на обучающем множестве, которое не бывает ни идеальным, ни бесконечно большим, неизбежно приводит к моделям, смещенным относительно истинной функции процесса, порождающего наблюдаемые данные. Преодолеть это смещение можно только с использованием *внешних критериев*.

Само возникновение термина «внешний критерий» обязано аналогии с термином «*внешнее дополнение*» (Бир, 1963), что в свое время служило косвенным обо-

снованием предпочтительности их применения. Сейчас стало ясно, что, поскольку теоретически строгого определения обоих этих понятий не существует, представляется несколько неуместной ссылка на теорему Геделя о неполноте (Нагель, Ньюмен, 1970, http://bit.ly/19iO3y5) как аргумент целесообразности использования «принципа внешнего дополнения». Однако если понимать под использованием внешних критериев процедуру, когда оценки параметров моделей β и суммы квадратов остатков вычисляются на несовпадающих множествах, то окажется, что внешние критерии почти всегда лучше внутренних в смысле эффективности прогноза. Они также являются незаменимыми при выборе оптимального числа параметров моделей, как эффективное средство против их переусложнения (см. приложение к книге Розенберга и др. (1994), написанное Ю. П. Юрачковским, http://bit.ly/1GEFFWZ).

Прямой путь к объективной оценке ошибки воспроизводимости предсказаний модели – это использовать для проверки (также «валидации», от англ. «*validation*») новые, независимые данные из того же источника наблюдений. Если нам нужно выбрать одну модель из числа многих, мы выбираем ту, которая оказалась лучшей при испытании в независимых условиях. Поскольку получить новые порции данных чаще всего возможности не имеется, то будет логичным разделить доступные данные случайным образом на две части – так называемые *обучающую* и *проверочную* выборки. Такой подход является определенной гарантией того, что эти две выборки независимы и одинаково распределены. Наконец, естественным будет сделать эту процедуру многократной.

Перекрестная проверка, или «*кросс-проверка*» («*cross-validation*», *CV*), является общей процедурой эмпирического оценивания моделей, построенных по прецедентам. В ходе *многократного случайного* разбиения исходной выборки на обучающую и проверочную оценка параметров модели выполняется только на обучающей выборке, тогда как оценка погрешности прогноза отклика $\hat{y}$ делается для значений из проверочной совокупности. Если исходные выборки независимы, то средняя ошибка перекрестной проверки дает *несмещенную* оценку ошибки регрессии. Это выгодно отличает ее от средней ошибки на обучающей выборке, которая при переусложнении модели может оказаться чрезмерно оптимистично заниженной.

Стандартной считается методика $r \times k$-кратной кросс-проверки («*r×k-fold cross-validation*»), когда исходная выборка случайным образом r раз разбивается на k блоков («*folds*») равной (или почти равной) длины. При реализации каждой повторности r («*replication*») один из блоков по очереди становится проверочной последовательностью, а объединение всех остальных блоков – обучающей выборкой. При этом генерируется $r \times n$ рассчитанных по модели значений отклика $\hat{y}$ (n – общее количество наблюдений), по которым оценивается ошибка кросс-проверки s_{CV}.

Частным случаем является перекрестная проверка с последовательным исключением одного наблюдения («*leave-one-out CV*»), то есть $k = n$. При этом строится n моделей по $(n - 1)$ выборочным значениям, а исключенное наблюдение каждый

раз используется для расчета ошибки прогноза. Задавать число повторностей r здесь смысла уже не имеет. Такая процедура называется *скользящим контролем* (Вапник с соавт., 1984, с. 59–65, http://bit.ly/1xqow10), хотя в последнее время этот термин стали распространять и на иные формы кросс-проверки (см., например, http://bit.ly/1NuEvze). Мы здесь возвращаемся к первоначальному значению этого понятия.

В среде R кросс-проверка модели может быть выполнена с использованием нескольких функций:

- **CVlm**(df, form.lm, m) из пакета DAAG, где df – исходная таблица с данными, form.lm – формула линейной модели, m – число блоков (сюда подставляется значение k);
- **cv.glm**(df, glmfit, K = n) из пакета boot, где df – исходная таблица с данными, glmfit – объект обобщенной линейной модели типа glm, K – число блоков (по умолчанию выполняется скользящий контроль);
- **cvLm**(lmfit, folds = **cvFolds**(n, K, R), cost) из пакета cvTools , где lmfit – объект обобщенной линейной модели lm, K – число блоков, R – число повторностей, cost – наименование одной из пяти разновидностей критериев кросс-проверки; аналогичная функция **cvLmrob**() осуществляет кросс-проверку робастных моделей – см. раздел 7.4.

Тестовыми характеристиками качества моделей, подвергаемых процедуре кросс-проверки, могут быть различные статистики, основанные на средней разности эмпирических и прогнозируемых значений y, которые, например, в случае скользящего контроля могут иметь вид:

- среднеквадратичная ошибка: $s_{CV} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$;
- среднее абсолютное отклонение: $d_{CV} = \sum_{i=1}^{n}|y_i - \hat{y}_i|/n$;
- коэффициент детерминации: $R^2_{CV} = 1 - s^2_{CV}/s^2_y$, где s^2_y – оценка дисперсии зависимой переменной.

7.6. Полиномиальные и нелинейные модели регрессии

Полиномиальная регрессия

Поскольку вычислительные аспекты общей линейной модели достаточно просты, естественным является стремление исследователей представить «криволинейные» зависимости («*curvilinear regression*») в рамках линейной регрессии.

Есть два подхода к понятию линейности регрессионной модели: линейность по переменным и линейность по параметрам. Функция $y = f(x_1, \ldots, x_m)$ *линейна по* всем независимым *переменным* лишь тогда, когда любая производная df/dx_i не включает x_j, $j = 1, \ldots, m$. К счастью, многие нелинейные модели можно представить в форме, линейной по параметрам. *Линейность по параметрам* – это свойство регрессионных моделей, позволяющее рассматривать их как линейные функции

с точки зрения оценки параметров (подробнее см. http://bit.ly/1G5ZoOD). Соответствующий процесс преобразования модели называется *линеаризацией*.

Наиболее распространенным способом представления нелинейности отклика при помощи линейной модели является использование полиномов вида $y = \beta_0 + \beta_1 x_1 + \beta_2 x^2 + ... + \beta_k x^k$. Действительно, достаточно заменить x^2 на новую переменную z_2, x^3 – на новую переменную z_3 и т. д., – и модель превратится в обычную линейную регрессионную модель, параметры которой можно легко оценить при помощи функции **lm**().

Рассмотрим в качестве примера анализ зависимости частоты (*Y*) встречаемости аллеля *Lap*94 в генах мидии *Mytilus edulis* от расстояния местонахождения популяции (в милях) до г. Саундпорт (*X*). Эти данные используются для построения полиномиальной регрессии в книге Sokal & Rolf (1995, p. 667), которая является, пожалуй, одной из лучших монографий по применению статистики в биологии. Повторим выполнение этого анализа в среде R:

```
Y <- c(0.155, 0.15, 0.165, 0.115, 0.11, 0.16, 0.17, 0.355,
     0.43, 0.305, 0.315, 0.46, 0.545, 0.47, 0.45, 0.51, 0.525)
X <- c(1, 9, 11, 17, 18.5, 25, 36, 40.5, 42, 44, 45, 51, 54,
     55.5, 57, 61, 67)
```

Выполним аппроксимацию данных кубическим полиномом с использованием функции **poly**(X, degree = 3, raw = TRUE). Параметр raw принят равным TRUE, чтобы получить натуральные значения x^k, поскольку иначе формируется ортогональный полином Чебышева:

```
lm.3 <- lm(Y ~ poly(X, 3, raw = TRUE))
summary(lm.3)

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              2.010e-01  5.467e-02   3.676  0.00279 **
poly(X, 3, raw = TRUE)1 -1.371e-02  6.760e-03  -2.028  0.06360 .
poly(X, 3, raw = TRUE)2  6.091e-04  2.211e-04   2.755  0.01637 *
poly(X, 3, raw = TRUE)3 -4.922e-06  2.080e-06  -2.366  0.03418 *
---
Residual standard error: 0.0548 on 13 degrees of freedom
Multiple R-squared: 0.9061,     Adjusted R-squared: 0.8844
F-statistic:  41.8 on 3 and 13 DF,  p-value: 6.134e-07
```

Можно предположить, что полином иной степени (то есть большей или меньшей 3) будет лучше аппроксимировать данные представленных наблюдений. Рассчитаем комплект критериев качества моделей для значений степени полинома k от 1 (простая линейная регрессия) до 4 (полином 4-й степени):

```
# Создаем список из объектов всех четырех моделей
allModel <- lapply(1:4, function(k)
                   lm(Y ~ poly(X, k, raw = TRUE)))

extract <- function(fit) {
```

```
  sigma <- summary(fit)$sigma  # среднеквадратичная ошибка
  R2.adj <- summary(fit)$adj.r.squared  # скорректиров. R2
  aic <- AIC(fit)              #  AIC-критерий
  out <- data.frame(sigma = sigma, R2.adj = R2.adj, AIC = aic)
return(out) }

result <- lapply(allModel, extract) # Список результатов

# Преобразуем список результатов в таблицу
result <- as.data.frame(matrix(unlist(result), nrow = 4, byrow = T))
colnames(result) <- c("M.sigma", "M.R2adj", "M.AIC")

#   Вычисление среднеквадратичной ошибки кросс-проверки
M.ErCV <- sapply(1:4, function(k) {
     n <- length(X); Err_S <- 0
   for(i in 1:n)    { Xcv <- X[-i]
        lm.temp <- lm( Y[-i] ~ poly(Xcv, k, raw = TRUE))
        YR <- predict(lm.temp, newdata = data.frame(Xcv = X))[i]
        Err_S <- Err_S + (Y[i] - YR)^2  }
   sqrt(Err_S/n) })

cbind(result, M.ErCV)
     M.sigma   M.R2adj     M.AIC     M.ErCV
1 0.06955197 0.8137430 -38.51702 0.07418150
2 0.06315947 0.8464073 -40.96789 0.07202873
3 0.05479741 0.8843851 -45.05640 0.06904193
4 0.05406893 0.8874387 -44.87215 0.07535719
```

Видно, что стандартное отклонение для остатков `M.sigma` и коэффициент детерминации `M.R2adj` изменяются монотонно (то есть убывают и возрастают соответственно), тогда как информационный критерий `M.AIC` и стандартная ошибка кросс-проверки `M.ErCV` достигают минимума при `k = 3` (рис. 106).

```
  plot(X, Y, type = "p", pch = 22, bg = "yellow",
       xlab = "Расстояние", ylab = "Частота аллеля")
sapply(1:4, function(k) points(X, predict(lm(Y ~ poly(X, k,
       raw = TRUE))), type = "l", col = k, lwd = 2 ))
legend("topleft", c("k = 1", "k = 2", "k = 3", "k = 4"),
       col = 1:4, lwd = 2)
```

Внимательный читатель заметит, что полученные нами результаты несколько отличаются от представленных в книге Sokal & Rolf (1995). Дело в том, что мы не стали предварительно выполнять угловое преобразование Фишера для долей $\phi = 2\arcsin(\sqrt{p})$, поскольку ряд авторитетных современных специалистов в области статистики относятся к нему резко негативно: «*При малых объемах выборок и малых значениях долей его применение приводит к ложной чувствительности (мощности) соответствующих критериев и ложным выводам о значимости различий, которых в реальности нет*» (Хромов-Борисов и др., 2004, с. 65, http://bit.

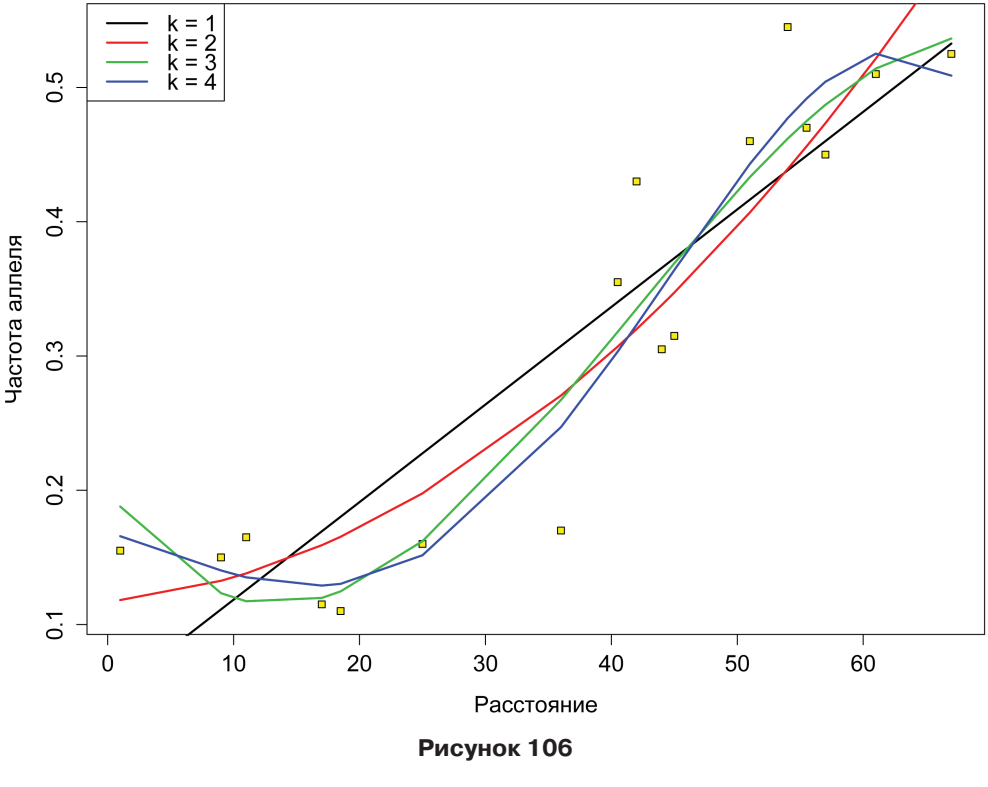


Рисунок 106

ly/1yqmNUN). Впрочем, в данном случае отличия незначительны, особенно при графической интерпретации кривых.

Нелинейная регрессия

В ряде случаев приведение нелинейной функции к линейной форме невозможно, и тогда подгонку значений коэффициентов модели следует выполнять стандартными методами нелинейного программирования – методом Ньютона-Рафсона, симплекс-методом Нелдера-Мидда и прочими. Для этого в среде R чаще всего используется функция `nls()`. Приведем два характерных примера.

Ставится задача оценить чувствительность водных беспозвоночных к уровню минерализации воды на примере речных экосистем Приэльтонья. Область варьирования фактора солености S разбивается на последовательность подинтервалов, а в качестве значений отклика y подсчитывается доля P_k проб из k-го диапазона x, в которых встретился тот или иной анализируемый вид.

Будем использовать логистическую модель нелинейной регрессии, основанную на функции $P = \theta_1 / \{1 + e^{-(\theta_2 + \theta_3 S)}\}$, которая зависит от трех параметров $\theta = (\theta_1, \theta_2, \theta_3)$ и графически выглядит как симметричная S-образная кривая. Эта модель основана на предположении, что скорость изменения отклика dy/dx имеет нелиней-

ный характер и достигает своего максимума в точке перегиба кривой. Параметр θ_1 определяет расстояние между верхней и нижней асимптотами, а константы θ_2 и θ_3 регулируют, соответственно, наклон и изгиб моделируемой кривой.

Выполним подгонку параметров модели для вида *Tanytarsus kharaensis*:

```
S1 <- c(3, 7.5, 12.5, 17.5, 22.5, 27.5)
p1 <- c(50, 23, 45, 25, 2, 9)

log.ss1 <- nls(p1 ~ SSlogis(S1, phi1, phi2, phi3))

summary(log.ss1)
Formula: p1 ~ SSlogis(S1, phi1, phi2, phi3)
Parameters:
     Estimate Std. Error t value Pr(>|t|)
phi1  39.3387     7.5842   5.187   0.0139 *
phi2  17.9981     4.7656   3.777   0.0325 *
phi3  -0.8797     8.2858  -0.106   0.9221
---
Residual standard error: 12.88 on 3 degrees of freedom
Number of iterations to convergence: 1
Achieved convergence tolerance: 9.036e-06
```

Протокол вывода результатов работы функции **nls**() в части оценок параметров аналогичен таковому у функции **lm**(). Исходя из полученных результатов, модель можно записать в следующем виде:

$$P = 39.3/\{1 + \exp(0.88S - 18)\}.$$

«Ложкой дегтя» здесь является статистическая незначимость оценки коэффициента θ_3 модели, что определяется малым числом степеней свободы при выбранном способе представления данных (при рассмотрении обобщенных линейных моделей в разделе 8.3 мы покажем, как те же выборочные наблюдения можно использовать существенно эффективнее).

Рассчитаем коэффициент детерминации R^2 и доверительные интервалы нелинейной модели регрессии методом аппроксимации (пример заимствован с сайта http://bit.ly/1D3H1Ji):

```
Rsquared <- 1 - var(residuals(log.ss1))/var(p1)

# Рассчитанные по модели значения отклика
x <- 0:max(S1)
pr1 = predict(log.ss1, data.frame(S1 = x))
## заметим, что независимая переменная в SSlogis имеет имя "S1",
## и поэтому мы изменяем имя столбца в таблице предикторных значений
### вычисляем ошибку регрессии линейной аппроксимацией

se.fit <- sqrt(apply(attr(pr1, "gradient"), 1,
            function(mat) sum(vcov(log.ss1)*outer(mat, mat))))
PCI <- pr1 + outer(se.fit, qnorm(c(.5, .025,.975)))
```

Ошибка регрессии в области перегиба кривой также чрезвычайно высока, но в данном случае нам важнее методическая составляющая примера.

Любой биологический вид в зависимости от уровня воздействующего фактора может находиться в трех фазах: благоприятного режима, переходного состояния и угнетения. Поэтому после оценки параметров θ регрессионной модели выделим три критические точки пределов экологической толерантности, то есть расчетные величины солености при некоторых значениях $\pi\theta_1$. Пусть минерализация в точках, где $\pi = 0.05$ и $\pi = 0.95$, соответствует минимальной и максимальной статистическим границам встречаемости (Rutherford & Kefford, 2005, http://bit.ly/1bW3cqE), а при $\pi = 0.5$ – пороговой величине солености, за которой наступает резкое снижение популяционной устойчивости. Рассчитаем эти критические точки в R:

```
# Критические точки отклика:
a <- coef(log.ss1)[1]*c(0.05, 0.5, 0.95)
[1]  1.966937 19.669366 37.371795
# Критические точки солености:
plx <- approx(pr1, x, xout = a)$y
[1] 20.70461 17.99787 15.29205
```

Обратите внимание на функцию **approx**(), выполняющую линейную аппроксимацию произвольных таблиц, которую мы использовали, чтобы найти значение независимой переменной по заданной величине отклика.

Отобразим все вычисленные компоненты на графике (рис. 107):

```
matplot(x, PCI,type = "l", xlab = "Минерализация, г/л",
        ylab = "Доля встречаемости в пробе, %",
        lty = c(1, 3, 3), lwd = c(2, 1, 1), ylim = c(0, 60))
matpoints(S1, p1, pch = 20)
text(5, 5, bquote(R^2==.(round(Rsquared, 3))))
text(plx[3], 0, expression('S'[0.95]))
text(plx[2], 0, expression('S'[0.50]))
text(plx[1], 0, expression('S'[0.05]))
sapply(1:3, function(i)
         lines(c(0, plx[i], plx[i]), c(a[i], a[i], 0), lty = 2))
```

На рис. 107 видно, что исследуемый вид имеет отчетливо выраженное пороговое значение солености $S_{0.5} = \theta_2 = 18$ г/л и достаточно узкий диапазон от $S_{0.95} = 15.3$ г/л до $S_{0.05} = 20.7$ г/л переходной фазы снижения толерантности, разделяющий зону максимальной встречаемости (слева) и фазу «угнетенного состояния» (справа).

Рассмотрим второй пример, иллюстрирующий одно статистическое заблуждение, характерное для продукционной гидробиологии второй половины XX века и частично заскочившее в XXI век (Зотин, Зотин, 1999, с. 85–88, http://bit.ly/1P7TYqL). Предположим, что исследователь из некоторых теоретических соображений желает оценить параметры модели, описывающей степенную зависимость следующего вида:

$$Y_i = aX_i^k + \varepsilon_i. \qquad (1)$$

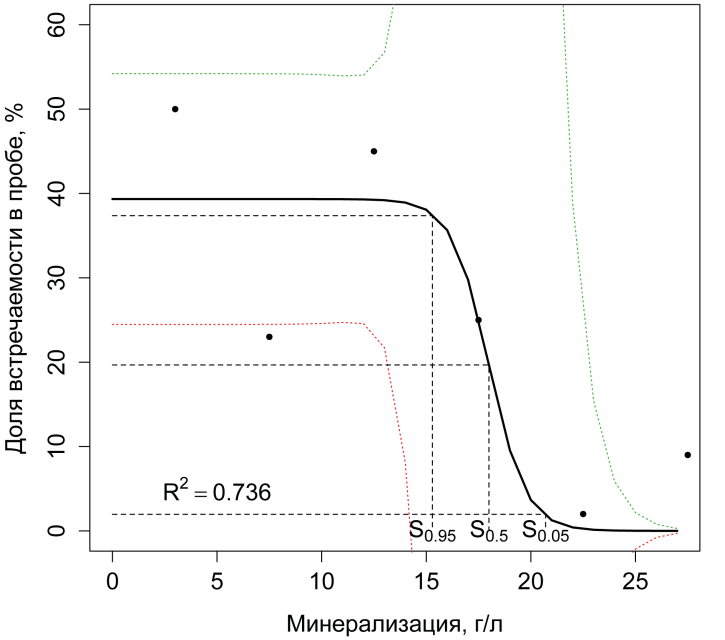


Рисунок 107

Поскольку оценить параметры нелинейной модели трудно, то А. А. Умновым (Журнал общей биологии. 1976. Т. 37. № 1. С. 71–86) предлагается выполнить логарифмирование данных, применить МНК и рассчитать коэффициенты линейной модели:

$$\lg Y_i = \lg a + k \lg X_i + \varepsilon_i. \qquad (2)$$

Являются ли уравнения (1) и (2) математически идентичными и можно ли коэффициенты уравнения (2) использовать для описания степенной зависимости (1)? Ответ на этот вопрос найдем, проанализировав пример из статьи А. А. Умнова (1976), посвященный моделированию потребления кислорода (Q, мм^3 O_2/час×экз.) в зависимости от массы тела пескожила *Arenicola marina* (W, г). Построим сначала модель степенной регрессии:

```
W <- c(2.37, 2.82, 3.12, 3.21, 3.22, 3.24, 3.38, 3.6, 4, 4.65,
  4.91, 4.93, 5.72, 5.95, 7.65, 8.26, 8.5, 10.62, 14.6, 18.76)
Q <- c(105.33, 192.63, 177.13, 221.84, 95.9, 107, 202.5,
  105.4, 182.38, 198.2, 167.45, 195, 108.8, 233.13, 190.17,
  254.35, 191.88, 416.87, 472.75, 538.34)

m.ppo <- nls(Q ~ a*W^k, start = list(a = 55, k = 0.8))

summary(m.ppo)
```

```
Formula: Q ~ a * W^k
Parameters:
  Estimate Std. Error t value Pr(>|t|)
a 53.36171   10.59057   5.039 8.54e-05 ***
k  0.78470    0.08612   9.112 3.66e-08 ***
---
Residual standard error: 56.24 on 18 degrees of freedom
Number of iterations to convergence: 4
Achieved convergence tolerance: 1.858e-06
```

Теперь найдем коэффициенты линейной модели после логарифмирования переменных:

```
m.lpo <- lm(log(Q) ~ I(log(W)))

summary(m.lpo)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.1550     0.2151  19.320 1.75e-13 ***
I(log(W))     0.6725     0.1237   5.437 3.64e-05 ***
---
Residual standard error: 0.308 on 18 degrees of freedom
Multiple R-squared: 0.6216,     Adjusted R-squared: 0.6006
F-statistic: 29.57 on 1 and 18 DF,  p-value: 3.638e-05

exp(coef(m.lpo)[1])
 (Intercept)
   63.74962
```

Многие биологи полагают, что величина коэффициента $k = 0.75$ одинакова для всех животных – от простейших до млекопитающих, – и называют ее универсальной константой Хеммингсена. Как видим, коэффициенты полученных моделей существенно различаются:

- $Q = 53.36W^{0.785}$ – истинно нелинейная модель (1);
- $Q = 63.75W^{0.672}$ – нелинейная модель, полученная в результате линеаризации (2).

Почему так произошло? Согласно известным положениям регрессионного анализа (Дрейпер, Смит, 1987), нелинейная функция может быть приведена к линейной форме только в том случае, если ошибка обоих уравнений регрессии остается аддитивной, то есть зависимая переменная должна оставаться *суммой* своего математического ожидания и ошибки. Однако при обратном преобразовании логлинейного уравнения (2) в уравнение (1) имеет место *мультипликативное* вхождение ошибок ε_i в нелинейное уравнение отклика: $Y_i = aX_i^k \cdot \varepsilon_i$, что далеко не идентично стандартной регрессионной модели с аддитивными ошибками $Y_i = aX_i^k + \varepsilon_i$. Иными словами, выражения (1) и (2) представляют собой два принципиально разных уравнения регрессии, и поэтому использование для модели $Y = \alpha X^k$ МНК-оценок a и k из уравнения $\lg Y_i = \lg a + k\lg X_i + \varepsilon_i$ неизбежно при-

ведет к искаженным результатам. Это обстоятельство легко просматривается на следующем графике (рис. 108):

```
plot(W, Q, type = "p", pch = 22, bg = "yellow",
     xlab = "Масса тела, г", ylab = "Потребление кислорода")
lines(W, predict(m.ppo), lwd = 2)
lines(W, exp(predict(m.lpo)), col = 6, lwd = 2)
legend("topleft", c("Нелинейная", "Линеаризация"),
       col = c(1, 6), lwd = 2)
```

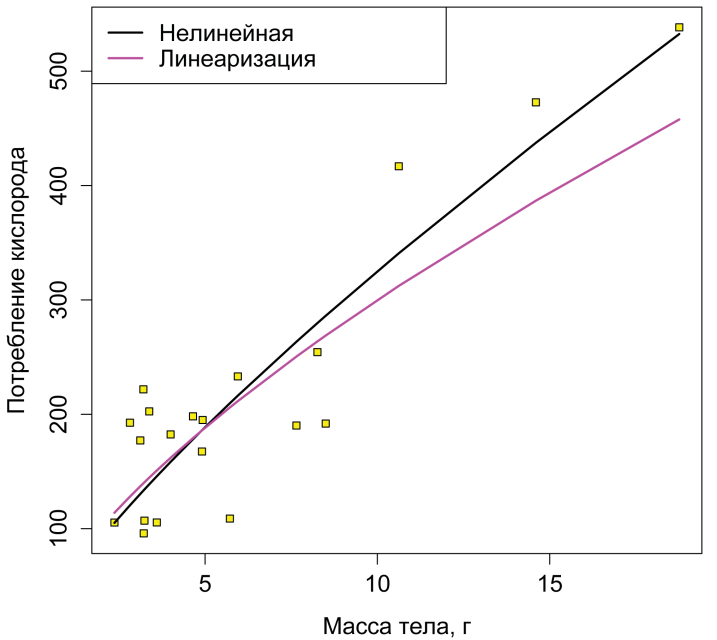


Рисунок 108

Очевидно, что модель, построенная с использованием линеаризации, некорректно описывает исследуемую зависимость, особенно при большой массе тела. В частности, стандартное отклонение остатков у линейной модели с логарифмированными переменными больше такового у «правильной» нелинейной модели (56.24):

```
sqrt(sum((Q - exp(predict(m.lpo)))^2)/(n-2))
[1] 60.85349
```

В многочисленных публикациях последних десятилетий в области продукции водоемов приводятся таблицы значений коэффициентов *a* и *k* для расчета удельных энергозатрат на обмен для различных групп организмов, однако к этим результатам следует относиться весьма критично, поскольку за давностью лет осталось непонятным, для каких моделей они были рассчитаны.

7.7. Модель множественной регрессии и выбор ее спецификации

Естественным обобщением парной регрессии (см. раздел 7.2) является множественная регрессионная модель («*multiple regression model*»), рассматривающая влияние нескольких ($m > 1$) предикторов на зависимую переменную Y:

$$E[Y|x_1, x_2, ..., x_m] = f(x_1, x_2, ..., x_m),$$

где $f(...)$ – произвольная функция m переменных.

Первым естественным приближением функции $f(...)$ является линейная зависимость (если, конечно, характер наблюдаемых данных не противоречит этому), что приводит к модели следующего общего вида:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_m X_m + \varepsilon.$$

Каждый регрессионный коэффициент $\beta_j, j = 1, 2, ..., m$, численно отражает степень влияния предиктора X_j на условное математическое ожидание $E(Y|x_1, x_2, ..., x_m)$ отклика Y, если мысленно принять, что все остальные объясняющие переменные модели остаются постоянными.

Классическая (то есть отвечающая условиям теоремы Гаусса-Маркова) модель множественной регрессии является частным случаем общей линейной модели (см. раздел 6.2, посвященный дисперсионному анализу). В связи с этим в отношении случайных ошибок ε, или остатков, делаются те же предположения, что и для простой регрессии:

- статистическая независимость наблюдений;
- гомоскедастичность дисперсии во всем диапазоне значений предикторов;
- нормальное распределение остатков с математическим ожиданием, равным 0: $\varepsilon \sim N(0, \sigma)$.

Как и в случае парной регрессии, анализ начинается с расчета величин $\hat{\beta}_j$ – выборочных оценок коэффициентов модели $\beta_j, j = 1, 2, ..., m$. Далее выполняется проверка статистических гипотез, позволяющая сделать вывод о надежности найденных точечных оценок коэффициентов, интерпретируемости найденного эмпирического уравнения регрессии и степени его соответствия результатам наблюдений.

Для проверки гипотезы H_0: $\hat{\beta}_j = 0$ используется, как правило, статистика $t = \hat{\beta}_j / S_{\hat{\beta}_j}$, где $S_{\hat{\beta}_j}$ – стандартная ошибка соответствующего коэффициента регрессии. Поскольку при справедливой нулевой гипотезе H_0 имеет место распределение Стьюдента, нетрудно найти соответствующее ему p-значение. Наряду с проверкой статистической значимости, как правило, выполняется также оценка доверительных интервалов β_j.

Пусть $\hat{\varepsilon} = (y_i - \hat{y}_i)$, $i = 1, ..., n$ – отклонения выборочных величин y_i зависимой переменной от значений $\hat{y}_i$, получаемых по уравнению регрессии. Тогда для оценки качества полученной модели можно использовать описанные ранее критерии, с той лишь разницей, что сейчас при их расчете учитывается число предикторов:

- стандартное отклонение остатков $RSE = \sqrt{(\sum \hat{\varepsilon}_i^2) / (n-m-1)}$;
- коэффициент детерминации $R^2 = 1 - \sum \hat{\varepsilon}_i^2 / \sum (y_i - \bar{y})^2$;
- скорректированный коэффициент детерминации $R^2_{adj} = 1 - (1-R^2)\frac{n-1}{n-m-1}$, который налагает «штраф» на добавление новых параметров в регрессионную модель;
- *F*-статистика $F = \frac{R^2}{1-R^2} \cdot \frac{n-m-1}{m}$, *p*-значение для оценки статистической значимости которой получают на основе распределения Фишера;
- информационный критерий Акаике $AIC = n \ln(\sum \hat{\varepsilon}_i^2 / n) + 2m$ и некоторые его разновидности – байесовский информационный критерий (BIC), критерий Шварца и др. (см. раздел 7.5).

В разделе 6.1 были рассмотрены основные пункты протокола разведочного анализа данных. Многие из этих статистических процедур обеспечивают построение хорошо интерпретируемых моделей:

- анализ мультиколлинеарности предикторов;
- отбор информативных предикторов;
- регрессионная диагностика и оценка статистической значимости параметров;
- анализ степени нелинейной связи между переменными.

Рассмотрим пример с изучением потребности во сне у животных, использованный ранее в разделах 4.4 (таблица данных с заполненными пропусками была сохранена нами для дальнейшего использования) и 6.1. Там была проанализирована корреляционная матрица и показана высокая мультиколлинеарность предикторов, оцененная на основе значений фактора инфляции дисперсии *VIF*. Чем сильнее мультиколлинеарность предикторов, тем менее надежной является оценка распределения суммы объясненной вариации по отдельным предикторам с помощью метода наименьших квадратов.

Полная модель и обоснование необходимости ее оптимизации

Рассчитаем выборочные коэффициенты регрессионной модели, содержащей все имющиеся предикторы:

```
load(file = "sleep_imp.Rdata")
M <- lm(Sleep ~ BodyWgt + BrainWgt + Span + Gest +
        Pred + Exp + Danger, data = sleep_imp3)
summary(M)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 15.717666   1.147964  13.692  < 2e-16 ***
BodyWgt     -0.002149   0.001644  -1.307  0.19665
BrainWgt     0.002105   0.001802   1.169  0.24768
```

```
Span        -0.073393   0.038519  -1.905  0.06206 .
Gest        -0.004159   0.005916  -0.703  0.48507
Pred         2.539990   0.815866   3.113  0.00296 **
Exp          0.232926   0.581090   0.401  0.69012
Danger      -4.269622   1.046721  -4.079  0.00015 ***
---
Residual standard error: 3.283 on 54 degrees of freedom
Multiple R-squared: 0.5707,     Adjusted R-squared: 0.515
F-statistic: 10.25 on 7 and 54 DF,  p-value: 4.272e-08

AIC(M)
[1] 332.7936
```

В данном случае мультиколлинеарность приводит к тому, что оценки большинства коэффициентов полной линейной модели регрессии оказываются статистически незначимыми по *t*-критерию, хотя *F*-критерий свидетельствует о значимости всей модели в целом. Считается, что наиболее эффективный путь устранения мультиколлинеарности – исключение из регрессионной модели незначимых коэффициентов, или, выражаясь точнее, отбор *информативного комплекса* из q переменных ($q < m$).

Причины, по которым стоит проводить селекцию «лучшего подмножества» предикторов, Faraway (2006) видит в следующем:

1. Принцип бритвы Оккама утверждает, что из нескольких вероятных объяснений явления лучшим является самое простое. Компактная модель, из которой удалены избыточные предикторы, лучше объясняет имеющиеся данные.
2. Ненужные предикторы добавляют шум к оценке значимости других факторов, которыми мы интересуемся. Иначе (часто ограниченные) степени свободы будут тратиться впустую.
3. При наличии коллинеарности некоторые переменные будут «пытаться сделать одну и ту же работу», то есть объяснить вариацию значений зависимой переменной.
4. Если модель используется для прогнозирования, то можно сэкономить время и/или деньги, не измеряя избыточных переменных.

Модель с настроенными параметрами, основанная на q предикторах из m, которые обеспечивают минимум некоторого критерия качества $L(\beta, x, q)$, называется *моделью оптимальной структуры*. Поскольку глобальный оптимум такого показателя при больших m достичь невозможно, представляется разумным получить для последующего содержательного анализа некоторый набор субоптимальных моделей-претендентов.

Как отмечалось ранее (см. раздел 7.5), использование в качестве критериев качества регрессионных моделей коэффициента детерминации R^2, *F*-критерия и других статистик, основанных на стандартном отклонении остатков, не всегда бывает достаточным. Поиск наилучшей модели множественной регрессии на их основе является, как правило, «*жадным*» («*greedy*») процессом, то есть предпо-

чтение будет всегда отдаваться модели с наибольшим числом параметров: $q \to m$. Более взвешенный подход реализуется с использованием информационных критериев (например, AIC), налагающих «штраф» на добавление новых параметров.

Таким образом, когда говорят об оптимальных моделях регрессии, то имеют в виду одну или несколько компактных моделей, обеспечивающих субэкстремальные значения выбранного критерия качества (например, минимум AIC или максимум R^2_{adj}). Существует целый арсенал методов, реализующих процедуры выбора моделей с оптимальной структурой:

- последовательные, или пошаговые, методы;
- полный перебор всех возможных сочетаний предикторов;
- использование специальных методов оценивания, когда увеличение числа параметров модели сопровождается наложением штрафа на критерий ее качества (гребневая регрессия, лассо-алгоритм, метод наименьших углов);
- алгоритмы эволюционного поиска: метод модельной закалки («*simulated annealing*»), генетический алгоритм («*genetic algorithm*»), случайный поиск с адаптацией (СПА; см. Лбов, 1981 на сайте http://bit.ly/1GU8fUh) и др.

Пошаговые алгоритмы селекции переменных

Шаговый регрессионный анализ (Дрейпер, Смит, 1987) представлен процедурами трех типов:

- «пошаговое включение» («*stepwise forward selection*») – на первом шаге из всех m предикторов выбирается наилучший по некоторому заданному критерию; на втором шаге выбирается предиктор, который дает оптимальное решение в сочетании с первым предиктором, и т. д. Алгоритм останавливается, когда критерий достигает экстремума и в модель включено q переменных;
- «пошаговое исключение» («*stepwise backward selection*») – на первом шаге перебираются все комбинации из m переменных и исключается наименее информативный предиктор с точки зрения заданного критерия; эти шаги повторяются, пока критерий не достигнет экстремума или не будут выполнены какие-то иные условия;
- комбинированный алгоритм, в котором этапы включения и исключения предикторов сменяют друга.

Выполним выбор оптимальной модели по критерию Акаике, используя комбинированный подход:

```
Mstep <- step(M, direction = "both")
summary(Mstep)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 15.54848    1.09002  14.264  < 2e-16 ***
Span        -0.06371    0.02571  -2.478  0.01613 *
Pred         2.54554    0.77185   3.298  0.00167 **
Danger      -4.22022    0.78506  -5.376 1.42e-06 ***
```

```
---
Residual standard error: 3.245 on 58 degrees of freedom
Multiple R-squared: 0.5494,     Adjusted R-squared: 0.5261
F-statistic: 23.57 on 3 and 58 DF,  p-value: 4.21e-10

AIC(Mstep)
[1] 327.7868
```

К такой же модели можно было бы прийти «вручную», отбрасывая из полной модели `M` поочередно предикторы с наибольшими *p*-значениями. Отметим, что наряду с уменьшением AIC значение R^2_{adj} возросло, несмотря на некоторый рост ошибки модели *RSE*.

Некоторые авторы утверждают (Blanchet et al., 2008, http://bit.ly/1FaFDSh), что стандартный метод пошагового включения переменных в модель недостаточно эффективен, поскольку переоценивает долю объясненной дисперсии и приводит к повышенному риску ошибки I рода. Один из вариантов решения этой проблемы реализован в функции **`forward.sel`**() из пакета `packfor` («*forward selection with permutation*»; пакет доступен на сайте http://bit.ly/1GcoHQJ). Реализованный в ней метод селекции последовательно включает в модель предиктор, имеющий наибольший частный коэффициент корреляции Пирсона с откликом r_{yx}. Чтобы предотвратить «перенасыщение» модели, включение новой переменной выполняется, только если: *а*) статистическая значимость r_{yx} по рандомизационному тесту находится на заданном уровне α или *б*) r_{yx} превышает некоторый заданный порог.

```
library(packfor)
forward.sel(Y = sleep_imp3$Sleep, X = sleep_imp3[, -3:5], alpha = 0.20)
Procedure stopped (alpha criteria): pvalue for variable 5 is 0.314000 (superior to
0.200000)
  variables order         R2     R2Cum  AdjR2Cum         F  pval
1       Exp     6 0.38759370 0.3875937 0.3773869 37.974172 0.001
2      Span     3 0.03672481 0.4243185 0.4048039  3.763824 0.065
3    Danger     7 0.05144769 0.4757662 0.4486507  5.692052 0.021
4      Pred     5 0.07455370 0.5503199 0.5187634  9.450187 0.002
```

В результате была отобрана модель из 4 переменных, три из которых (`Exp`, `Pred` и `Danger`) представляют собой сильно коррелирующие переменные (см. раздел 6.1).

Построение «всех возможных моделей»

Действительно ли мы нашли оптимальную модель, выполнив описанный выше анализ? Для сравнения рассмотрим алгоритм перебора всех возможных предикторов, который вполне реализуем при числе переменных менее 8. Подходящая функция, найденная на сайте http://bit.ly/1FaFLB5, не слишком хороша, поскольку требует перестройки исходной таблицы и фиксированного списка имен переменных (`y`, `x1`, `x2` и т. д.). Однако она предоставляет читателям простор для самостоятельного творчества и содержит многие полезные для обучения фрагменты кода:

```
all.possible.regressions <- function(dat, k){
    n <- nrow(dat)
    regressors <- paste("x", 1:k, sep = "")
    lst <- rep(list(c(T, F)), k)
    regMat <- expand.grid(lst);
    names(regMat) <- regressors
    formular <- apply(regMat, 1, function(x)
            as.character(paste(c("y ~ 1", regressors[x]), collapse = "+")))
    allModelsList <- apply(regMat, 1, function(x)
            as.formula(paste(c("y ~ 1", regressors[x]), collapse = "+")))
    allModelsResults <- lapply(allModelsList,
            function(x, data) lm(x, data = data), data = dat)
    n.models <- length(allModelsResults)
    extract <- function(fit) {
        df.sse <- fit$df.residual
        p <- n - df.sse -1
        sigma <- summary(fit)$sigma
        MSE <- sigma^2
        R2 <- summary(fit)$r.squared
        R2.adj <- summary(fit)$adj.r.squared
        aic <- AIC(fit)
        sse <- MSE*df.sse
        bic <- n*log(sse) + log(n)*(p+2)
        out <- data.frame(df.sse = df.sse, p = p, SSE = sse, MSE = MSE,
            R2 = R2, R2.adj = R2.adj, AIC = aic, BIC = bic)
        return(out)
    }
    result <- lapply(allModelsResults, extract)
    result <- as.data.frame(matrix(unlist(result), nrow = n.models, byrow = T))
    result <- cbind(formular, result)
rownames(result) <- NULL
colnames(result) <- c("model", "df", "p", "SSE", "MSE", "R2",
"R2.adj", "AIC", "BIC")
    return(result)
}
```

Перестраиваем таблицу и переименовываем столбцы с исходными переменными:

```
ind.yx <- c(5, 1:2, 6:10)
names.xy <- c("y", paste("x", 1:7, sep = ""))
yx <- sleep_imp3[,ind.yx]
data.frame(Old = colnames(yx), New = names.xy, Index = ind.yx)
       Old New Index
1    Sleep   y     5
2  BodyWgt  x1     1
3 BrainWgt  x2     2
4     Span  x3     6
5     Gest  x4     7
6     Pred  x5     8
```

```
7      Exp  x6     9
8   Danger  x7    10
```

```
colnames(yx) <- names.xy
```

Получаем список из всех возможных моделей, которых насчитывается 128:

```
all.mod <- all.possible.regressions(dat = yx, k = 7)
# Вывод в буфер обмена:
write.table(all.mod, "clipboard", sep = "\t"); head(all.mod[order(all.mod$AIC), ], 10)
                model  df p      SSE      MSE        R2    R2.adj      AIC      BIC
     y ~ 1+x3+x5+x7  58 3 610.8497 10.53189 0.5494240 0.5261183 327.7868 418.3564
   y ~ 1+x1+x3+x5+x7  57 4 598.3928 10.49812 0.5586124 0.5276379 328.5094 421.2061
   y ~ 1+x3+x4+x5+x7  57 4 600.5966 10.53678 0.5569869 0.5258982 328.7373 421.4341
      y ~ 1+x4+x5+x7  58 3 622.9865 10.74115 0.5404716 0.5167029 329.0066 419.5762
   y ~ 1+x2+x3+x5+x7  57 4 605.4408 10.62177 0.5534137 0.5220743 329.2354 421.9321
y ~ 1+x1+x2+x3+x5+x7  56 5 587.6900 10.49446 0.5665071 0.5278023 329.3904 424.2143
   y ~ 1+x3+x5+x6+x7  57 4 609.6351 10.69535 0.5503199 0.5187634 329.6634 422.3602
y ~ 1+x1+x3+x4+x5+x7  56 5 596.8407 10.65787 0.5597573 0.5204499 330.3484 425.1723
y ~ 1+x1+x3+x5+x6+x7  56 5 598.3770 10.68530 0.5586241 0.5192155 330.5078 425.3316
y ~ 1+x2+x3+x4+x5+x7  56 5 600.5008 10.72323 0.5570575 0.5175091 330.7274 425.5513
```

Отметим, что комбинированный шаговый метод позволил нам выбрать оптимальную модель (согласно AIC) при имеющемся наборе предикторов. Характеристики модели, полученной с использованием функции **forward.sel**(), представлены в седьмой строке приведенного списка. Однако если использовать для сортировки моделей коэффициент детерминации R^2_{adj}, то список выглядел бы несколько иначе.

Полный перебор всех возможных комбинаций предикторов можно также осуществить с использованием функции **regsubsets**() из пакета leaps:

```
library(leaps)
leaps <- regsubsets(Sleep ~ BodyWgt + BrainWgt + Span + Gest +
                    Pred + Exp + Danger, data = sleep_imp3, nbest = 7)
summary(leaps)
```

Если число переменных велико, то функция **regsubsets**() автоматически начнет использовать различные оптимизированные алгоритмы последовательного перебора для формирования подмножеств предикторов.

Далее мы можем графически (рис. 109) изучить степень участия каждого предиктора в подогнанных моделях, выполнив сортировку по интересующему нас критерию – см. аргумент scale = **c**("bic", "Cp", "adjr2", "r2"):

```
plot(leaps, scale = "adjr2")
```

Пошаговое включение предикторов в сочетании с перекрестной проверкой

Как было отмечено ранее (раздел 7.5), выбор оптимальной модели может быть основан на использовании внешних критериев качества, оцениваемых в результате перекрестной проверки. Ничто не запрещает нам использовать подобные кри-

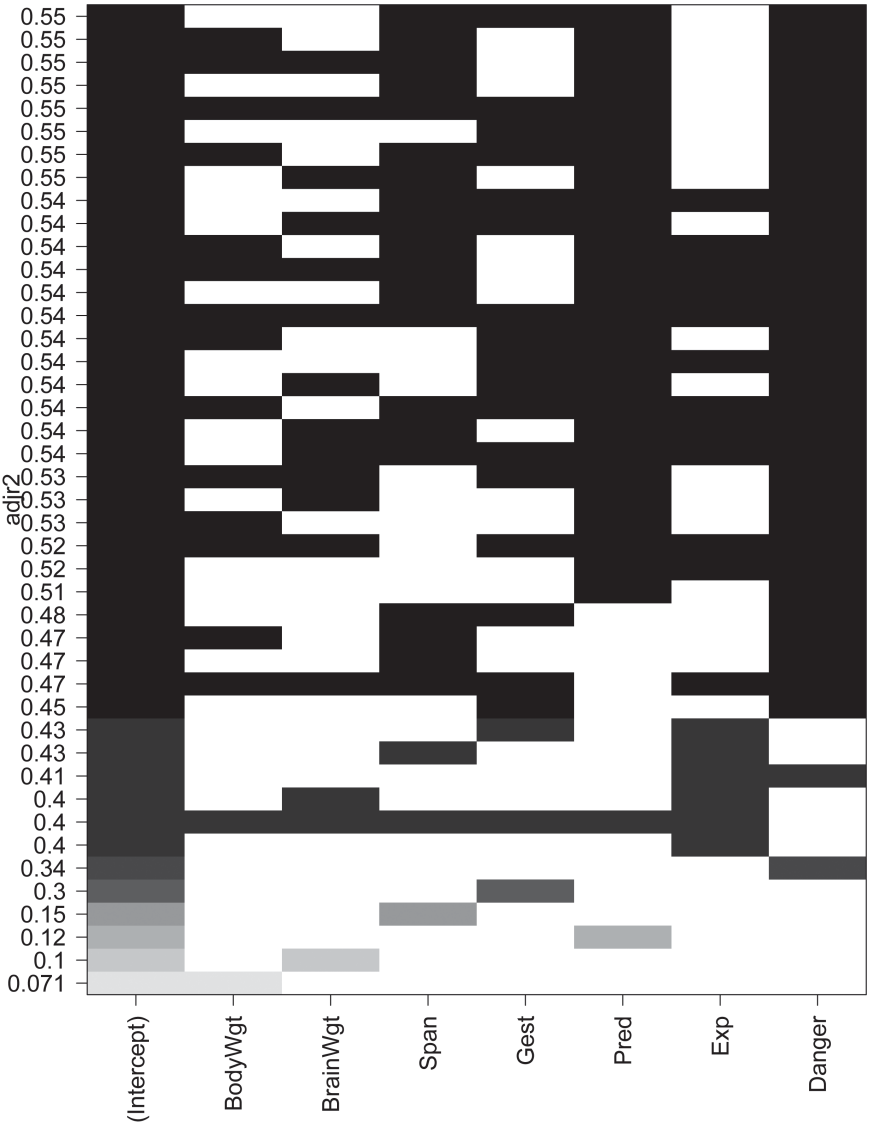


Рисунок 109

терии и при реализации алгоритмов пошагового включения и/или исключения предикторов. Один из таких подходов («*forward selection*») представлен в пакете `FWDselect` (Miller, 1990, http://bit.ly/1CpwrHu), который основан на скользящем контроле («*leave-one-out cross-validation*»), то есть подгоняются n регрессионных моделей по $(n - 1)$ выборочным значениям, а исключенное наблюдение каждый раз используется для расчета ошибки предсказания:

$$s_{CV} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}.$$

Алгоритм осуществляет перебор всех уровней сложности модели $\{k = 1, 2, ...\}$ и для каждого уровня находит модель, оптимальную с точки зрения критериев s_{CV} или R^2_{CV}. Примем глубину перебора k равной семи и построим график зависимости ошибки s_{CV} от k (рис. 110):

```
# Алгоритм "Forward selection" на основе перекрестной проверки
library(FWDselect)
qob = qselection(x = sleep_imp3[, -(3:5)],
                 y = sleep_imp3$Sleep, qvector = c(1:7),
                 method = "lm", criterion = "variance")
#  Вывод графика зависимости внешнего критерия от сложности модели
plot(qob)
```

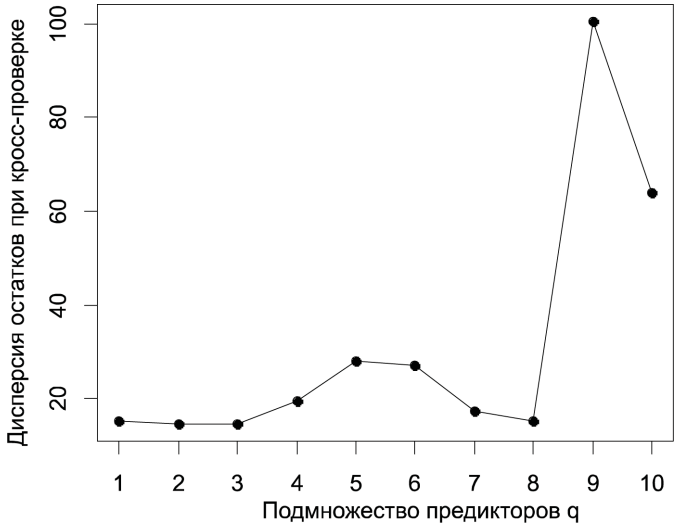


Рисунок 110

В качестве оптимальной принимается модель той размерности подмножества предикторов, которая обеспечивает оптимум выбранного внешнего критерия. В нашем случае минимум ошибки предсказания соответствует включению в модель четырех предикторов:

```
selection(x = sleep_imp3[, -(3:5)], y = sleep_imp3$Sleep,
          q = 3, criterion = "variance",
          method = "lm", family = "gaussian")
**************************************************
Best subset of size q = 3 : Pred   Span   Danger
Information Criterion Value - variance : 14.55175
**************************************************
```

Неожиданностей, как видим, не произошло – нами получена та же модель, что и при шаговом отборе по AIC-критерию.

7.8. Диагностика моделей множественной регрессии

Основные подходы, используемые для диагностики регрессионных моделей, были рассмотрены нами в разделе 7.3. Здесь мы применим некоторые из этих подходов для диагностики множественной регрессии, а именно для описанных в предыдущем разделе двух моделей зависимости продолжительности сна `Sleep` от экологических и таксономических показателей у животных (полная модель `M` и полученная шаговой селекцией модель `Mstep`). Кроме того, будут представлены некоторые дополнительные приемы диагностики.

Сравнение нескольких альтернативных моделей

Для начала выясним, имеются ли между моделями `M` и `Mstep` статистически значимые различия. Если сравниваемые модели являются *вложенными* (то есть когда модель 1 полностью включает весь набор параметров β_j модели 2), то при небольших различиях в степенях свободы оценить статистическую значимость между дисперсиями остатков RSE_1 и RSE_2 можно с использованием последовательного («*sequential*») теста Фишера:

$$F_0 = \frac{(RSS_2 - RSS_1)/q}{RSS_2/(n-k-1)} = \frac{n-k-1}{q} \times \frac{R_2^2 - R_1^2}{1-R_2^1}.$$

Этот тест можно выполнить при помощи базовой функции **`anova`**`()`:

```
anova(M, Mstep)
Analysis of Variance Table
Model 1: Sleep ~ BodyWgt + BrainWgt + Span + Gest + Pred + Exp + Danger
Model 2: Sleep ~ Span + Pred + Danger
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     54 582.06
2     58 610.85 -4    -28.79 0.6677 0.6172
```

Приведенная процедура дисперсионного анализа (так называемого I типа) не всегда позволяет выполнить корректный анализ, поскольку получаемые с ее помощью *p*-значения зависят от порядка перечисления предикторов в формуле модели (ситуация особенно усложняется при работе с несбаллансированными наборами данных). С связи с этим предпочтение стоит отдать функции **`Anova`**`()` из пакета `car`, в которой тест II типа оценивает значимость добавления каждого нового параметра с учетом частных эффектов предикторов, уже включенных в модель:

```
library(car)
Anova(M, Mstep)
Anova Table (Type II tests)
```

```
Response: Sleep
          Sum Sq Df F value    Pr(>F)
BodyWgt    18.42  1  1.7491 0.1911822
BrainWgt   14.72  1  1.3977 0.2419281
Span       39.13  1  3.7155 0.0588098 .
Gest        5.33  1  0.5058 0.4798043
Pred      104.47  1  9.9196 0.0025848 **
Exp         1.73  1  0.1644 0.6865898
Danger    179.35  1 17.0288 0.0001192 ***
Residuals 610.85 58
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Из представленных результатов можно сделать вывод, что исключение из полной модели комплекса переменных BodyWgt, BrainWgt, Gest, Exp не приведет к статистически значимому увеличению внутренней ошибки модели, и, если это отвечает интересам исследователя, он может принять для использования более компактную модель.

Однако равноценны ли эти модели при прогнозировании новых данных? Выполним для этого оценку ошибки при кросс-проверке с разбиением исходной выборки на $k = 3$ блока:

```
library(DAAG)
(cvObjM <- cv.lm(df = sleep_imp3, M, m = 3, seed = 0))
  ss   df  Overall ms
5073   62        81.8

(cvObjMstep <- cv.lm(df = sleep_imp3, Mstep, m = 3, seed = 0))
  ss   df  Overall ms
1213   62        19.6
```

Сумма квадратов отклонений расчетных и эмпирических значений для полной модели M в четыре раза превышает *CV*-ошибку модели Mstep. Внимательное изучение протокола работы программы позволяет сделать вывод, что эта разница во многом определяется сильной асимметрией распределения массы тела животных BodyWgt.

Диагностика допущений в отношении остатков модели

Так же, как и в случае простой регрессии, диагностика множественной модели начинается с проверки следующих стандартных предположений:

- нормальность распределения остатков и однородность их дисперсии;
- независимость остатков от значений предикторов X и прогнозов отклика $\hat{y}$;
- адекватность линейной формы модели для описания имеющихся данных.

Оценить зависимость остатков от прогнозируемых значений и нормальность распределения $\hat{\varepsilon}$ можно при помощи функции **plot**() (рис. 111) либо с использованием функции **sm.density**() пакета sm (рис. 112):

```
par(mfrow = c(1, 2)) #  Рис. 111
plot(Mstep, which = c(1, 2))

library(sm)
sm.density(rstudent(Mstep), model = "normal") #  Рис. 112
shapiro.test(resid(M))
        Shapiro-Wilk normality test
W = 0.973, p-value = 0.1777
```

Рисунок 111

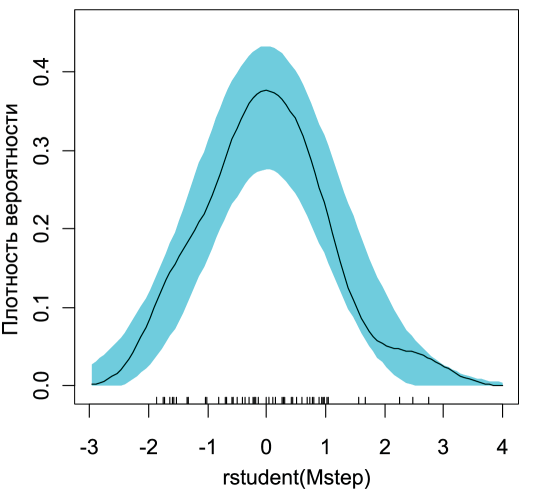


Рисунок 112

По рис. 111, 112 видно, что «выбросы» среди остатков полученной модели достаточно умеренны и у нас нет оснований отклонить гипотезу о нормальности их распределения. Гипотеза о нормальности распределения остатков подтверждается также результатами теста Шапиро-Уилка.

Как мы уже знаем (см. раздел 7.3), использованная в приведенном выше коде функция **rstudent**() выполняет расчет *стьюдентизированных остатков*. Благодаря тому, что такие остатки имеют распределение, близкое к распределению Стьюдента, мы можем использовать их для формальной проверки нулевой гипотезы о том, что некоторое наблюдение y_i не является выборосом (то есть хорошо согласуется с рассматриваемой моделью). В разделе 7.3 мы выполнили такую проверку, самостоятельно рассчитав наблюдаемое и критическое значения t-критерия для наблюдения с максимальным стьюдентизированным остатком. Однако для этого можно было бы использовать и готовую функцию – **outlierTest**() из пакета car. При расчете соответствующих p-значений эта функция автоматически вносит поправку Бонферрони (Bonferonni p):

```
outlierTest(Mstep, cutoff = 0.05)
No Studentized residuals with Bonferonni p < 0.05
Largest |rstudent|:
   rstudent unadjusted p-value Bonferonni p
21     2.84           0.00632        0.392
```

Как видим, для модели Mstep выбросы отсутствуют, и предположение о том, что наблюдение 21 с максимальным стьюдентизированным остатком не отличается от остальных, можно принять с вероятностью ошибки $p_{adj} = 0.392$. В противном случае целесообразно было бы удалить аномальное наблюдение и повторить подгонку модели.

Одним из подходов для выявления потенциально влиятельных наблюдений, то есть наблюдений, оказывающих существенное воздействие на оценки параметров модели (см. раздел 7.3), является использование так называемых *диаграмм добавленных переменных* («*added-variable plots*»). Такие диаграммы получают следующим образом:

- строят регрессионную зависимость отклика y от всех имеющихся предикторов x, за исключением x_i, и получают остатки $\hat{\delta}$ этой регрессии. Такие остатки соответствуют той части вариации y, которую не удается объяснить с помощью всех предикторов, кроме x_i;
- строят регрессионную зависимость предиктора x_i от остальных предикторов x (за исключением, конечно, самого x_i) и получают остатки $\hat{\gamma}$. Эти остатки отражают часть вариации x_i, которую не удается объяснить с помощью других предикторов;
- зависимость $\hat{\delta}$ от $\hat{\gamma}$ изображают графически. Угол наклона регрессионной прямой зависимости $\hat{\delta}$ от $\hat{\gamma}$ соответствует β_i, то есть эффекту предиктора x_i с учетом эффектов всех остальных предикторов.

Таким образом, диаграммы добавленных переменных отражают *частный эффект* предиктора x_i на отклик y. Интерес на этих диаграммах представляют не-

обычные наблюдения, «выпадающие» из общего тренда. Построить такие диаграммы автоматически можно при помощи функции **avPlots**() из пакета car или ее более обобщенной версии – **leveragePlot**():

```
par(mfrow = c(1, 2))
leveragePlot(Mstep, "Span")
leveragePlot(Mstep, "Danger")
```

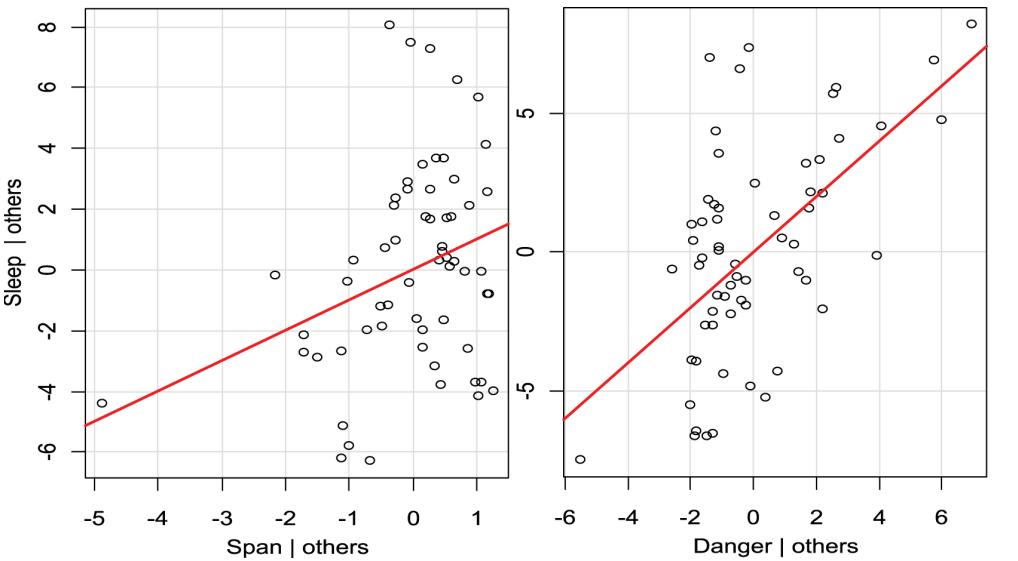


Рисунок 113

Наконец, проверить отсутствие автокорреляции в остатках модели (то есть некоторой периодической составляющей в них) можно с использованием критерия Дарбина-Уотсона (подробнее см. http://bit.ly/1bq9SNV):

```
durbinWatsonTest(Mstep)
 lag Autocorrelation D-W Statistic p-value
   1       0.2251362      1.509941   0.052
 Alternative hypothesis: rho != 0
```

Обратим теперь внимание на проблему статистической значимости параметров полученной модели. В общем случае оценка доверительных интервалов коэффициентов множественной регрессии параметрическими методами представляет собой нетривиальную проблему и основывается на сложных аналитических формулах, учитывающих ковариацию между предикторами. Эта задача упрощается с использованием процедур бутстрепа и рандомизации, техника реализации которых, описанная нами для парной регрессии, в равной степени может применяться и для многомерного случая.

С помощью функции **boot**() для найденной регрессионной модели рассчитаем доверительные интервалы и покажем на графике характер взаимного влияния и совместного распределения коэффициентов моделей для отдельных пар предикторов (рис. 114):

```
library(boot); library(car)

# функция, возвращающая вектор коэффициентов:
bootF <- function(data, indices){ data <- data[indices, ]
         mod <- lm(Sleep ~ Span + Pred + Danger, data = data)
  coefficients(mod) }
Sleep.boot <- boot(sleep_imp3, bootF, 1000)

# функция dataEllipse() входит в пакет car
dataEllipse(Sleep.boot$t[, 2], Sleep.boot$t[, 3],
            xlab = "Span", ylab = "Pred",
            cex = 0.3, levels = c(0.9, 0.95, 0.99), robust = T)
```

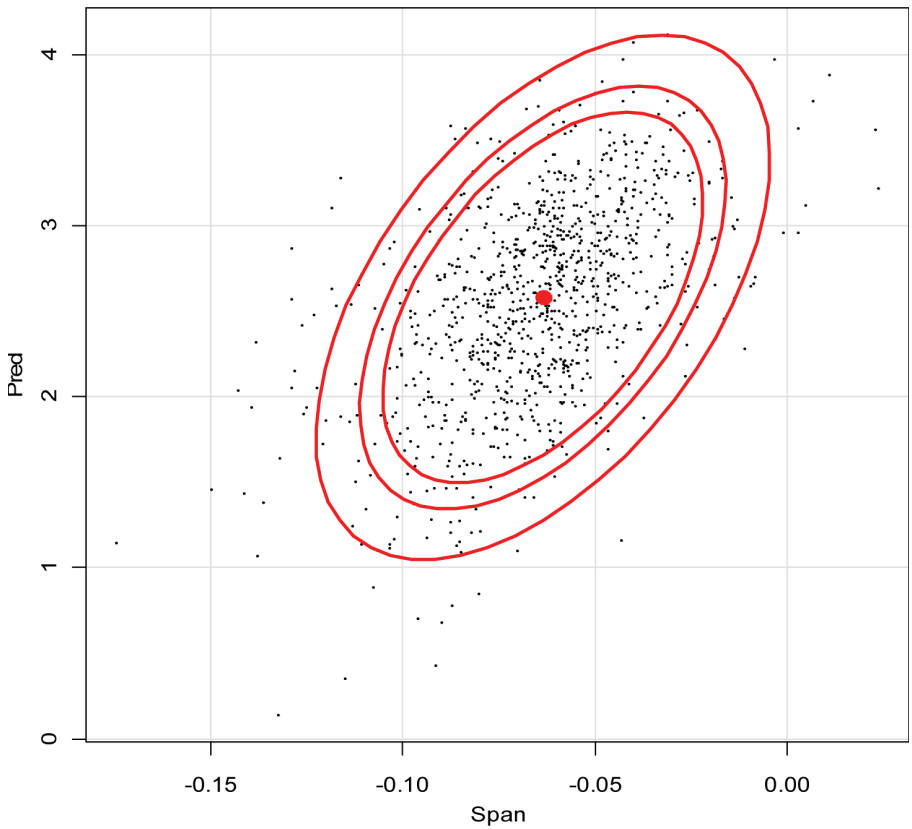


Рисунок 114

На рис. 114 показано поле корреляции бутстреп-оценок коэффициентов для двух переменных – Span (продолжительность жизни) и Pred (степень хищничества), а также доверительные эллипсы бутстреп-интервалов для вероятностей 0.9, 0.95 и 0.99, основанные на вычислении робастных оценок ковариационной матрицы параметров модели (Fox, 2002).

```
boot.ci(Sleep.boot, index = 2, type = "bca") # Для Span
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates
CALL :
boot.ci(boot.out = Sleep.boot, type = "bca", index = 2)
Intervals :
Level       BCa
95%   (-0.1120, -0.0204 )
```

Поскольку доверительный интервал для коэффициента переменной Span не включает значение 0, есть основания считать этот коэффициент статистически значимым.

Учет нелинейного характера влияния предикторов на отклик

Одной из проблем идентификации адекватных регрессионных моделей является проверка справедливости их линейной спецификации. Как уже отмечалось ранее, существующие в природе процессы в подавляющем большинстве своем являются нелинейными, а приведение их к линейной форме – лишь одна из возможностей удобной аппроксимации, которая может оказаться весьма грубой. Помимо соответствующего анализа остатков (см. раздел 7.3), нелинейность исследуемой зависимости в некоторых случаях можно выявить при помощи матричных диаграмм рассеяния (см. раздел 6.1).

Функция **scatterplotMatrix**() из пакета car позволяет построить диаграммы, отражающие как линейную аппроксимацию зависимости для каждой пары переменных, так и сглаживающую кривую с доверительными интервалами:

```
library(car)
scatterplotMatrix(~ Sleep + BodyWgt + BrainWgt + Span +
    Pred + Danger, data = sleep_imp3, diag = "boxplot")
```

По рис. 115 видно, что распределение биометрических показателей массы тела (BodyWgt) и массы мозга (BrainWgt) имеет сильно асимметричный характер: наряду с подавляющим большинством животных, обладающих небольшим весом, есть две особи с весом в несколько тонн. Можно усмотреть нелинейность также в зависимости продолжительности сна (Sleep) от продолжительности жизни (Span).

Однако проблема в том, как отразить эти визуальные впечатления в спецификации множественной регрессионной модели.

Простейший способ учесть нелинейность в модели – это заменить в правой части уравнения регрессии определенные предикторы на их некоторые разумные нелинейные преобразования и проверить, как изменится при этом принятый крите-

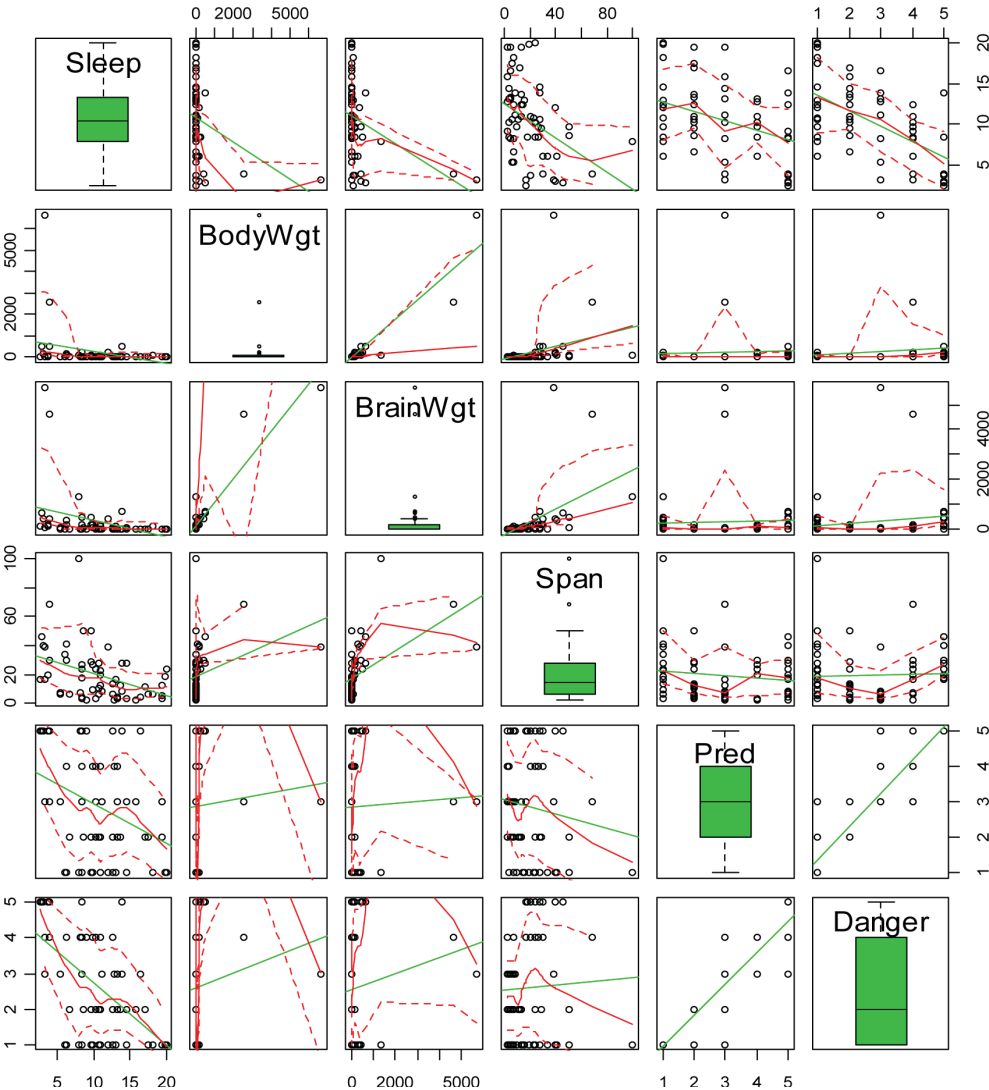


Рисунок 115

рий качества модели. В качестве примера добавим в исходную таблицу с данными новые переменные: прологарифмируем значения массы тела ln(BodyWgt) и массы мозга ln(BrainWgt), вычислим соотношение BodyWgt/BrainWgt, а также обратную величину 1/Span.

```
df_nl <- data.frame(ln_BodyWgt = log(sleep_imp3$BodyWgt),
       ln_BrainWgt = log(sleep_imp3$BrainWgt),
       Brain.Body = sleep_imp3$BrainWgt/sleep_imp3$BodyWgt,
```

```
       rev_Span = 1/sleep_imp3$Span)
ind.yx <- c(5, 1:2, 6:10)
sleep_nl <- cbind(sleep_imp3[, ind.yx], df_nl)
```

Заменим в формуле полной модели `M` предиктор `BodyWgt` на `ln_BrainWgt` и оценим изменение внешнего критерия – ошибку при однократной кросс-проверке с разделением на три блока:

```
M.nl1 <- lm(Sleep ~ ln_BodyWgt + BrainWgt + Span + Gest +
            Pred + Exp + Danger, data = sleep_nl)

(cvObjM.nl1 <- cv.lm(df = sleep_nl, M.nl1, m = 3, seed = 0))
  ss   df   Overall ms
1043   62         16.8
```

Столь незначительное изменение спецификации модели привело к тому, что ошибка кросс-проверки уменьшилась в пять раз (с 81.8 до 16.8) и оказалась меньше, чем для наилучшей модели `Mstep`, полученной на основе внутренних критериев.

Заметим, что ранее нам не удалось с использованием критерия AIC найти лучшую модель, чем `Mstep`. Комбинированный пошаговый метод, неустойчиво работающий в условиях высокой мультиколлинеарности, по-видимому, остановился на модели с почти полным составом предикторов и локальным минимумом AIC, далеким от глобального минимума. Функция **`glmulti`**`()` из одноименного пакета, реализующая *генетический алгоритм*, также выбрала в качестве оптимальной модель `Mstep` (подробности см. в книге Шитиков, Розенберг, 2014). Это обстоятельство подтверждает важный принцип разграничения сфер применимости критериев двух типов: если внутренние критерии ориентированы на построение как можно более компактных моделей, служащих целям объяснения, то внешние критерии призваны обеспечить в первую очередь низкую ошибку прогноза.

7.9. Регуляризация множественной регрессии

При оценивании параметров множественной линейной регрессии методом наименьших квадратов минимизируется функция потерь $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. Существует несколько причин, по которым полученные таким образом оценки могут оказаться неудовлетворительными:

- *верность предсказаний*: несмотря на то что получаемые при помощи МНК оценки параметров модели являются несмещенными, они характеризуются высокой неустойчивостью. Это означает, что даже при незначительных изменениях обучающих выборок оценки параметров могут значительно варьировать, а это, в свою очередь, будет сопровождаться неточными предсказаниями на новых данных;
- *интерпретация параметров и переобучение модели*: включение в модель большого числа предикторов (особенно коррелирующих друг с другом)

затрудняет выявление наболее важных из них и может приводить к переобучению модели. В связи с этим исследователь часто желает найти некоторое оптимальное подмножество наиболее информативных предиктов. Рассмотренные нами выше методы шаговой регрессии (раздел 7.7), к сожалению, грешат неустойчивыми решениями;

- *некорректность задачи:* в целом ряде областей можно столкнуться с данными, в которых число переменных намного превышает число имеющихся наблюдений (например, в генетических исследованиях, когда активность экспрессии тысяч генов измеряют лишь у нескольких пациентов). При таком сценарии математическая задача восстановления регрессии методом наименьших квадратов является некорректной, поскольку не имеет однозначного решения (подробнее см. http://bit.ly/1K5KacR).

Одним из подходов, широко применяемых для борьбы с указанными проблемами, является *регуляризация* – добавление дополнительной информации к исходному условию с целью предотвратить переобучение модели или решить некорректно поставленную задачу (http://bit.ly/1FeZ3d9).

С математической точки зрения, идея регуляризации заключается в том, чтобы в функцию потерь *RSS*, которую мы минимизируем при нахождении оценок параметров модели при помощи МНК, добавить дополнительный член, зависящий от самих этих параметров. Например, вместо *RSS* можно было бы минимизировать выражение:

$$L_1 = RSS + \lambda \sum_{i=1}^{n} |\beta_i|,$$

или другой вариант:

$$L_2 = RSS + \lambda \sum_{i=1}^{n} \beta_i^2.$$

Тем самым мы вводим штраф за неоправданно большие значения β_i, причем величина этого штрафа пропорциональна величине некоторого *гиперпараметра* λ («*tuning parameter*»), с помощью которого теперь можно настраивать алгоритм подгонки модели и пытаться получать более устойчивые решения. Первый вариант носит название L_1*-регуляризации* и применяется в *лассо-регрессии Тибширани*, второй – L_2*-регуляризации*, или регуляризации Тихонова и лежит в основе *гребневой регресии* (интересное обсуждение реализации этих методов в R приведено на сайте http://bit.ly/1HrMsmV). В приведенных ниже примерах использования гребневой регрессии и метода лассо мы будем применять набор данных `sleep_nl` по продолжительности сна животных (см. предыдущий раздел).

Гребневая регрессия

Гребневая регрессия («*ridge regression*») реализует классическую L_2-регуляризацию Тихонова. В матричном виде гребневая регрессия представляет собой линейную модель

$$\hat{\mathbf{y}} = \mathbf{H}_{ridge}\mathbf{y},$$

где

$$\mathbf{H}_{ridge} = \mathbf{X}(\mathbf{X}^{\mathbf{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathbf{T}},$$

то есть к матрице проекции (см. раздел 7.3) перед ее обращением добавляется диагональная матрица $\lambda\mathbf{I}$, называемая «гребнем» («*ridge*»). В результате матрица становится хорошо обусловленной, оставаясь в то же время «похожей» на исходную.

При $\lambda \to 0$ регуляризованное решение стремится к МНК-решению. При увеличении λ вектор коэффициентов $\hat{\beta}$ становится все более устойчивым и жестко определенным (то есть фактически происходит понижение эффективной размерности решения). При $\lambda \to \infty$ чрезмерная регуляризация приводит к вырожденному решению ($\hat{\beta} \to 0$), хотя на практике такой подход не используется, то есть коэффициенты не «ужимаются» до 0. Для нахождения оптимального значения λ чаще всего применяют так называемый *обобщенный критерий перекрестной проверки GCV* (от «*generalized cross-validation*»; подробности см. в оригинальной статье Golub et al., 1979, http://bit.ly/1Jdk3mB).

Построим модель гребневой регрессии при помощи функции **lm.ridge**() из пакета MASS. Зададим последовательность из нескольких значений λ (параметр lambda), поскольку оптимальное значение этого параметра заведомо неизвестно:

```
library(MASS)
M.ridge <- lm.ridge(Sleep ~ ., data = sleep_nl, lambda = seq(0, 2, 0.1))
plot(x = M.ridge$lambda, y = M.ridge$GCV, type = "o")
```

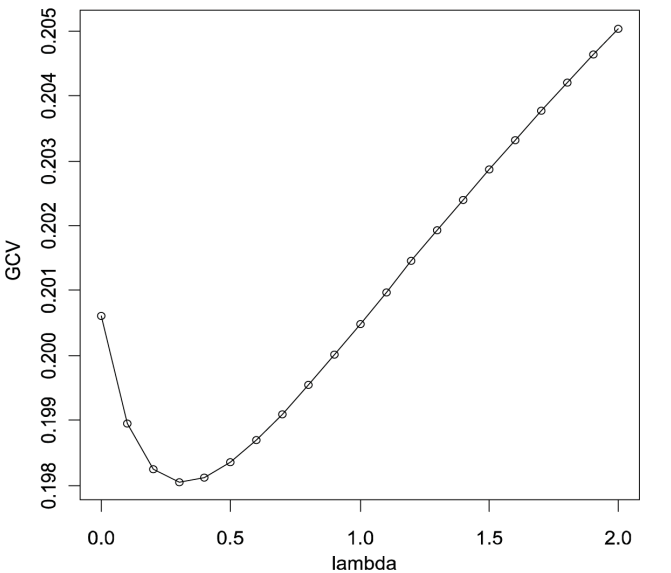


Рисунок 116

Функция **lm.ridge**() рассчитала серию из 20 моделей со значениями λ от 0 до 2. Найдем коэффициенты модели, для которой λ обеспечивает минимум критерия *GCV*:

```
lambda <- M.ridge$GCV[which.min(M.ridge$GCV)]
[1] 0.3

(M.ridge1 <- lm.ridge(Sleep ~., data = sleep_nl, lambda = lambda))
            BodyWgt  BrainWgt      Span      Gest       Pred
19.437482 -0.002578  0.002567 -0.081775 -0.002898   2.400439
      Exp    Danger ln_BodyWgt ln_BrainWgt Brain.Body  rev_Span
-0.026019 -3.955231   1.261660   -1.667830   0.094902 -8.150611

# Коэффициенты модели
beta.M.ridge1 <- coef(M.ridge1)
m <- length(beta.M.ridge1)

# Остатки модели
resid.ridge <- sleep_nl$Sleep - beta.M.ridge1[1] -
 as.matrix(sleep_nl[, 2:m])%*%beta.M.ridge1[2:m]

# Найдем число степеней свободы гребневой регрессии
d <- svd(as.matrix(sleep_nl[, 2:m]))$d
df <- nrow(sleep_nl) - sum(d^2/(lambda + d^2))
[1] 51.37781

# Средний квадрат отклонений модели
rss.ridge <- sum(resid.ridge^2)/df
[1] 10.23191
```

Поскольку найденное значение параметра регуляризации λ = 0.3 не столь значительно превышает 0, коэффициенты модели несильно отличаются от модели, полученной для той же таблицы данных методом наименьших квадратов.

Лассо-регрессия Тибширани

Метод регрессии «*лассо*» (*LASSO*, от англ. «*Least Absolute Shrinkage and Selection Operator*») реализует описанную выше L_1-регуляризацию (Tibshirani, 1996, http://stanford.io/1yQLtvb). В сравнении с гребневой регрессией замена штрафного слагаемого $\lambda\sum_{j=1}^{m}\beta_j^2$ на $\lambda\sum_{j=1}^{m}|\beta_j|$ приводит к качественно иному поведению вектора коэффициентов в ходе оптимизации: некоторые коэффициенты становятся равными нулю, что позволяет одновременно выполнить отбор информативных предикторов.

Иными словами, если ввести ограничение-неравенство, регулирующее чрезмерно большую сумму модулей стандартизованных коэффициентов $\sum_{j=1}^{m}|\beta_j^0|\le\tau$, то при некотором $\tau_{опт}$ достигается компромисс между величиной ошибки ре-

грессии и размерностью используемого пространства переменных, выраженного суммой абсолютных значений $|\beta^0|$. При значении *параметра фракционирования* $s = \tau / \sum_{j=1}^{m} |\beta_j^0| = 1$ лассо-регрессия сводится к обычному методу наименьших квадратов, а по мере его уменьшения $s \to 0$ формируемая модель становится все более лаконичной (с точки зрения числа входящих в нее параметров), пока все коэффициенты не станут равными 0. Оптимальную величину s находят путем минимизации *C_p-критерия Мэллоу* («*Mallow's C_p*», http://bit.ly/1P7mqGa) либо с использованием перекрестной проверки.

Построим регрессионную лассо-модель с использованием функции **lars**() из одноименного пакета:

```
library(lars)
Xmat <- as.matrix(sleep_nl[, -1])
M.las <- lars(Xmat, sleep_nl[, 1], type = "lasso")
plot(M.las, plottype = "coefficients")
plot(M.las, plottype = "Cp")

# Параметр фракционирования s найдем перекрестной проверкой
set.seed(0); s <- cv.lars(Xmat, sleep_nl[, 1])
```

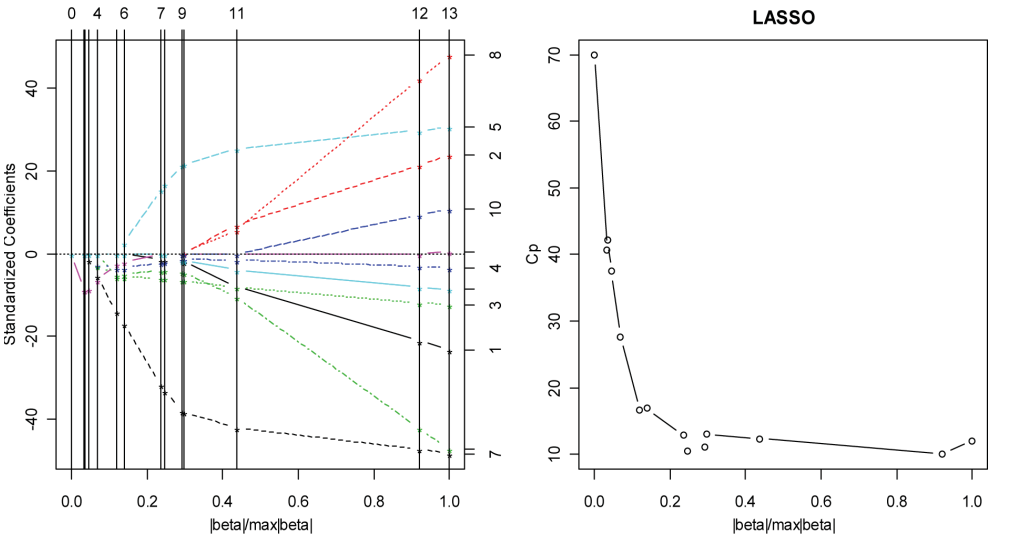


Рисунок 117

На рис. 117 слева видно, как изменяются значения стандартизованных коэффициентов регрессии при увеличении параметра фракционирования *s*. При $s = 0$ все коэффициенты равны 0, а при $s = 1$ получаем полную модель. Вертикальные линии отмечают значения *s*, при которых происходит обнуление определенных коэффициентов. На рис. 117 справа показано, как в тех же условиях изменяется

критерий Мэллоу. Минимум ошибки наблюдается при $s = 0.22$. Из всех возможных коэффициентов модели при этом оптимальном значении s семь оказались ненулевыми:

```
(bestfrac <- s$index[which.min(s$cv)])
 0.2222222

las.coef <- predict(M.las, Xmat, s = bestfrac,
                    type = "coefficient", mode = "fraction")
las.coef  # Просмотр коэффициентов модели
$coefficients
   BodyWgt    BrainWgt        Span        Gest        Pred         Exp
 -0.000172    0.000000  -0.042027  -0.002149    1.171393  -0.0246898
    Danger  ln_BodyWgt ln_BrainWgt Brain.Body    rev_Span
-2.6286958   0.0000000  -0.2210219   0.0000000   0.0000000

# Остатки модели
las.resid = sleep_nl$Sleep - predict.lars(M.las, Xmat,
            s = bestfrac, type = "fit", mode = "fraction")$fit
rss.lasso <- sum(las.resid^2)/(nrow(sleep_nl) - 7)
11.28749
```

В связи со сходными математическими предпосылками, лежащими в основе методов лассо и гребневой регрессии, у пользователя может возникнуть вопрос о том, какой из них предпочесть. Ответ на этот вопрос, как это часто бывает, зависит от конкретной ситуации, а именно от стоящей перед исследователем задачи и свойств данных. В целом ни один из этих методов не обладает явным преимуществом над другим. Метод лассо хорошо подходит для случаев с большим числом предикторов, когда у исследователя имеются основания полагать, что истинные значения коэффициентов некоторой части предикторов равны нулю. Соответственно, лассо следует использовать при необходимости снизить размерность исходного пространства предикторов и выбрать наиболее информативные из них. В свою очередь, гребневую регрессию стоит применять в случаях, когда зависимая переменная является функцией от многих предикторов с близкими значениями коэффициентов. Безусловно, оптимальное число предикторов для моделей, построенных по реальным данным, никогда не известно заранее, и поэтому следует активно пользоваться методами перекрестной проверки (James et al., 2013).

7.10. Регрессия на главные компоненты

Продолжая разговор о методах снижения размерности исходной задачи, особенно при наличии многих коррелирующих друг с другом предикторов, в этом разделе мы рассмотрим метод *главных компонент* (*«principal component analysis»*, PCA) – один из фундаментальных методов многомерного анализа.

Предположим, что мы имеем матрицу наблюдений $\mathbf{X}$ размерностью $n \times m$, которая может быть представлена в многомерном пространстве облаком точек, имеющим

конфигурацию удлиненного сплющенного эллипсоида (рис. 118). Представим эти точки в новой системе координат, которая будет лучше отражать внутреннюю структуру анализируемых данных. Первую ось PC_1 (ее называют первой *главной компонентой*) проведем в направлении наибольшей дисперсии данных, совпадающем с главной осью многомерного эллипсоида. Вторую главную компоненту PC_2 проведем перпендикулярно к PC_1 в направлении наибольшего разброса наблюдений в этой ортогональной плоскости. Третью главную компоненту и все последующие находят по аналогичному принципу.

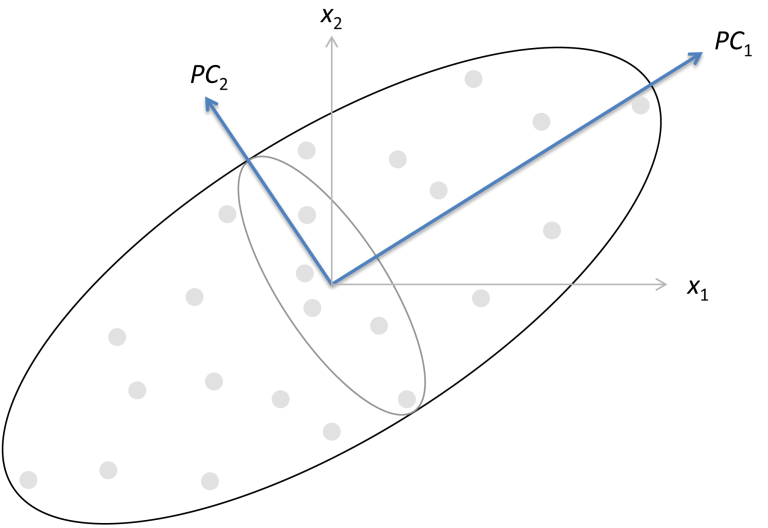


Рисунок 118

Чего мы этим достигли? Во-первых, в новой системе координат оси расположены ортогонально, а это значит, что переменные PC_1, PC_2 и т. д. теперь статистически независимы и проблема мультиколлинеарности, о которой мы много говорили в предыдущих разделах, попросту отпала. Во-вторых, главные компоненты упорядоченно связаны с общим разбросом в данных, то есть первая из них объясняет наибольшую долю вариации исходных данных, вторая – несколько меньшую долю и т. д. Наконец, выделив u осей новой системы координат ($m > u$), можно сократить количество предикторов, то есть уменьшить размерность задачи, пожертвовав некоторой минимальной долей неучтенной исходной дисперсии.

Результатом PCA-преобразования матрицы наблюдений **X** является матрица **T** *факторных значений*, или «счетов» («*scores*»), размерностью $n \times u$. Другая матрица **P** размерностью $u \times m$ содержит *факторные нагрузки* («loadings») – коэффициенты пропорциональности, обеспечивающие проекцию данных из m-мерного пространства исходных переменных в u-мерное пространство главных компонент (подробнее см., например, http://bit.ly/1FhPha2):

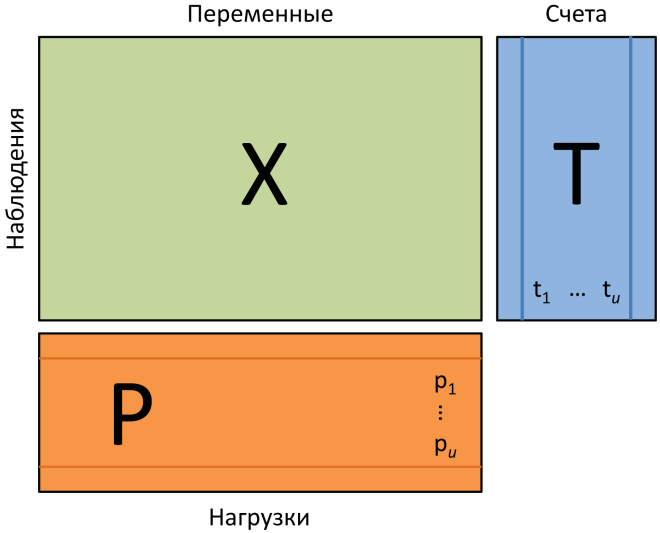


Рисунок 119

Каждая k-я строка матрицы **P** состоит из оценок коэффициентов p_{jk}, отражающих долю участия j-й исходной переменной в формировании k-й главной компоненты (фактически это проекция x_j на новую k-ю ось). Математически связь между этими матрицами выглядит как простое линейное преобразование $\mathbf{XP} \rightarrow \mathbf{T}$. Проблема остается только в том, как вычислить нагрузки p_{jk} на главные компоненты.

Пусть $\mathbf{C} = \mathbf{X}^t\mathbf{X}$ – корреляционная матрица (значения **X** были предварительно стандартизованы). Тогда для представленных матриц имеет место следующее соотношение:

$$\mathbf{T}^t\mathbf{T} = \mathbf{P}^t\mathbf{C}\mathbf{P} = \Lambda, \text{ где } \Lambda = \text{diag}\{\lambda_1, ..., \lambda_m\},$$

то есть дисперсии столбцов матрицы **T** соответствуют *собственным значениям* λ_j («*eigen values*») матрицы **C**, являющимся диагональными элементами матрицы Λ. Что такое собственные значения и как их рассчитать – задача линейной алгебры, на которой мы не будем останавливаться. Важно то, что, найдя значения λ_j, мы легко получаем и все остальное.

Регрессионное уравнение, использующее в качестве независимых переменных главные компоненты, в матричной форме имеет вид $\mathbf{y} = \mathbf{T}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, и его коэффициенты могут быть найдены обычным методом наименьших квадратов.

Остальные детали метода рассмотрим на примере с изучением потребности во сне у животных, уже знакомом нам по предыдущим разделам. Для реализации PCA используем базовую функцию **princomp**():

```
load(file = "sleep_imp.Rdata")
sleep_imp <- as.data.frame(scale(sleep_imp3))
```

```
Sleep.pca <- princomp(~BodyWgt + BrainWgt + Span + Gest +
                      Pred + Exp + Danger, data = sleep_imp)
summary(Sleep.pca)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4     Comp.5      Comp.6      Comp.7
Standard deviation     1.877693 1.4294425 0.8997436 0.5019613 0.41554081 0.226892702 0.180001283
Proportion of Variance 0.511933 0.2966861 0.1175442 0.0365851 0.02507213 0.007474891 0.004704517
Cumulative Proportion  0.511933 0.8086191 0.9261634 0.9627485 0.98782059 0.995295483 1.000000000
```

Из результатов видим, что первая главная компонента объясняет 51.2% общей вариации данных (`Proportion of Variance`), а первые три компоненты вместе – 92.6% (`Cumulative Proportion`). *Критерий Кайзера-Гуттмана* (http://bit.ly/1OF6XCw) рекомендует оставить для дальнейшего анализа только те главные компоненты, собственные значения которых превышают среднее всех собственных значений (рис. 120):

```
# Оценка необходимого числа главных компонент:
ev <- Sleep.pca$sdev
ev[ev > mean(ev)]  # Критерий Кайзера-Гуттмана

# Столбиковая диаграмма собственных значений:
barplot(ev, main = "Собственные значения", col = "bisque", las = 2)
abline(h = mean(ev), col = "red")
legend("topright", "Среднее собственных значений", lwd = 1, col = 2, bty = "n")
```

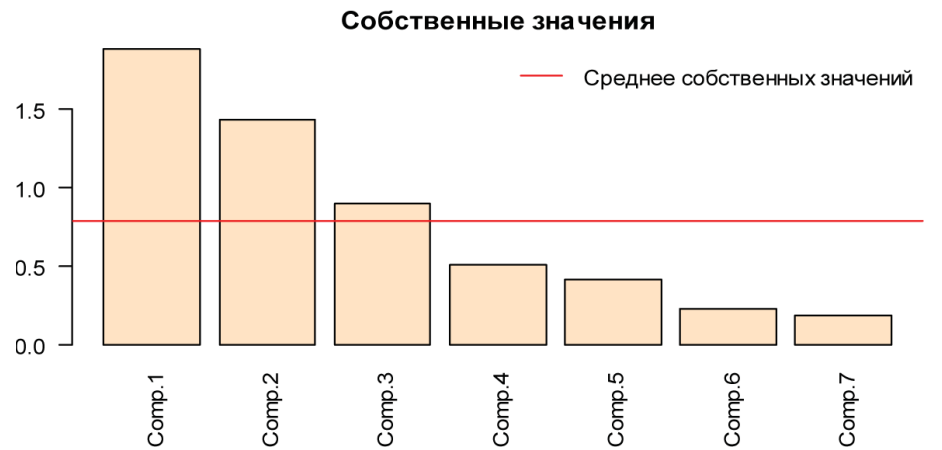


Рисунок 120

Анализируя таблицу нагрузок на главные компоненты, можно предположить, что вторая главная компонента связана с наблюдаемыми переменными `Pred` и `Danger`, а третья – с `BodyWgt` и `Span`, тогда как на `Comp.1` относительно равномерно проецируются все исходные предикторы:

```
P <- loadings(Sleep.pca)[,1:3]
print(P,3)
         Comp.1 Comp.2   Comp.3
BodyWgt  -0.382 -0.323  0.53402
BrainWgt -0.414 -0.363  0.30478
Span     -0.288 -0.312 -0.72764
Gest     -0.468 -0.163 -0.15051
Pred     -0.267  0.558  0.15573
Exp      -0.435  0.265 -0.21344
Danger   -0.346  0.507  0.00313
```

Подготовим данные для регрессионного анализа (то есть составим матрицу счетов **T**) и выполним построение линейной модели на главные компоненты:

```
T <- predict(Sleep.pca)[, 1:3]
sleep_PCA <- as.data.frame(cbind(Sleep = sleep_imp3[, 5], T))
M.PCA <- lm(Sleep ~ ., data = sleep_PCA)

summary(M.PCA)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.5774     0.4526  23.371  < 2e-16 ***
Comp.1        1.5518     0.2410   6.438 2.56e-08 ***
Comp.2       -0.5103     0.3166  -1.612   0.1125
Comp.3        1.0908     0.5030   2.169   0.0342 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.564 on 58 degrees of freedom
Multiple R-squared: 0.4567,     Adjusted R-squared: 0.4286
F-statistic: 16.25 on 3 and 58 DF,  p-value: 8.801e-08

AIC(M.PCA)
[1] 339.3941
```

По величине критерия AIC полученная модель регрессии на главные компоненты оказалась несколько хуже многих других моделей, построенных по тем же данным. Кроме того, такую модель сложнее интерпретировать.

Метод *частных наименьших квадратов* («*partial least squares*» или «*projection into latent structure*», PLS) также использует проекцию предикторов на оси главных компонент, но дополнительно выделяет подмножество латентных переменных, в пространстве которых связь между зависимой переменной и предикторами достигает максимального значения (то есть формирует вектор весов зависимой переменной для главных компонент). Реализовать PLS-регрессию в среде R можно при помощи одноименного пакета. Ниже приведен пример модели, подогнанной функцией **plsr**() на основе четырех главных компонент (эмпирически было найдено, что использование четырех компонент обеспечивает минимум ошибки):

```
library(pls)
M.pls <- plsr(Sleep ~ ., 4, data = sleep_imp3, method = "oscorespls")
summary(M.pls)
Data:   X dimension: 62 9
        Y dimension: 62 1
Fit method: oscorespls
Number of components considered: 4
TRAINING: % variance explained
       1 comps  2 comps  3 comps  4 comps
X         96.1    99.10    99.99   100.00
Sleep     10.7    19.64    30.42    53.96


beta.pls <- drop(coef(M.pls))
     BodyWgt     BrainWgt         NonD        Dream         Span         Gest         Pred
-0.003965451  0.004859369  0.298043025  0.076676001 -0.161543351 -0.007841193 -0.036476207
          Exp      Danger
 -0.046134970 -0.055274459

# Расчет ошибки:
resid.pls <- drop(M.pls$resid)[, 4]
rss.pls <- sum(resid.pls^2)/(nrow(sleep_imp3) - 5)
rss.pls
[1] 10.95129
```

Определение того, какая из моделей продолжительности сна животных, описанных в последних разделах, является оптимальной, представляет собой непростую проблему из-за отсутствия сопоставимых объективных критериев качества. В частности, мы не стали рассчитывать критерий AIC для моделей с регуляризацией и PLS-метода из-за ряда проблем вычислительного характера. Поэтому приведем ниже сравнительную таблицу средних квадратов остатков:

```
# Сравнение RSS семи моделей
c(rss.M = rss.ls <- sum(M$resid^2)/M$df.residual,
rss.Mstep = rss.ls <- sum(Mstep$resid^2)/Mstep$df.residual,
rss.M.nl1 = rss.ls <- sum(M.nl1$resid^2)/M.nl1$df.residual,
rss.ridge = rss.ridge,
rss.lasso = rss.lasso,
rss.M.PCA = rss.ls <- sum(M.PCA$resid^2)/M.PCA$df.residual,
rss.pls = rss.pls)
    rss.M rss.Mstep rss.M.nl1 rss.ridge rss.lasso rss.M.PCA   rss.pls
 10.77889  10.53189  11.11991  10.23191  11.28749  12.70027  10.95129
```

Хотя наиболее точная подгонка модели к данным была выполнена с применением гребневой регрессии, выше было показано, что использование критерия *RSS* для оценки качества не всегда оправдано. Альтернативное решение этой проблемы может дать вычислительный эксперимент с использованием имитационных методов (см. раздел 7.1).

7.11. Сравнение эффективности различных моделей при прогнозировании

Выше были рассмотрены некоторые методические проблемы построения линейных моделей, в том числе: *а*) методы *оценивания параметров* модели («*paramter estimation*»); *б*) поиск оптимального сочетания предикторов («*model selection*») и *в*) сравнительная диагностика моделей-претендентов. При этом мы старались опираться на общепринятые внутренние критерии качества моделей (коэффициент детерминации, *F*-критерий, информационные критерии Акаике, Байеса, Шварца и др.), которые могут косвенно отражать «объясняющую, или познавательную, ценность» результатов регрессионного анализа. Внешние критерии, такие как ошибка кросс-проверки, ориентированы в значительной мере на «потребительскую ценность» модели, то есть ее максимальную эффективность при прогнозировании величины отклика на новых данных. Подробное обсуждение различий в функциях *объяснения* и *прогнозирования* при моделировании наблюдаемых феноменов можно найти в фундаментальной монографии Г. С. Розенберга (2013, http://bit.ly/1Qpk1u2).

Многократное деление исходной выборки на *обучающую* и *проверочную*, по которой осуществляется «подстройка» («*tuning*») параметров модели, в большинстве случаев весьма эффективно, но не всегда позволяет получить несмещенную оценку качества прогноза, особенно на небольших выборках. Несмещенную оценку качества прогноза можно получить только на «свежих» данных, не принимавших никакого участия в формировании функции регрессии, то есть с использованием *экзаменационной* последовательности. В связи с этим ниже мы будем использовать не реальные, а сымитированные наборы данных, что позволит сформировать из одной условной генеральной совокупности и обучающую, и экзаменационную выборки любого необходимого объема.

В этом разделе использованы методические материалы профессора К. Саттона (Clifton Sutton, George Mason University, http://bit.ly/1EjzGWM). Описание использованных алгоритмов построения моделей и особенности их реализации в R приведены в предыдущих разделах.

Формирование исходных данных для построения моделей

Сгенерируем обучающую выборку умеренного объема ($n_t = 50$), включающую отклик `y` и шесть независимых переменных `x1-x6`. Для этого используем функции **`runif`**`()` и **`rnorm`**`()`, получая реализации случайных величин из распределений с заданными параметрами:

```
# Установим стартовое "зерно" генератора случайных чисел для
# воспроизводимости примера:
set.seed(632)
n = 50  #  Объем генерируемой выборки
x1 <- runif(n, 0, 1)
x2 <- x1 + rnorm(n, 0, 0.25)
```

```
x3 <- (x1 + x2)/2 + runif(n, 0, 0.1)
x4 <- runif(n, 0, 1)
x5 <- (2*x4 + rnorm(n, 0, 0.25))/2 + runif(n, 0, 0.1)
x6 <- runif(n, 0, 1)
y <- 3 + x1 + x2 + 0.5*x3 + 0.75*x4 + 0.5*x5 + 0.5*x6 + rnorm(n, 0, 1)

# Для последующих вычислений найдем средние и дисперсии:
x <- cbind(x1, x2, x3, x4, x5, x6)
mx <- apply(x, 2, mean)
varx <- apply(x, 2, var)

# Стандартизация всех предикторов X:
x <- scale(x)
trdata <- data.frame(cbind(y, x))

print(trdata, digits = 4)
       y        x1       x2        x3       x4       x5       x6
1  5.137  0.911254  1.72750  1.332126 -1.40850 -1.59800  1.05042
2  5.288 -0.865048 -1.11143 -1.157837  1.47749  1.49121  0.87614
… … … …
48 4.214  0.972390  0.85598  0.873112 -1.18340 -0.91012  0.53214
49 5.245 -0.201314  0.32514  0.019597  0.51390  0.65354 -1.68411
50 5.110  0.721397 -0.00579  0.425851 -1.53167 -1.94168  0.35642

attach(trdata)
```

Как показано в разделе 6.1, необходимым этапом разведочного анализа данных являются оценка коэффициентов корреляции и визуализация парных зависимостей между исследуемыми переменными. Выполним это для таблицы `trdata`:

```
library(Hmisc)

# Функция расчета корреляционной матрицы с p-значениями:
corsig <- function(x){
     x <- as.matrix(x) ; R <- rcorr(x)$r ; p <- rcorr(x)$P
     ipa <- lower.tri(p, diag = FALSE) ; R[ipa] <- p[ipa]
 return (R)
}

round(corsig(trdata), 4)
        y     x1     x2     x3      x4      x5      x6
 y      1  0.693  0.553  0.651  -0.011   0.028   0.014
x1      0      1  0.768  0.918  -0.319  -0.268  -0.015
x2      0      0      1  0.953  -0.201  -0.180  -0.002
x3      0      0      0      1  -0.272  -0.240  -0.023
x4  0.939  0.023  0.161  0.056       1   0.911   0.019
x5  0.847  0.059  0.211  0.092       0       1   0.083
x6  0.920  0.919  0.989  0.876   0.894   0.564       1
```

Выше главной диагонали полученной матрицы представлены коэффициенты корреляции Пирсона, а ниже – статистическая значимость их отличий от 0. Очевидно, что переменные y, x1, x2 и x3 образуют тесно связанный коллинеарный комплекс. Также высоко коррелируют между собой предикторы x4 и x5.

Общая линейная модель и ее тестирование на проверочной выборке

Получив некоторое представление о взаимодействиях переменных, построим несколько альтернативных моделей линейной регрессии. Начнем с обыкновенной МНК-модели, включающей все независимые переменные:

```
ols1 <- lm(y ~ x1 + x2 + x3 + x4 + x5 + x6)

summary(ols1)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.18630    0.14205  36.510   <2e-16 ***
x1           0.61073    0.83894   0.728    0.471
x2          -0.50784    1.09678  -0.463    0.646
x3           0.90961    1.77767   0.512    0.611
x4           0.15647    0.36113   0.433    0.667
x5           0.18453    0.35716   0.517    0.608
x6           0.03007    0.14731   0.204    0.839
---
Residual standard error: 1.004 on 43 degrees of freedom
Multiple R-squared: 0.5352,     Adjusted R-squared: 0.4703
F-statistic: 8.251 on 6 and 43 DF,  p-value: 5.747e-06
```

Бытует неверное мнение, что величины *t*-критерия для оценки значимости оценок коэффициентов модели в какой-то мере отражают относительную информативность каждого предиктора. В нашем случае все коэффициенты примерно в равной степени статистически незначимы, что является простым следствием небольшого объема исходной обучающей выборки. Если, например, ту же модель построить на выборке с $n = 1\,000\,000$, то все коэффициенты, как «по волшебству», становятся статистически значимыми (то есть наблюдается эффект «слишком большой» выборки, который мы обсуждали в разделе 5.3, посвященном анализу статистической мощности):

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.001321   0.003618  829.61   <2e-16 ***
x1          1.001048   0.018097   55.32   <2e-16 ***
x2          0.997782   0.017753   56.20   <2e-16 ***
x3          0.502433   0.034600   14.52   <2e-16 ***
x4          0.737344   0.008519   86.55   <2e-16 ***
x5          0.507769   0.007787   65.21   <2e-16 ***
x6          0.501448   0.003460  144.93   <2e-16 ***
```

```
---
Residual standard error: 0.9988 on 999993 degrees of freedom
Multiple R-squared: 0.4371,     Adjusted R-squared: 0.4371
F-statistic: 1.294e+05 on 6 and 999993 DF,  p-value: < 2.2e-16
```

Для независимой проверки предсказательных свойств модели ols1 и всех последующих моделей сформируем репрезентативную экзаменационную выборку, состоящую из $n_e = 5000$ наблюдений, полученных по тому же алгоритму случайной генерации. Чтобы наш эксперимент был корректным (то есть чтобы обучающая и тестовая выборки были сравнимы), при стандартизации переменных экзаменационной выборки будем использовать соответствующие средние и дисперсии нестандартизованной обучающей выборки.

```
gx1 <- runif(5000, 0, 1)
gx2 <- gx1 + rnorm(5000, 0, 0.25)
gx3 <- (gx1 + gx2)/2 + runif(5000, 0, 0.1)
gx4 <- runif(5000, 0, 1)
gx5 <- (2*gx4 + rnorm(5000, 0, 0.25))/2 + runif(5000, 0, 0.1)
gx6 <- runif(5000, 0, 1)
gy <- (3 + gx1 + gx2 + 0.5*gx3 + 0.75*gx4 + 0.5*gx5 + 0.5*gx6 + rnorm(5000, 0, 1))
gx <- cbind(gx1, gx2, gx3, gx4, gx5, gx6)
for (i in 1:6) gx[, i] <- (gx[, i]-mx[i])/sqrt(varx[i])
gendata <- data.frame( cbind(gy, gx) )
names(gendata) <- c("y", "x1", "x2", "x3", "x4", "x5", "x6")
summary(gendata)
```

Сравнение эффективности моделей для прогнозирования будем осуществлять, оценивая средний квадрат ошибки прогноза на данных из экзаменационной выборки:

```
ols1.mspe <- mean( (gy - predict(ols1, newdata = gendata))^2 )
1.194602
```

Значения этого показателя (.mspe), вычисленные для всех тестируемых моделей, мы в итоге представим в одной сводной таблице.

Выбор информативного комплекса предикторов

Выполним процедуру пошаговых включений и исключений параметров модели на основе критерия Акаике:

```
ols2 <- step(ols1, direction = "both")

summary(ols2)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.1863     0.1367  37.931  < 2e-16 ***
x1            1.0415     0.1434   7.264 3.25e-09 ***
x5            0.3179     0.1434   2.217   0.0315 *
```

```
---
Residual standard error: 0.9668 on 47 degrees of freedom
Multiple R-squared: 0.5293,     Adjusted R-squared: 0.5093
F-statistic: 26.43 on 2 and 47 DF,  p-value: 2.04e-08
```

К такой же модели ols2 можно прийти и «вручную», исключая поочередно параметры с наибольшими *p*-значениями.

Выполним теперь процедуру построения «всех возможных моделей» **all.possible.regressions**(), скрипт для которой представлен в разделе 7.7.

```
all.mod <- all.possible.regressions(dat = trdata, k = 6)
all.mod[order(all.mod$AIC), ]
```

Отсортировав все 64 возможные модели по критерию Акаике, обнаружим, что пошаговый метод выбрал не модель № 1 с наименьшим AIC, а расположенную ниже в позиции № 2 (то есть последовательные алгоритмы не всегда находят глобальный оптимум):

№	model	df.sse	p	SSE	MSE	R^2	R^2.adj	AIC	BIC
1	y ~ 1+x1+x4	47	2	43.92783	0.934635	0.529353	0.509325	197.1274	204.7755
2	y ~ 1+x1+x5	47	2	43.93306	0.934746	0.529297	0.509267	197.1334	204.7814
3	y ~ 1+x1+x4+x5	46	3	43.69985	0.949997	0.531796	0.50126	198.8672	208.4273
4	y ~ 1+x1+x3+x5	46	3	43.8223	0.952659	0.530484	0.499863	199.0071	208.5673
…									…
57	y ~ 1	49	0	93.33496	1.904795	0	0	230.8097	234.6338
…									…
63	y ~ 1+x4+x6	47	2	93.30361	1.985183	0.000336	-0.0422	234.7929	242.441
64	y ~ 1+x4+x5+x6	46	3	92.5297	2.011515	0.008628	-0.05603	236.3765	245.9366

Зато алгоритм прямого поиска, реализованный в функции **forward.sel**() из пакета packfor, позволил выбрать линейную МНК-модель ols3 с наименьшим значением AIC-критерия (№ 1 в списке выше) из всех возможных моделей, которые могут быть получены на основе сгенерированных данных:

```
library(packfor)
forward.sel(trdata$y, trdata[,-1])
Procedure stopped (alpha criteria):
       pvalue for variable 3 is 0.632000 (superior to 0.050000)
  variables order        R2     R2Cum  AdjR2Cum         F  pval
1        x1     1 0.4800657 0.4800657 0.4692337 44.319355 0.001
2        x4     4 0.0492872 0.5293529 0.5093254  4.921943 0.036
ols3 <- lm(y ~ x1 + x4, data = trdata)
```

Наконец, еще одна модель ols4 может быть получена методом «*forward selection*», представленным в пакете FWDselect и основанным на внешних критериях перекрестной проверки:

```
library(FWDselect)
qob = qselection(x = trdata[, -1],
                 y = trdata$y, qvector = c(1:6),
     method = "lm", criterion = "variance")
plot(qob)
```

При анализе зависимости ошибки кросс-проверки от сложности модели искомый минимум соответствует модели ols4 лишь с одним предиктором x1:

```
selection(x = trdata[, -1], y = trdata$y, q = 1,
          criterion = "variance",
          method = "lm", family = "gaussian")
Best subset of size q = 1 : x1
Information Criterion Value - variance : 0.7212502

ols4 <- lm(y ~ x1, data = trdata)
```

Рассчитаем для всех трех типов моделей средние квадраты ошибки прогноза на данных из тестовой выборки:

```
ols2.mspe <- mean( (gy - predict(ols2, newdata = gendata))^2 )
  1.222605

ols3.mspe <- mean( (gy - predict(ols3, newdata = gendata))^2 )
  1.225119

ols4.mspe <- mean( (gy - predict(ols4, newdata = gendata))^2 )
  1.319385
```

Построим теперь серию моделей методами гребневой и лассо-регрессии, а также регрессии на главные компоненты. Каждый раз в их число мы будем включать (исключительно в методических целях) модель, которая при соответствующих условиях сводится к полной МНК-модели ols1.

Модели с использованием регуляризации

Построим серию моделей гребневой регрессии, в которых параметр λ находится тремя методами: *а)* kHKB (аббревиатура, берущая начало от фамилий создателей метода – Hoerl, Kennard и Baldwin), *б)* kLW (происхождение неясно) и *в)* обобщенной перекрестной проверкой (GCV). Оптимальное значение λ для GCV будем находить методом последовательных приближений, задав предварительно диапазон возможных значений от 1 до 10:

```
library(MASS)
lmridge <- lm.ridge(y ~ x1 + x2 + x3 + x4 + x5 + x6, data = trdata,
                    lambda = seq(0, 10, 1))

lmridge$kHKB   # lambda, полученная методом kHKB
2.713368

lmridge$kLW    # lambda, полученная методом kLW
4.039776

lmridge$GCV
        0         1         2         3         4         5         6         7         8         9        10
0.02240937 0.02135273 0.02112040 0.02099452 0.02092176 0.02088160 0.02086311 0.02085967 0.02086701 0.02088224 0.02090335
```

```
# обобщенная перекрестная проверка показала, что оптимальное значение
# lambda лежит между 6 и 8.
# Выполним несколько итераций для уточнения этого значения:
lm.ridge(y ~ x1 + x2 + x3 + x4 + x5 + x6, data=trdata, lambda = seq(6, 8, 0.25))$GCV
     6.00      6.25      6.50      6.75      7.00      7.25      7.50      7.75      8.00
0.02086311 0.02086104 0.02085982 0.02085938 0.02085967 0.02086063 0.02086220 0.02086434 0.02086701
lm.ridge(y ~ x1 + x2 + x3 + x4 + x5 + x6, data=trdata, lambda = seq(6.5, 7, 0.1))$GCV
      6.5       6.6       6.7       6.8       6.9       7.0
0.02085982 0.02085955 0.02085941 0.02085938 0.02085947 0.02085967
lm.ridge(y ~ x1 + x2 + x3 + x4 + x5 + x6, data=trdata, lambda = seq(6.7, 6.9,
0.05))$GCV
     6.70      6.75      6.80      6.85      6.90
0.02085941 0.02085938 0.02085938 0.02085941 0.02085947
lm.ridge(y ~ x1 + x2 + x3 + x4 + x5 + x6, data=trdata, lambda = seq(6.75, 6.85,
0.01))$GCV
    6.75     6.76     6.77     6.78     6.79     6.80     6.81     6.82     6.83     6.84     6.85
0.02085938 0.02085938 0.02085938 0.02085938 0.02085938 0.02085938 0.02085939 0.02085939 0.02085940 0.02085941 0.02085941

# Построим четыре модели гребневой регрессии с различными значениями
# lambda: 2.7 (kHKB), 4.0 (kLW), 6.8 (GCV) и 0 (OLS):
(ridge1 <- lm.ridge(y ~ x1 + x2 + x3 + x4 + x5 + x6, data=trdata, lambda = 2.7))
                    x1          x2          x3          x4          x5          x6
 5.18630128  0.74553130 -0.08257060  0.35314248  0.13822561  0.17157641  0.02063876

(ridge2 <- lm.ridge(y ~ x1 + x2 + x3 + x4 + x5 + x6, data=trdata, lambda = 4.0))
                    x1          x2          x3          x4          x5          x6
 5.18630128  0.70928392 -0.04437111  0.33906950  0.13068938  0.16810009  0.01982505

(ridge3 <- lm.ridge(y ~ x1 + x2 + x3 + x4 + x5 + x6, data=trdata, lambda = 6.8))
                   x1         x2         x3         x4         x5         x6
 5.18630128 0.64312193 0.01142751 0.32437084 0.11900851 0.15938204 0.01863222

(ridge4 <- lm.ridge(y ~ x1 + x2 + x3 + x4 + x5 + x6, data=trdata, lambda =0))
                    x1          x2          x3          x4          x5          x6
 5.18630128  0.61072813 -0.50783517  0.90961393  0.15646842  0.18452901  0.03006552
```

Рассчитаем для всех гребневых моделей средние квадраты ошибки прогноза на данных экзаменационной выборки (к сожалению, метод **predict**() для функции **lm.ridge**() не предусмотрен):

```
ridge1.mspe <- mean( (gy - (gx %*% ridge1$coef + ridge1$ym))^2 )
1.143391

ridge2.mspe <- mean( (gy - (gx %*% ridge2$coef + ridge2$ym))^2 )
1.129384

ridge3.mspe <- mean( (gy - (gx %*% ridge3$coef + ridge3$ym))^2 )
1.109517

ridge4.mspe <- mean( (gy - (gx %*% ridge4$coef + ridge4$ym))^2 )
1.189688
```

При построении лассо-моделей оптимальное значение параметра фракционирования s будем находить путем минимизации C_p-критерия Мэллоу и с использованием перекрестной проверки:

```
library(lars)
M.lasso <- lars(as.matrix(trdata[, 2:7]), trdata[, 1], type = "lasso")
plot(las, plottype = "coefficients"); plot(las, plottype = "Cp")

# Из графика для Cp получили параметр фракционирования s = 0.56
las.Cp <- predict(M.lasso, as.matrix(trdata[,2:7]), s = 0.56,
         type = "coefficient", mode = "fraction")

# Теперь попробуем рассчитать его значение перекрестной проверкой
r <- cv.lars(as.matrix(trdata[,2:7]),trdata[,1])
(bestfrac <- r$index[which.min(r$cv)])
0.434

las.CV <- predict(M.lasso, as.matrix(trdata[, 2:7]), s = bestfrac,
         type = "coefficient", mode = "fraction")
las.CV  # Вывод параметров модели
$coefficients
        x1         x2         x3         x4         x5         x6
0.77168874 0.00000000 0.04019982 0.00000000 0.08479717 0.00000000

# При s = 1 лассо-модель сводится к полной модели OLS1:
las.OLS <- predict(M.lasso, as.matrix(trdata[, 2:7]), s = 1,
          type = "coefficients", mode = "fraction")
las.OLS
$coefficients
         x1          x2          x3          x4          x5          x6
 0.61072813 -0.50783517  0.90961393  0.15646842  0.18452901  0.03006552
```

Средние квадраты ошибки прогноза на данных экзаменационной выборки:

```
predlas <- predict.lars(M.lasso, newx = gx, type = "fit",
                        mode = "fraction", s = 0.434)
las1.mspe <- mean( (gy - predlas$fit)^2 )
1.187218

predlas <- predict.lars(M.lasso, newx = gx, type = "fit",
                        mode="fraction", s=0.56)
las2.mspe <- mean( (gy - predlas$fit)^2 )
1.194602

predlas <- predict.lars(M.lasso, newx = gx, type = "fit",
                        mode = "fraction", s = 1)
las3.mspe <- mean( (gy - predlas$fit)^2 )
1.198071
```

Регрессия на главные компоненты

На первом этапе найдем значения главных компонент с использованием функции **princomp**() и выясним, как распределена по главным компонентам объясняемая ими доля общей дисперсии:

```
tr.pca <- princomp(~ x1 + x2 + x3 + x4 + x5 + x6 , data=trdata)
summary(tr.pca); plot(tr.pca)
Importance of components:
                          Comp.1    Comp.2    Comp.3     Comp.4     Comp.5       Comp.6
Standard deviation     1.7368644 1.2559060 0.9880699 0.47281774 0.28653718 0.0637169144
Proportion of Variance 0.5130439 0.2682483 0.1660344 0.03801983 0.01396319 0.0006904499
Cumulative Proportion  0.5130439 0.7812922 0.9473265 0.98534636 0.99930955 1.0000000000
```

Выведем для анализа факторные нагрузки на исходные переменные:

```
loadings(tr.pca)
Loadings:
   Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
x1 -0.510 -0.210         0.747        -0.361
x2 -0.492 -0.311        -0.649        -0.483
x3 -0.532 -0.278                       0.797
x4  0.335 -0.611               -0.709
x5  0.318 -0.632         0.129  0.694
x6                0.994
```

Вторым шагом найдем значения главных компонент для данных из экзаменационной выборки и перейдем к производным переменным в пространстве главных компонент:

```
pcomp.tr <- predict(tr.pca)
pc1 <- pcomp.tr[,1]
pc2 <- pcomp.tr[,2]
pc3 <- pcomp.tr[,3]
pc4 <- pcomp.tr[,4]
pc5 <- pcomp.tr[,5]
pc6 <- pcomp.tr[,6]
```

Рассчитаем модель линейной регрессии по шести главным компонентам. Эта модель фактически сводится к полной МНК-модели ols1, построенной по исходным переменным, хотя и отличается от нее значениями коэффициентов:

```
pcr1 <- lm(y ~ pc1 + pc2 + pc3 + pc4 + pc5 + pc6)

summary(pcr1)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.186301   0.142052  36.510  < 2e-16 ***
pc1         -0.433444   0.081787  -5.300 3.77e-06 ***
pc2         -0.438356   0.113107  -3.876 0.000358 ***
pc3         -0.008835   0.143767  -0.061 0.951285
```

```
pc4          0.750803   0.300438   2.499 0.016352 *
pc5         -0.097052   0.495755  -0.196 0.845716
pc6          0.751324   2.229428   0.337 0.737754
---
Residual standard error: 1.004 on 43 degrees of freedom
Multiple R-squared: 0.5352,     Adjusted R-squared: 0.4703
F-statistic: 8.251 on 6 and 43 DF,  p-value: 5.747e-06
```

Отметим такой парадокс: мы получили модель с характеристиками (*RSE*, R^2, R^2_{adj}, F-критерий), идентичными ols1, хотя проблема мультиколлинеарности переменных теперь отсутствует. Максимальный коэффициент корреляции между преобразованными переменными не превышает 0.25 (для исходных переменных он доходил до 0.989):

```
cor(pcomp.gen)
            Comp.1      Comp.2       Comp.3       Comp.4      Comp.5      Comp.6
Comp.1 1.000000000  0.13364140  0.003681865  0.019540693  0.10729636  0.25095819
Comp.2 0.133641402  1.00000000  0.040613638 -0.196402431  0.04131167  0.03071873
Comp.3 0.003681865  0.04061364  1.000000000  0.006022093 -0.21084556  0.15956346
Comp.4 0.019540693 -0.19640243  0.006022093  1.000000000 -0.09829697 -0.02395636
Comp.5 0.107296363  0.04131167 -0.210845556 -0.098296972  1.00000000  0.08981856
Comp.6 0.250958194  0.03071873  0.159563465 -0.023956355  0.08981856  1.00000000
```

Заметим, что в модели pcr1 статистически значимыми оказались коэффициенты при переменных p_1, p_2 и p_4. Поэтому вторую модель, основанную на подмножестве информативных главных компонент, получим, запустив пошаговую процедуру, основанную на критерии AIC:

```
pcr2 <- step(pcr1)

summary(pcr2)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.18630    0.13759  37.694  < 2e-16 ***
pc1         -0.43344    0.07922  -5.472 1.78e-06 ***
pc2         -0.43836    0.10955  -4.001 0.000227 ***
pc4          0.75080    0.29100   2.580 0.013137 *
---
Residual standard error: 0.9729 on 46 degrees of freedom
Multiple R-squared: 0.5335,     Adjusted R-squared: 0.5031
F-statistic: 17.53 on 3 and 46 DF,  p-value: 9.894e-08
```

Для получения прогнозируемых значений регрессии на главные компоненты и оценки ошибок моделей необходимо привести значения переменных из экзаменационной выборки в пространство новых координат:

```
pcomp.gen <- predict(tr.pca, newdata = gendata)
genpcdata <- data.frame(pcomp.gen)
names(genpcdata) <- c("pc1", "pc2", "pc3", "pc4", "pc5", "pc6")
```

```
pcr1.mspe <- mean( (gy - predict(pcr1, newdata = genpcdata))^2 )
1.194602

pcr2.mspe <- mean( (gy - predict(pcr2, newdata=genpcdata))^2 )
1.192678
```

Результаты и некоторые выводы

Сведем результаты приведенных выше расчетов в одну таблицу и отсортируем ее по увеличению ошибки:

```
mspe.all <- c(
ols1.mspe,    # OLS (все переменные)
ols2.mspe,    # МНК (2 переменные, комбинированные шаговым методом)
ols3.mspe,    # МНК (2 переменные, найденные полным перебором)
ols4.mspe,    # МНК (1 переменная, найденная методом "forward selection")
ridge1.mspe,  # гребневая модель (lambda получена методом kHKB)
ridge2.mspe,  # гребневая модель (lambda получена методом kLW)
ridge3.mspe,  # гребневая модель (lambda получена методом GVC)
ridge4.mspe,  # гребневая модель с lambda = 0
las1.mspe,    # лассо (s найден кросс-проверкой)
las2.mspe,    # лассо (s найден с использованием критерия Cp)
las3.mspe,    # лассо с s = 1
pcr1.mspe,    # PCA-регрессия с полным набором главных компонент
pcr2.mspe)    # PCA-регрессия с 3 главными компонентами

sort(mspe.all)
```

№ п/п	Тип регрессии	Средний квадрат ошибки
1	Гребневая (lambda получена перекрестной проверкой)	1.109517
2	Гребневая (lambda-параметр по kLW)	1.129384
3	Гребневая (lambda-параметр по kHKB)	1.143391
4	Лассо (параметр *s* найден минимизацией C_p-критерия Мэллоу)	1.187218
5	Гребневая с lambda = 0	1.189688
6	PCA-регрессия с 3 главными компонентами	1.192678
7	Лассо с *s* = 1	1.194602
8	PCA-регрессия с полным набором главных компонент	1.194602
9	МНК (все переменные)	1.194602
10	Лассо (параметр *s* найден перекрестной проверкой)	1.198071
11	МНК (2 переменные, найденные комбинированым шаговым методом)	1.222605
12	МНК (2 переменные, найденные полным перебором)	1.225119
13	МНК (1 переменная, найденная методом «forward selection»)	1.319385

Отметим, что три модели (7–9) совершенно идентичны, поскольку все они сводятся к линейной МНК-модели с полным набором переменных. Идентичной

должна быть и модель 5, однако по неясной причине вычислительного характера оценки ее параметров незначительно отличаются.

На основе полученных результатов можно сделать следующие выводы (которые, впрочем, не следует считать обобщением на все возможные ситуации):

1. Использование гребневой регрессии для прогнозирования существенно превысило эффективность остальных моделей в условиях умеренной мультиколленеарности данных.
2. Лассо-регрессия и регрессия на главные компоненты заняли промежуточное положение.
3. Обычная линейная МНК-модель показала наихудшие результаты; особенно это касается так называемых «оптимальных» моделей с информативным набором параметров, выбранных на основе AIC (см. по этому поводу также введение к разделу 7.9).
4. Статистическая значимость коэффициентов, оцененная по *t*-критериям, а также величина таких критериев, как дисперсионное отношение Фишера, коэффициент детерминации и информационный критерий AIC, могут служить оценками адекватности линейной модели, предназначенной для *объяснения* изучаемого явления, но не имеют решающего значения при выборе наиболее эффективной модели для *прогнозирования*.

Глава 8

Обобщенные, структурные и иные модели регрессии

8.1. Модели сглаживания

Как обсуждалось в предыдущей главе, основная цель регрессионного анализа состоит в осуществлении разумной (то есть адекватной поставленной задаче) *аппроксимации* математического ожидания отклика $E[Y|x_1, x_2, ..., x_m]$ по обучающей выборке с помощью некоторой функции регрессии $f(x_1, x_2, ..., x_m)$. Если в центре внимания находится попытка «объяснить» внутренние механизмы изучаемого явления и оценить при этом сравнительную роль отдельных факторов, то используются параметрические методы регрессионного анализа. Такой подход часто предполагает, что искомая модель имеет некоторую функциональную форму $f(x_1, x_2, ..., x_m)$ и полностью описывается конечным набором параметров.

Типичным примером параметрической модели является полиномиальная регрессия (см. раздел 7.6), параметры которой представлены вектором коэффициентов β и случайной ошибкой ε. Здесь на *всем диапазоне* значений x осуществляется приближение данных *одной и той же функцией* (то есть полиномом заданной степени). На отдельных участках оси x отклонения от кривой могут оказаться значительными, но если величина общей ошибки приемлема в практических целях, мы принимаем *единую модель* на всей области определения x.

Естественно, что неверная спецификация выбранной функциональной формы единой модели может привести к серьезным, а часто и непредсказуемым искажениям при интерпретации результатов. Поэтому возникла идея: а нельзя ли для разных интервалов x строить не одну, а несколько самостоятельных моделей или как-то динамически подстраивать коэффициенты полинома при движении слева направо по оси x? Так оказалось, что прекрасной альтернативой классической регрессии является непараметрическое *сглаживание* («*smoothing*»).

Непараметрическое сглаживание применяется, когда по условию поставленной задачи исследования важна эффективная интерполяция значений зависимой переменной с минимальной ошибкой, а конкретный набор оцениваемых параметров модели самостоятельного интереса не представляет. Кроме того, при сглаживании практически невозможно проводить *экстраполяцию*, то есть распространять оценку модели за пределы выборочного диапазона x для прогноза значений отклика Y.

Любопытна точка зрения (см. прекрасные лекции C. Shalizi на сайте http://bit.ly/1zaAWWq), что сглаживание, классическая регрессия и даже выборочное среднее – лишь частные случаи применения к данным линейных сглаживающих фильтров («*smoothers*»). Все они задаются общей функцией регрессии для произвольного текущего значения x_0 в следующей форме:

$$\hat{y}(x_0) = \sum_{i=1}^{n} y_i w(x_i, x_0),$$

где x_i и y_i – набор из n значений выборочного ряда, в котором для каждого значения предиктора x_i известно точное (в пределах ошибки измерения) значение отклика y_i; x_0 – значение предиктора, для которого мы хотим найти прогнозируемую величину $\hat{y}(x_0)$; $w(...)$ – некоторая функция расчета весовых коэффициентов, оцениваемых по заданной выборке.

Действительно, многие классические статистические формулы являются частным случаем приведенного выше выражения. Например, приняв $w(x_i, x_0) = 1/n$ для x_0 и всех x_i, мы получим в итоге простое среднее для y. Оценки отклика по методу k-ближайших соседей («*k-nearest neighbors*») основаны на взвешивающей функции $w(x_i, x_0) = 1/k$, если между x_i и x_0 находится менее k наблюдений, и $w(x_i, x_0) = 0$ в противном случае. Наконец, простая модель линейной регрессии (без свободного члена) приобретает привычный вид, если принять $w(x_i, x_0) = (x_i/ns_x^2)x_0$, поскольку

$$\hat{y}(x_0) = \hat{b}x_0 = x_0 \sum_{i=1}^{n} x_i y_i / \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i (x_i / ns_x^2) x_0.$$

Процедуры сглаживания и другие методы непараметрической аппроксимации предоставляют в настоящее время целый набор гибких и эффективных средств анализа неизвестных регрессионных зависимостей. Варьируя формой функции $w(x_i, x_0)$, можно получить различные версии алгоритмов сглаживания: скользящей средней, экспоненциальной функцией, ядерной функцией, сплайнами, локальной полиномиальной регрессией LOESS или LOWESS, методом Хольта-Винтерса и др. (Hastie at al., 2009). При сглаживании данных уже не производится их «втискивание в прокрустово ложе» фиксированной параметризации, а сама модель регрессии имеет динамический функциональный характер, подстраиваемый под текущие значения предикторов.

Модели сглаживания отклика y на шкале независимой переменной x основаны на предположении, что для оценки значения $\hat{y}(x_0)$ в точке x_0 наиболее ценными являются наблюдения, находящиеся в окрестности $[x_0 - h, x_0 + h]$, то есть в «окне» шириной $2h$.

Метод *локальной регрессии* (LOESS – акроним от «*local* regression») представляет собой процедуру вычисления параметров линейной $y_i = \beta_0(x_0) + \beta_1(x_0)x_i + \varepsilon_i$ или полиномиальной (квадратичной) модели $y_i = \beta_0(x_0) + \beta_1(x_0)x_i + \beta_2(x_0)x_i^2 + \varepsilon_i$. Форма записи $\beta_j(x_0)$ означает, что коэффициенты β_j подстраиваются динамически для каждого текущего значения x_0. Подгонка такой модели методом наименьших квадратов осуществляется с учетом вектора весов $w(x_0, x_i, h)$, которые тем больше, чем ближе точки исходной выборки x_i располагаются по отношению к x_0:

$$\min \sum_{i=1}^{n} w(x_0, x_i, h)[y_i - \beta_0(x_0) - \beta_1(x_0)x_i]^2,$$

то есть при этом выполняется скользящее усреднение данных в окне некоторой ширины $2h$.

На рис. 121 показан результат (кривая зеленого цвета) сглаживания двумя различными моделями 100 пар точек (x_i, y_i), сгенерированных со случайной гауссовой ошибкой относительно синусоиды $Y = \sin(4X)$, которая представлена на графике кривой синего цвета (пример заимствован из Hastie et al., 2009):

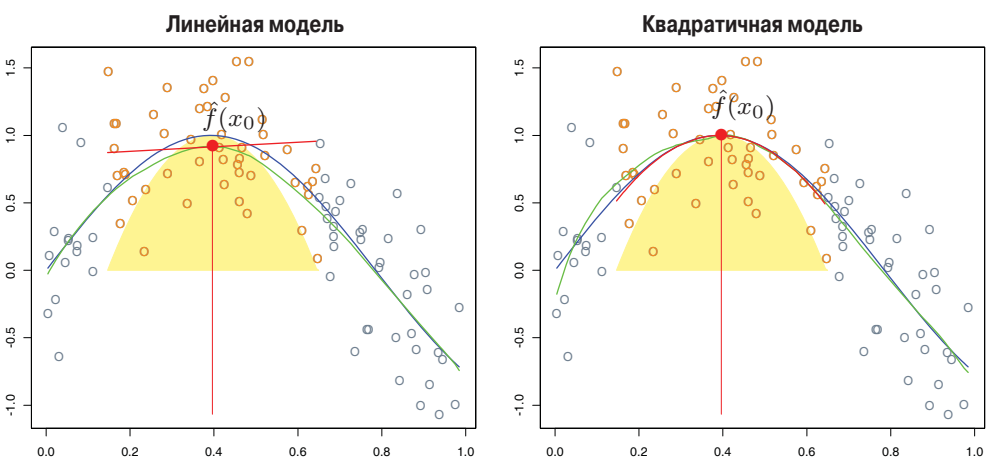


Рисунок 121

Сглаженные значения $\hat{y}(x_0)$ в точке x_0 (залита красным цветом) оцениваются по линейной или квадратичной модели, построенной по точкам, попавшим в «окно» (кружки красного цвета). Дополнительно при этом учитываются также веса w, причем наибольшие их значения приписываются наблюдениям, находящимся ближе к x_0. Аппроксимирующая кривая рассчитывается последовательно по мере скольжения «окна» слева направо по шкале x, в результате чего новая порция наблюдений включается в расчет локальных моделей взамен исключаемых старых.

Воспользовавшись такой процедурой, мы проделываем своеобразный «ядерный трюк» (Анатольев, 2009, http://bit.ly/1yLCzcP): нам уже нет необходимости задумываться над выбором базисной функции регрессии либо остерегаться, что не подтвердятся те или иные исходные предположения анализа. Нам достаточно вы-

брать степень локального полинома и ширину окна h (что, впрочем, тоже является непростой задачей).

На рис. 121 можно усмотреть важные отличия линейной сглаживающей модели от квадратичной: на концах области определения предпочтительнее более устойчивая линейная модель, тогда как при аппроксимации «горбов» (экстремумов) преимущество получает квадратичная модель.

Ширина окна на каждом шаге аппроксимации локальной регрессией обычно принимается равной стандартному отклонению задействованных точек (соответствующая ему гауссиана показана на вышеприведенном рисунке сегментом желтого цвета), то есть не является фиксированной. Поэтому важным параметром локальной регрессии является *степень сглаживания span = k / n*, которая соответствует доле (фракции) числа точек, используемых для подбора коэффициентов в окрестностях точки x_0, от общего объема выборки n (Cleveland, 1979).

Выбирая пример для демонстрации моделей сглаживания, мы постарались внести свою лепту (≅ 5 коп.) в рассмотрение проблемы всемирного потепления. Воспользуемся базой свободно распространяемых данных научной ассоциации *Berkeley Earth* и отыщем там массив наблюдений среднемесячной температуры в г. Москве и ее окрестностях (соответствующий файл с данными можно свободно скачать с сайта ассоциации – http://1.usa.gov/1HkCCDv, а также с сайта этой книги). Для удобства использования выделим ряд, охватывающий 260-летний (с 1753 по 2012 г.) временной период без пропусков, и преобразуем его в матрицу 260×12, где в столбцах будут представлены среднемесячные температуры за каждый год. Поскольку исходные данные содержат только отклонения (anomaly), то переведем матрицу в значения натуральных температур (базовые среднемесячные значения за период с января 1951 г. по декабрь 1980 г. представлены там же):

```
moscow.dat <- read.delim("TAVG_Moscow.txt", header = TRUE)
Year.list <- unique(moscow.dat$Year)
moscow.mat <- matrix(moscow.dat$Anomaly, nrow = 260, ncol = 12, byrow = TRUE)
Tbase <- c(-10.39, -9.28, -4.06, 4.80, 11.96, 16.16,
           17.76, 16.25, 10.77, 4.37, -1.86, -6.65)
moscow.nmat <- t(sapply(1:260,
function(i) moscow.mat[i,]+Tbase))
dimnames(moscow.nmat) = list(Year.list, 1:12)
head(moscow.nmat)
          1       2      3     4      5      6      7      8      9     10     11      12
1753 -10.168 -10.462 -1.864 4.772 12.496 16.405 17.899 16.014 11.101  4.793 -2.072 -12.491
1754 -10.737 -12.392 -4.202 7.699 12.498 17.522 17.826 14.832  9.764  4.518 -2.323  -6.163
1755 -11.605 -12.385 -4.582 4.812 13.596 18.586 19.175 15.056 10.549  4.626 -1.155  -8.790
1756  -7.467  -4.435 -3.497 5.885 12.295 18.091 18.455 14.325 10.830  3.344 -3.778  -9.951
1757 -12.616  -6.441 -3.169 6.169 12.907 20.278 22.321 17.593 12.540 -0.025 -2.585 -11.060
1758 -15.429 -10.831 -4.825 3.208 10.184 15.941 17.803 14.317  8.503 -1.073 -2.625  -8.531

# Сохраним матрицу данных для последующего использования
save(moscow.nmat, Year.list, file = "TAVG_Moscow.RData")
```

Проведем анализ динамики средней летней температуры (июнь–август). Попробуем сначала обойтись без моделей сглаживания: разобьем ряд на 10-летние отрезки, рассчитаем среднюю температуру за каждое десятилетие и построим ломаную кривую с уступами:

```
T.summer <- apply(moscow.nmat[, 6:8], 1, mean)

summary(T.summer)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  14.06   16.06   16.71   16.76   17.43   21.54

moscow.mat10 <- matrix(T.summer, nrow = 26, ncol = 10, byrow = TRUE)

plot(seq(1753, 2003, by = 10), rowMeans(moscow.mat10),
  type = "s", ylab = "Средняя температура, C",
  xlab = NA, col = "blue", lwd = 1.5, las = 2,
  cex.axis = 0.8, main = "Москва, июнь-август")
abline(h = mean(T.summer), col = 1, lty = 2, lwd = 1.2)
text(1830,16.85, adj = 0, "Общее среднее", cex = 0.75)
```

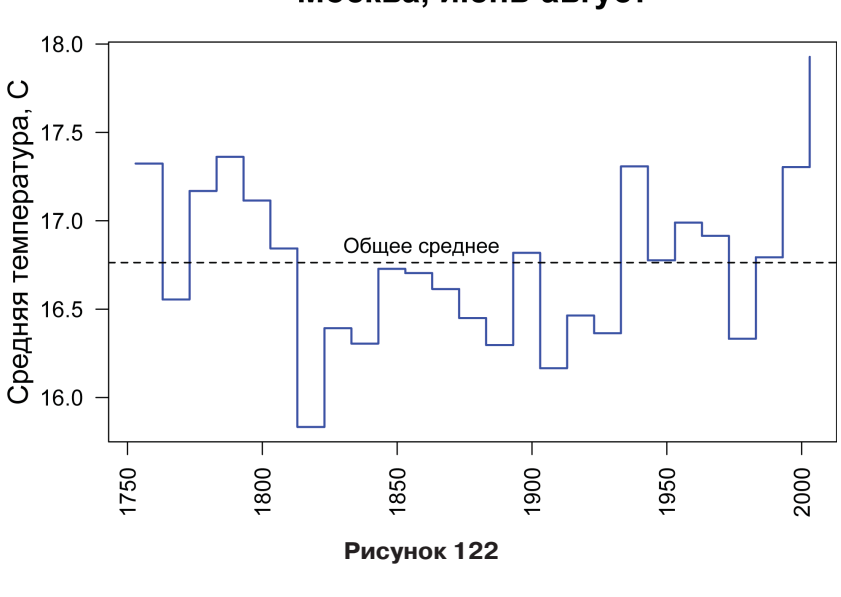


Рисунок 122

Выполним теперь аппроксимацию этих же данных с использованием локальной регрессии, реализованной в R функцией `loess()`. Чтобы подобрать адекватную степень сглаживания кривой, выполним построение серии моделей, варьируя величину span:

```
 # Создаем список значений параметра сглаживания:
spanlist <- c(0.15, 0.50, 0.8, 1.0)
plot(Year.list, T.summer, col = "grey", type = "l", xlab = "",
       ylab = "Средняя температура, C")
for (i in 1:length(spanlist)) {
  T.loess <- loess(T.summer ~ Year.list, span = spanlist[i])
  T.predict <- predict(T.loess)
  lines(Year.list, T.predict, col = i, lwd = 2)
}

legend("topleft", c(paste("span =",
       formatC(spanlist, digits = 2, format = "f"))),
              col = 1:length(spanlist), lwd = 2, bty = "n")
```

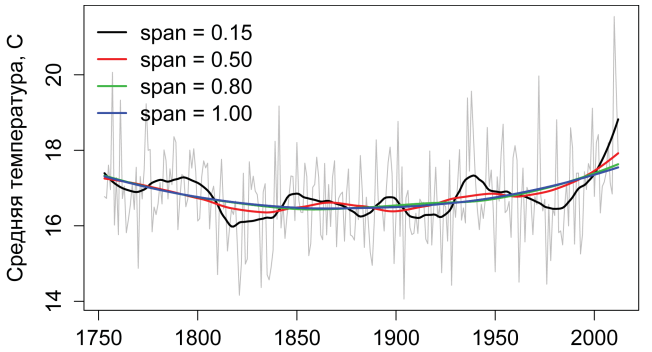


Рисунок 123

По рис. 123 хорошо видно, что после `span = 0.5` форма кривой практически стабилизируется. По результатам анализа нельзя сказать о значительном повышении летней температуры, хотя «кошмарное» лето 2010 г. внесло существенное возмущение и в модель, и в природу (в г. Тольятти, например, был практически уничтожен 300-летний сосновый лес).

Ядерная модель сглаживания

Ядерная модель сглаживания отклика Y на шкале независимой переменной X основывается практически на тех же исходных предпосылках, что и локальная регрессия. Основное отличие – в способе оценки значения $\hat{y}(x_0)$ в точке x_0. Например, для функции регрессии среднего значения отклика в окне $2h$ можно воспользоваться оценкой Надарайа-Уотсона (Nadaraya, Watson):

$$\hat{y}(t_0) = \sum_{i=1}^{n} y_i w(x_i, x_0, h) \Big/ \sum_{i=1}^{n} w(x_i, x_0, h),$$

где $w(x_1, x_0, h) = K(u)/h$, $u = (x_i - x_0)/h$, $K(u)$ – некоторая симметричная ядерная функция, интегрируемая на всем интервале варьирования данных. Если взять интеграл от $K(u)$ по всей области определения, которой может быть отрезок (обычно $[-1, 1]$) или вся числовая ось, то $\int K(u)du = 1$.

На рис. 124 (заимствован из Hastie et al., 2009) показана форма трех наиболее популярных ядерных функций:

1 – ядро Епанечникова $K(u) = 3(1 - u^2)/4$;

2 – «трижды кубическая функция» $(1 - |u|^3)^3$;

3 – функция нормального гауссова ядра $K(u) = (e^{-u^2/2}) / \sqrt{2\pi}$, которая, в отличие от двух предыдущих, не принимает значение 0 за пределами интервала $[-1, 1]$.

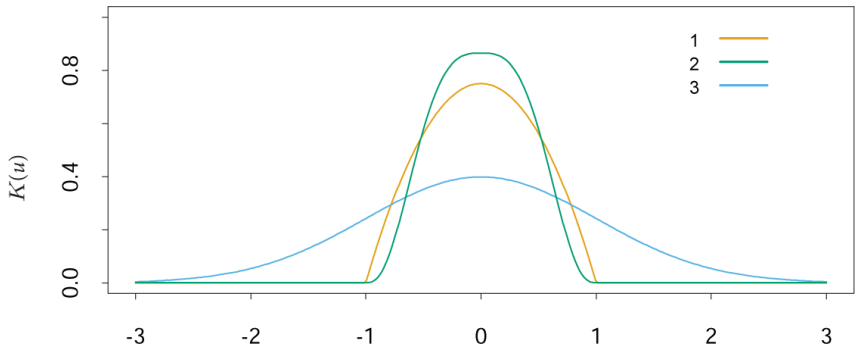


Рисунок 124

На рис. 125 (заимствован из Hastie et al., 2009) приведен пример сглаживания точек в окне, образованном ядром Епанечникова (область, закрашенная желтым цветом) относительно текущего значения x_0: синим показана теоретическая кривая, зеленым – кривая ядерного сглаживания, в подстройке весов которой каждый раз участвуют только точки, отмеченные красным цветом.

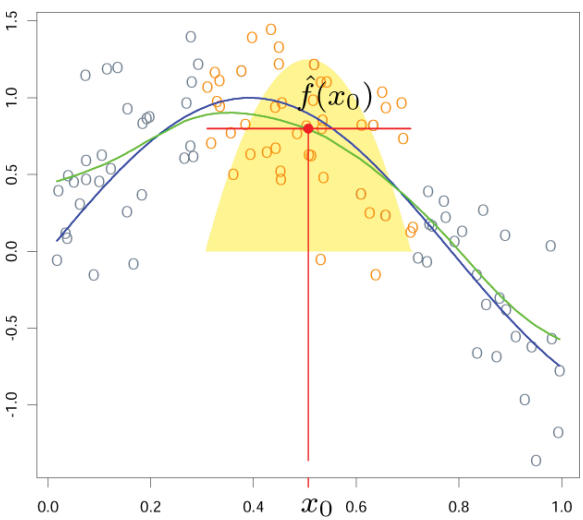


Рисунок 125

Как и в случае с локальной регрессией, для успешной аппроксимации нам необходимо выбрать ширину окна h и вид $K(u)$ ядерной функции (если нет иных предпочтений, разумно просто воспользоваться гауссовым ядром).

Роль параметра ширины окна h чрезвычайно велика. Слишком высокие значения h приводят к явлению сверхсглаженности: кривая вырождается в прямую и перестает отражать истинную структуру данных. При низких же значениях h модель начнет следовать за случайными флуктуациями в данных, и кривая будет выглядеть чересчур извилистой. Сложность проблемы здесь в том, что если попытаться минимизировать традиционный критерий качества модели – среднеквадратичную ошибку, то она устремится к нулю и выберет окно минимальной ширины с одним-единственным значением (то есть каждое наблюдение будет объясняться только им самим). Такое сглаживание нас, конечно, не устраивает, и поэтому разумной альтернативой поиска оптимального h будут внешние критерии – два варианта функции кросс-проверки (см. раздел 7.5):

$$CV_2(h)=\frac{1}{n}\sum\nolimits_{i=1}^{n}(y_i-\hat{y}_{-1}(x_i))^2 \quad \text{или} \quad CV_1(h)=\frac{1}{n}\sum\nolimits_{i=1}^{n}\left|y_i-\hat{y}_{-1}(x_i)\right|,$$

где $\hat{y}_{-1}(x_i)$ – оценка Надарайя-Уотсона в точке x_i, основанная на всех наблюдениях, кроме i-го. Оптимальная (в смысле процедуры скользящего контроля) ширина окна h^{CV} соответствует минимуму ошибки $CV(h)$ на независимом наборе данных.

Приведем функцию на языке R, которая рассчитывает этот оптимум для обоих критериев кросс-проверки, – параметр `band` функции ядерного сглаживания **`ksmooth`**():

```
# Функция оценки оптимальной ширины окна кросс-проверкой
bandcrossval <- function(x, y, nstep = 20,
                         bmax = 0, L1 = TRUE){
   if (bmax==0){bmax<- 1.5*sqrt(var(x))}
   bstep <- bmax/nstep
       n <- length(x)
        SSE<- c(1:nstep)*0
   for(i in 1:nstep){
         for(k in 2:(n-1)){
             xx <- c(x[1:(k-1)], x[(k+1):n])
             yy <- c(y[1:(k-1)], y[(k+1):n])
             kss<- ksmooth(xx, yy, "normal", ban = (i*bstep), x.points=x[k])
          if(L1==FALSE) {SSE[i] <- SSE[i]+(y[k]-kss$y )^2 }
          if(L1==TRUE) {SSE[i] <- SSE[i]+ abs(y[k]-kss$y )}
    }}
   k <- c(1:nstep)*bstep
  return(k[SSE == min(SSE, na.rm = TRUE)])
 }
```

Рассчитаем теперь среднюю зимнюю температуру в г. Москве для каждого астрономического года, начиная с 1753 г.:

```
T.winter <- apply(moscow.nmat[, c(1, 2, 12)], 1, mean)

summary(T.winter)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-15.050 -10.400  -8.925  -8.894  -7.331  -3.294

#  Сглаживание ядерной функцией и вывод графика:
bandcrossval(Year.list, T.winter, L1=FALSE)
[1] 67.67976  #  Квадрат ошибки при кросс-проверке

bandcrossval(Year.list, T.winter)
[1] 50.75982  #  Абсолютная ошибка при кросс-проверке

plot(Year.list, T.winter, ylab = "Средняя температура, C",
        xlab = "", col = "grey", type = "l",
        main = "Москва: январь, февраль, декабрь")
lines(ksmooth(Year.list, T.winter, "normal", bandwidth = 5), col = "gold")
lines(ksmooth(Year.list, T.winter, "normal", bandwidth = 15), col = "red")
lines(ksmooth(Year.list, T.winter, "normal", bandwidth = 25), col = "green")
lines(ksmooth(Year.list, T.winter, "normal", bandwidth = 50.8), col = "blue", lwd=2)
lines(ksmooth(Year.list, T.winter, "normal", bandwidth = 67.7), col = 1, lwd = 2)
legend("bottomright", lwd = c(1, 1, 1, 2, 2),
c("band = 5", "band = 15", "band = 25", "band = 50.8",
 "band = 67.7"), col = c("gold", "red", "green", "blue", 1))
```

Рисунок 126

Рассчитанные с помощью функции **bandcrossval**() оценки оптимальной ширины окна bandwidth равны 67.7 (квадрат ошибки) и 50.8 (абсолютная ошибка при кросс-проверке). Тенденция на повышение зимней температуры, начиная с 1950 г., выглядит с использованием этих параметров сглаживания вполне очевидной.

Сплайны

Аппроксимация сплайнами выполняет сглаживание в виде композиции из нескольких «кусочков» гладкой функции – как правило, полиномов. Свое происхождение термин «сплайн» («*spline*») ведет от длинных гибких линеек, которые использовались в прошлом в качестве лекал для проведения плавных кривых через заданные точки. Пусть отрезок оси абсцисс $[a, b]$ разбит на $(k + 1)$ частей точками $a_1 < ... < a_k$. Сплайном, или кусочно-полиномиальной функцией степени m с k *сопряжениями* в точках $a_1, ..., a_k$, называется функция $\varphi_j(x)$, которая на каждом интервале $(a_j, a_j + 1)$, где $j = 0, ..., k$, описывается алгебраическим полиномом $P_j(x)$ степени m. Коэффициенты полиномов $P_j(x)$ согласованы между собой так, чтобы выполнялись условия непрерывности функции $\varphi_j(x)$ и ее $(m - 1)$ производных в узлах сопряжений. Поскольку функция имеет непрерывно дифференцируемые переходы в точках сочленения, участки состыковываются между собой так, что итоговая кривая в целом также оказывается гладкой. Наиболее употребительны кубические сплайны с $m = 3$ (Wood, 2006).

В общем случае мерой близости некоторой кривой g к данным обучающей выборки является среднеквадратичная ошибка $MSE = \sum_{i=1}^{n}[y_i - g(x_i)]^2 / n$. Если не ограничивать количество сопряжений сплайна, то можно достичь хорошей аппроксимации данных, но получить кривую, имеющую много резких локальных изменений. И наоборот – плавность кривой обычно сопровождается увеличением ошибки сглаживания. Компромисс между этими двумя противоречивыми целями достигается за счет введения штрафа за нарушение плавности, равного интегралу от квадрата второй производной $SPL = \int(g''(x))^2 dx$. Если ввести параметр сглаживания λ, то функция оптимизации сплайна будет иметь вид $L(g, \lambda) = MSE + \lambda SPL$. Оптимальное значение λ находят с использованием бутстрепа или перекрестной проверки (Hastie et al., 2009; Maindonald & Braun, 2010).

В функции **smooth.spline**(), которая реализует в R аппроксимацию сплайнами, степень сглаживания задается параметром spar, который связан монотонной функцией с величиной λ и обычно (но не обязательно) изменяется на интервале [0; 1]. Можно либо указать его конкретное значение, либо задать аргумент cv = TRUE для нахождения оптимума λ перекрестной проверкой.

Рассмотрим в качестве примера сглаживание сплайнами временного ряда разности летней и зимней температур в г. Москве за последние 260 лет:

```
T.delta <- T.summer - T.winter
summary(T.delta)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  19.08   23.92   25.76   25.66   27.29   32.45
```

```
# Выводим сплайн-кривые для разных значений параметра spar:
plot(Year.list, T.delta, col = "grey", type = "l", xlab = "",
     ylab = "Средняя разность температур, C")
abline(lm(T.delta ~ Year.list), col = "grey55", lwd = 2)
lines(smooth.spline(Year.list, T.delta, spar = 0.3), col = "gold")
lines(smooth.spline(Year.list, T.delta, spar = 0.5), col = "red")
lines(smooth.spline(Year.list, T.delta, spar = 0.75), col = "green", lwd = 2)
lines(smooth.spline(Year.list, T.delta, spar = 1.0), col = "blue", lwd = 2)

T.spline <- smooth.spline(Year.list,T.delta, cv = TRUE)
T.spline$spar  # Параметр сглаживания найден кросс-проверкой
[1] 1.499964

lines(T.spline, lwd = 2) # Оптимальная кривая
legend("bottomleft", c("spar = 0.3", "spar = 0.5",
                       "spar = 0.75", "spar = 1", "spar = 1.5"),
                       lwd = c(1, 1, 2, 2, 2),
                       col = c("gold", "red", "green", "blue", "black"))
```

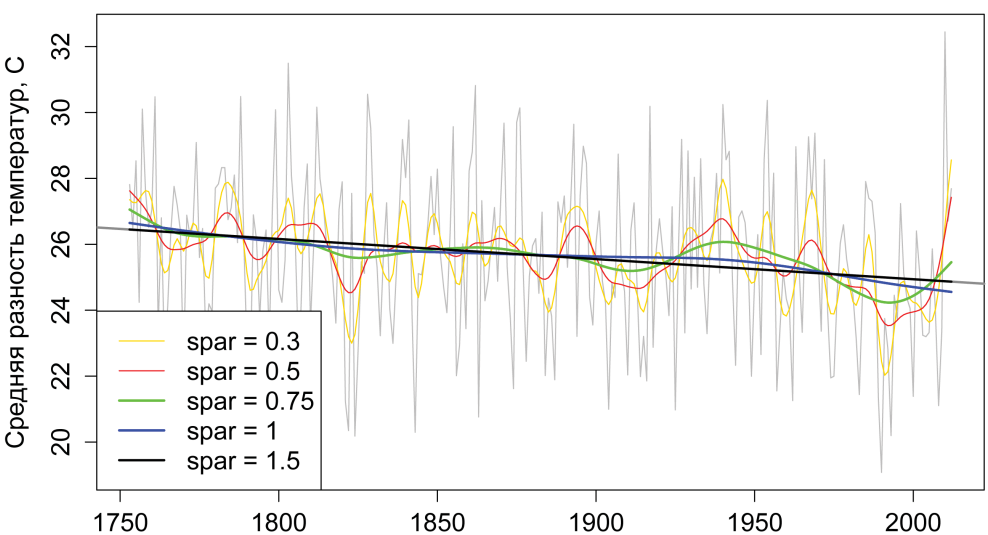


Рисунок 127

Интересно, что параметр сглаживания, найденный перекрестной проверкой, приводит нас к сплайну, совпадающему с линейной регрессией. Другими словами, аппроксимирующая линия обладает свойством нулевой кривизны и вполне ощутимо указывает на то, что климат становится менее континентальным. «А все-таки Она нагревается...» – так говорят о Земле современные Галилеи.

8.2. Обобщенные модели регрессии

Обобщенные модели расширяют класс общих линейных и нелинейных моделей регрессии, связывая зависимую переменную с предикторами посредством задаваемой *функции связи* («*link function*»), причем допускается наличие у отклика произвольного распределения, отличающегося от нормального. При этом охватываются широко используемые статистические модели, такие как линейная регрессия для откликов с нормальным распределением, логистические модели для факторов с двумя уровнями, лог-линейные модели для счетных данных, модели с дополняющим двойным логарифмированием для интервал-цензурированных данных выживания и многие другие статистические модели.

Обобщенные линейные модели (GLM, от «*generalized linear model*») и *обобщенные аддитивные модели* (GAM, от «*generalized additive model*») представляют собой частные случаи общего уравнения нелинейной регрессии

$$y = g^{-1}(\sum_{i=1}^{n} \beta_i q_i(x_i)),$$

где $g(y^{-1})$ является функцией связи, а $q_i(x_i)$ – произвольные функции нелинейного преобразования независимых переменных. Если принимаются некоторые условия тождественности этих функций, то мы получаем, например, следующие классы моделей:

- при $q_i(x_i) = x_i$ – обобщенную линейную модель $y = g^{-1}(\sum_{i=1}^{n} \beta_i x_i)$;
- при $g(y) = y$ – обобщенную аддитивную модель $y = \sum_{i=1}^{n} \beta_i q_i(x_i)$;
- при $g(y) = y$ и $q_i(x_i) = x_i$ – обычную линейную регрессию $y = \sum_{i=1}^{n} \beta_i x_i$.

Обобщенные линейные модели (McCullagh & Nelder, 1989; Venables & Ripley, 2002) дают возможность использовать различные функции связи, конкретный выбор которых зависит от природы случайного распределения зависимой переменной. Существует множество возможных сочетаний функций связи по отношению к закону распределения зависимой переменной, и некоторые из них могут оказаться одинаково приемлемыми для моделируемых данных. Поэтому при выборе функции связи можно руководствоваться априорными теоретическими соображениями или тем, какой вариант «на глаз» кажется наиболее подходящим.

Функция **glm**() для подгонки обобщенной линейной модели имеет вид:

```
glm(formula, family = family.generator, data = data.frame).
```

Единственный, по сравнению с функцией **lm**(), новый параметр `family` служит инструментом для описания типа модели и вызывает функцию, которая генерирует заданное распределение отклика. Генератор `family.generator` имеет вид:

<наименование распределения>(`link` = <наименование функции связи>).

Рассмотрим некоторые из комбинаций, часто задаваемых параметром `family`:

1. Нормальное распределение: `family = gaussian`. Если мы располагаем выборочными значениями переменных y и x, измеренными в непрерывных

шкалах, и нет оснований отказываться от предположения о нормальном распределении остатков, то функция преобразования является тождеством $g(y) = y$ (то есть `link = identity`). В некоторых случаях оказывается полезным нормализующее преобразование отклика путем логарифмирования (`link = log`) или получения обратной величины (`link = inverse`).

2. Биномиальное распределение: `family = binomial`. Применимо только к переменным, представляющим бинарный отклик или число регистрируемых событий. Пробит- и логит-регрессия основаны на функциях связи `link = probit` и `link = logit`; в моделях дожития для интервально цензурированных данных можно использовать также дважды логарифмическую функцию связи (`link = cloglog`).
3. Гамма-распределение: `family = Gamma`. Применимо к переменным с положительными значениями масштаба (`scale`) со скосом в направлении больших значений. Если выборочное значение меньше или равно 0, то такие наблюдения должны быть удалены.
4. Распределение Пуассона: `family = poisson`. Может быть представлено в виде числа событий, зарегистрированных за определенный период времени или на определенном участке пространства, и применимо к переменным с неотрицательными целыми значениями. К распределению Пуассона часто применяют логарифмическую функцию связи (`link = log`) для подгонки к счетным данным (частотам событий) лог-линейной модели (то есть создают непрерывный «суррогат» исходного распределения дискретной переменной).
5. Квазиправдоподобные генераторы: `family = quasi`. В данном случае вместо точного распределения отклика указываются функция связи и форма зависимости дисперсии от среднего (например: `variance = "1/mu^2"`, `variance = "constant"`).

При использовании обобщенных аддитивных моделей (Hastie & Tibshirani, 1990; Wood, 2006) вида $g(y) = \beta_0 + \sum_{i=1}^{p} q_i(x_i) + \varepsilon$ в качестве функции связи $g(y)$ чаще всего ограничиваются тождеством $g(y) = y$, но, например, в случае биномиального распределения широкое применение находят и логит-модели $g(y) = \log(y/(1-y))$.

Аддитивные модели являются более гибким инструментом аппроксимации, чем линейные модели, поскольку вид лучшей преобразующей функции $q_i(x_i)$ для каждой переменной может быть предварительно установлен по результатам мета-анализа или подобран в ходе специальных исследований (предположим, мы знаем, что перед включением показателя семейного дохода в эконометрическую модель из него следует извлекать кубический корень). Если никаких исходных предпосылок о виде $q_i(x_i)$ нет, можно использовать любую гладкую функцию, например $q(x) = (1/x)$, логарифмическую функцию $q(x) = \log(x)$ и т. д. Однако наиболее эффективно применение описанных выше непараметрических моделей сглаживания локальной регрессией или сплайнами. В сущности, GAM часто представляет собой лишь средство обобщения частных функций в единое целое без каких-либо параметрических предположений относительно формы q_i.

Методы подгонки аддитивных моделей реализованы в R в функции **gam**() из пакета mgcv, которая позволяет получать «экономные» модели с оптимальным числом степеней свободы, используя встроенный механизм перекрестной проверки. При необходимости в качестве многомерных сглаживающих функций могут использоваться изотропические сплайны или результаты их преобразования в тензорном пространстве. Синтаксис функции вполне обычен:

```
gam(formula, family = gaussian, data = data.frame,…),
```

но включает еще два десятка дополнительных параметров, объяснение которых потребовало бы пересказа монографии Wood (2006).

Метод наименьших квадратов, реализованный в **lm**(), оптимален лишь при нормальном законе распределения остатков ε_1, ..., ε_n и не подходит для оценивания параметров моделей GLM и GAM. К асимптотически оптимальным значениям неизвестных параметров для широкого класса вероятностных моделей приводит уже упоминавшийся выше универсальный *метод максимального правдоподобия* (MLE, от «*maximum likelihood estimation*»).

Принцип максимального правдоподобия состоит в том, что в качестве «наиболее правдоподобного» значения параметров берут значения Θ, максимизирующие вероятность получить при n опытах имеющуюся выборку $X = (x_1, ..., x_n)$. Предположим, что нам известен закон распределения дискретной случайной величины X, определяемый одним параметром Θ, но неизвестно численное значение этого параметра. Пусть $p(x_i, \Theta)$ – вероятность того, что в результате испытания величина X примет значение x_i. *Функцией правдоподобия* случайной величины X называют функцию аргумента Θ, определяемую по формуле

$$L(x_1, x_2, ..., x_n; \Theta) = p(x_1, \Theta)p(x_2, \Theta)...p(x_n, \Theta).$$

Тогда в качестве искомой точечной оценки параметра Θ принимают такое его значение, при котором функция правдоподобия достигает максимума: $\theta_{MLE} = \text{argmax}L(x|\theta)$, $\theta \in \Theta$.

Для поиска этого экстремума необходимо найти частную производную функции правдоподобия и приравнять ее к нулю: $\partial L(x, \theta_o)/\partial\theta = 0$. Если вторая производная в критической точке отрицательна, то это – точка максимума, соответствующая искомому значению параметра. Если нам необходимо найти k параметров θ, то формируется и решается система из k уравнений правдоподобия.

Например, в стандартном методе наименьших квадратов находят минимум суммы квадратов разностей между эмпирическими и полученными по модели значениями отклика $\sum_{i=1}^{n}(y_i - \mu_i)^2$. Чтобы найти наиболее правдоподобные значения μ_i, необходимо проанализировать функцию максимального правдоподобия L. Если предположить, что функция плотности распределения каждого y_i подчиняется нормальному закону, а параметр дисперсии σ постоянен для всех наблюдений, то

$$L(\mu, y) = \sum_{i=1}^{n}\{-(y_i - \mu_i)^2 / 2\sigma^2 - \ln(\sqrt{2\pi\sigma}\},$$

а наиболее правдоподобные MLE-оценки μ_i будут идентичны полученным при помощи МНК.

Если мы проводим регрессионный анализ в рамках предположений классической модели, то можно записать функцию правдоподобия в терминах остатков:

$$L(\mathbf{e}|\theta; \sigma^2) = -n\ln(2\pi)/2 - n\ln(\sigma^2)/2 - [(\mathbf{Y} - \mathbf{X}\theta)^T(\mathbf{Y} - \mathbf{X}\theta)]2\sigma^2.$$

Однако другие функции вероятности распределения остатков (например, функция Пуассона) приводят к другим выражениям для функции максимального правдоподобия. Тогда для нахождения искомых параметров μ_i (а следовательно, и коэффициентов моделей) необходимо воспользоваться итеративными нелинейными процедурами оптимизации функции L (Справочник ..., 1989).

Поскольку функции $L(...)$ и $\ln L(...)$ достигают максимума при одном и том же значении θ, с вычислительной точки зрения удобнее искать максимум логарифмической функции правдоподобия $\ln L$ («*log-likelihood*»), то есть

$$LL = \ln\{L(\theta \mid x, y)\} \rightarrow \max.$$

Для формальной оценки адекватности моделей GLM или GAM, а также для сравнения качества нескольких построенных моделей обычно используется такая статистика, как девианс («*deviance*»), непосредственно вытекающая из оценок максимального правдоподобия. *Девианс* G^2 для анализируемой модели M, основанной на наборе данных y, определен как

$$G^2 = -2\,(LL_{\mathrm{M}} - LL_{\mathrm{S}}),$$

где LL_{S} – максимум логарифма функции правдоподобия для полной, или «насыщенной» («*saturated*»), модели S, которая содержит так много параметров θ_{S}, чтобы по возможности точно восстановить значения y выборочных данных (Singer & Willett, 2003, p. 122). На практике вероятность правдоподобия такой модели S, точно воспроизводящей данные, обычно близка к 1, а логарифм максимума ее функции правдоподобия равен 0. При нормальном законе распределения такая модель регрессии S включает n параметров, а результаты ее прогноза совпадают с выборочными наблюдениями.

Для тестируемой модели M величина LL_{M} всегда отрицательна, и тогда $LL_{\mathrm{S}} > LL_{\mathrm{M}}$ для любых M. Таким образом, оценка статистической величины девианса G^2 по своему смыслу оказывается идентичной знакомой нам сумме квадратов остатков классической модели регрессии.

Другой экстремальной моделью является нуль-модель с одним параметром, который просто соответствует среднему значению зависимой переменной. Как правило, девианс нуль-модели превышает девианс тестируемой модели M, а распределение их разницы может быть аппроксимировано χ^2-распределением. Таким образом, мы можем сделать вывод о значимости модели M, если разность девиансов нуль- и M-моделей по абсолютной величине достаточно велика и статистически превышает значение 0.

8.3. Модели пробит- и логит-регрессии

Зачастую зависимая переменная Y носит альтернативный (дихотомический, или бинарный) характер: «выражен или нет эффект от воздействия лекарственного препарата», «обнаружен или нет биологический вид в отобранной пробе», «голосование за или против на референдуме» и т. д. Если этот показатель выступает в качестве зависимой переменной, то наш анализ сводится, по сути, к расчету вероятности P наступления события или наличия требуемого свойства. На величину отклика P может оказывать влияние один или несколько факторов X, таких как «количество введенного лекарственного препарата», «температура водоема», «интенсивность мультимедийной нагрузки на электорат» и т. д.

Сформулируем условия и способ построения линейной модели для бинарного отклика, которая позволила бы количественно оценить степень влияния независимого предиктора x и/или «объяснить» внутренние механизмы моделируемого процесса. Возникает вопрос: можем ли мы использовать для этого обычную модель регрессии, основанную на методе наименьших квадратов: $P = p(Y = 1) = \beta_0 + \beta_1 x + \varepsilon$?

Многочисленные экспериментальные данные показывают, что кривая зависимости $P = f(x)$ часто носит ярко выраженный нелинейный S-образный характер, асимптотически приближаясь к значениям $P_0 = 0$ и $P_1 = 1$, что напоминает интегральную функцию плотности нормального распределения. При этом обычно делается предположение, что доля альтернатив $p(Y = 1)$ является случайной величиной, подчиненной биномиальному закону с параметрами (n, π). Очевидно, что при расчете такой модели методом МНК нарушаются основные предпосылки линейной регрессии и возникают следующие проблемы, связанные с особенностями биномиального распределения отклика:

1) *проблема наличия гетероскедастичности*: поскольку для бинарной переменной $\sigma^2 = pq$, то итоговую дисперсию вероятности для модели с одним предиктором можно представить как $\sigma^2_{\varepsilon_i} = (\beta_0 + \beta_1 x_i)(1 - \beta_0 - \beta_1 x_i)$, то есть с изменением x меняется и дисперсия;
2) *проблема ненормального распределения ошибок*: остатки $\hat{\varepsilon}_i = 1 - \beta_0 - \beta_1 x_i$ при $y_i = 1$ и $\hat{\varepsilon}_i = -\beta_0 - \beta_1 x_i$ при $y_i = 0$, то есть вариация $\hat{\varepsilon}$ является ограниченной и ненормально распределенной, а значит, стандартная ошибка регрессии и t-статистика неверны;
3) *проблема правильной спецификации модели*: предсказываемые вероятности должны «уместиться» в интервале [0, 1], а обычная регрессионная модель может вполне предсказывать значения далеко за пределами этого интервала.

Логика преодоления ограничений классической линейной модели состоит в том, чтобы трансформировать значения частот P в некоторые непрерывные количественные величины Z, лежащие в диапазоне $(-\infty, +\infty)$, а затем полученные значения регрессии преобразовать обратно в вероятности. Как было показано в предыдущем разделе, такое преобразование выполняется с использованием

функции связи, вид которой выбирают, исходя из практических соображений и с учетом предположений о законе распределения отклика.

Пробит-регрессия для моделирования зависимости «доза–эффект»

Рассмотрим токсикологический эксперимент, в котором под величиной эффекта P понимается доля погибших подопытных животных за фиксированное время в выборке объемом n при внешнем вредном воздействии с интенсивностью D. Можно показать (Справочник.., 1989, т. 1, с. 337), что при воздействии дозы D вероятность гибели животного, случайно отобранного из всей совокупности, равна

$$P(x) = \frac{1}{\sqrt{2\pi}} \int_{\infty}^{x} \{e^{-0.5(y-\mu)^2/\sigma^2}\} dy = \Phi\left(\frac{x-\mu}{\sigma}\right),$$

где $x = \ln(D)$, $\Phi(...)$ – интегральная функция плотности стандартного нормального распределения $N(\mu, \sigma)$.

Если принять $\beta_0 = -\mu/\sigma$ и $\beta_1 = 1/\sigma$, то получим

$$P = \Phi(\beta_0 + \beta_1 x + \varepsilon),$$

или

$$\Phi^{-1}(P) = \beta_0 + \beta_1 x + \varepsilon.$$

Таким образом, с использованием обратной функции $\Phi^{-1}(...)$ величина $\hat{P}(x)$ определяется единственным образом по вычисленному значению $\hat{\beta}_0 + \hat{\beta}_1 x$. В статистической среде R значения прямой и обратной функции $\Phi(...)$ по заданной вероятности легко найти с использованием функций **qnorm**(p) и **pnorm**(q):

```
p = c(0.05, 0.1, 0.25, 0.4, 0.5, 0.6, 0.75, 0.9, 0.95)
q = qnorm(p)

rbind(q, p = pnorm(q))
       [,1]      [,2]       [,3]       [,4] [,5]      [,6]      [,7]     [,8]     [,9]
q -1.644854 -1.281552 -0.6744898 -0.2533471  0.0 0.2533471 0.6744898 1.281552 1.644854
p  0.050000  0.100000  0.2500000  0.4000000  0.5 0.6000000 0.7500000 0.900000 0.950000
```

Значение квантиля $q = \Phi^{-1}(P)$, используемое в моделях в качестве аргумента функции связи, называют *пробитом* (буквально «*probit*» означает «*probability unit*», то есть «единица вероятности»). Как показано выше, $probit(p = 0) = -\infty$, $probit(p = 0.05) = -1.645$, $probit(p = 0.5) = 0$, $probit(p = 0.95) = 1.645$, $probit(p = 0) = +\infty$. Во времена, когда графики строили на миллиметровой бумаге, во избежание отрицательных величин к значению квантиля добавляли число 5 и *пробитом* называли полученный результат. Сейчас эта операция считается излишней.

Таким образом, для получения прогнозируемой величины эффекта P достаточно оценить параметры $\hat{\beta}_0$ и $\hat{\beta}_1$ модели простой регрессии на величину дозы D, пос-

ле чего найденное пробит-значение трансформировать в вероятность *P*. Все эти операции выполняет функция **glm**() при использовании следующего общего синтаксиса:

```
glm(formula = <формула>, data = <таблица с данными>,
    family = binomial(link = "probit")),
```

где family определяет закон распределения отклика, а описанный выше параметр link – вид функции связи (см. раздел 8.2).

Рассмотрим построение пробит-модели на следующем примере (исследования лаборатории герпетологии и токсинологии ИЭВБ РАН, http://bit.ly/1PTCoXR). Группам мышей одинаковой массы (20±1.0 г) подкожно вводили возрастающие дозы ядов гадюк из разных местообитаний и через сутки подсчитывали число погибших и выживших животных. Группы формировались отдельно из мышей-самцов и мышей-самок. Поставим задачу построить зависимость «доза–эффект», оценить величину средней полулетальной дозы LD_{50} и проверить гипотезы: *а*) об отсутствии различий в токсичности яда гадюк из разных регионов и *б*) об одинаковой чувствительности самцов и самок к действию яда.

```
# Загружаем данные из файла (с сайта книги)
df <- read.table(file = "Гадюки.txt", header = TRUE, sep = "\t")
head(df)
   пол местооб доза численность погибло выжило
1    M    Перм 0.15           3       0      3
2    M    Перм 0.30           3       0      3
3    M    Перм 0.75           3       1      2
14   F    Перм 1.48           3       1      2
15   F    Перм 2.23           3       3      0
16   F    Перм 2.97           3       3      0

levels(df$местооб)
[1] "Моск"  "НжНов" "Новг"  "Перм"  "Самар" "Татар" "Хвал"
```

При расчете пробит-регрессии в R отклик можно задавать в виде матрицы, в которой первый столбец содержит число погибших, а второй – число выживших особей:

```
# Строим модель на всем имеющемся материале
dead <- cbind(df$погибло, df$выжило)
M1 <- glm(dead ~ доза, family = binomial(link = "probit"), data = df)

summary(M1)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.3540     0.2262  -10.40   <2e-16 ***
доза          1.5181     0.1364   11.13   <2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
    Null deviance: 335.48  on 124  degrees of freedom
Residual deviance: 112.00  on 123  degrees of freedom
AIC: 190.61
Number of Fisher Scoring iterations: 6
```

Мы получили пробит-модель вида

$$probit(P) = -2.35 + 1.52D$$

со статистически значимым коэффициентом угла наклона $\hat{\beta}_1$ (p-значение << 0.001). Формат вывода результатов отличается от формата, получаемого с помощью знакомой нам функции **lm**(), лишь в деталях, за которыми тем не менее скрываются серьезные математические отличия. Приведенные выше результаты расчетов включают:

- Null deviance – девианс «пустой» модели, не включающей ни одного параметра, кроме β_0;
- Residual deviance – остаточный девианс, который косвенно соответствует дисперсии в данных, оставшейся необъясненной после включения в модель предиктора доза.

Смысл информационного критерия AIC был описан нами в разделе 7.5. Наконец, в последней строке указываются итерации, выполненные перед схождением алгоритма, по которому вычисляются параметры модели (их число не должно быть слишком велико).

Проведем теперь оценку статистической значимости по критерию χ^2 отличий девианса полученной логит-модели от девианса нуль-модели без предикторов:

```
M0 <- glm(dead ~ 1, family = binomial(link = "probit"), data = df)

anova(M1, M0, test = "Chisq")
Analysis of Deviance Table
Model 1: dead ~ доза
Model 2: dead ~ 1
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1       123     112.00
2       124     335.48 -1  -223.48 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Разность между девиансами моделей 1 и 2 весьма велика, что определяет вывод о высокой статистической значимости полученной регрессионной зависимости.

Построим график зависимости «доза–эффект» для модели M1. Если пробиты пересчитать в вероятности, то получим типичную S-образную кривую (рис. 128):

```
df.plot = data.frame(df, p = M1$fit,
        пробит = predict(M1, type =  "link"),
        эффект = df$погибло/(df$погибло+df$выжило))
df.plot = df.plot[order(df.plot$доза),]
```

```
plot(df.plot$доза, df.plot$p, type = "l", lwd = 2,
            xlab = "Доза",ylab = "Доля умерших")
points(df.plot$доза, df.plot$эффект, cex = 0.7, pch = 23 +
    as.numeric(df.plot$пол), bg = 1 + as.numeric(df.plot$пол))
legend("bottomright", c("Самки","Самцы"), pt.bg = 2:3,
            pch = 24:25, cex = 0.7)
```

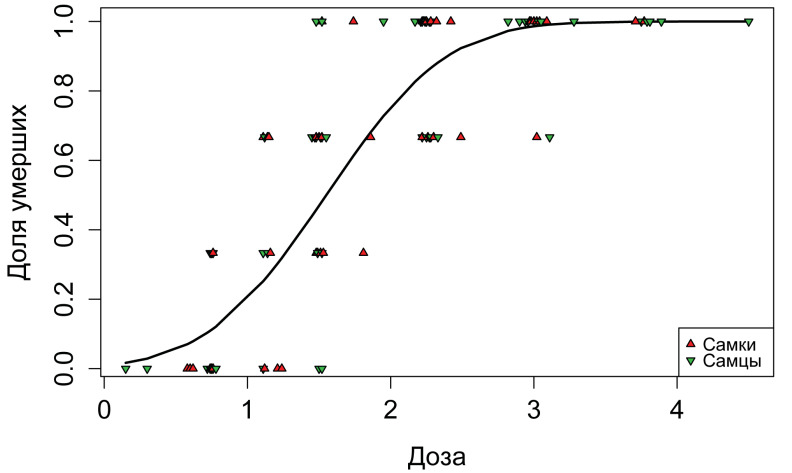


Рисунок 128

Если рассматривать на графике распределение экспериментальных точек относительно модельной кривой, то визуально ошибка аппроксимации достаточно велика. Усложним модель и включим в качестве дополнительной переменной фактор пол:

```
M2b <- glm(dead ~ доза * пол,
          family = binomial(link = "probit"), data = df)

summary(M2b)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.40332    0.32609  -7.370 1.71e-13 ***
доза         1.54307    0.19659   7.849 4.19e-15 ***
полМ         0.09592    0.45256   0.212    0.832
доза:полМ   -0.04833    0.27295  -0.177    0.859
```

В новой пробит-модели M2b свободный член (Intercept) и коэффициент при переменной доза корректируются в зависимости от пола животных, то есть к ним добавляются значения 0.096 и –0.048 соответственно, но только при условии, что выборка мышей состоит из самцов. Однако эти приращения столь малы, что статистически значимо не отличаются от 0 (p-значения, оцененные по z-критерию для этих коэффициентов, равны 0.832 и 0.859 соответственно).

Вывод об отсутствии различий в чувствительности самцов и самок к действию яда гадюк подтверждает и сравнение девиансов двух моделей M1 и M2b: после добавления в качестве предиктора пола дисперсия, оставшаяся необъясненной, снижается статистически незначимо (Resid. Deviance падает лишь с 112.0 до 111.96):

```
anova(M1b, M2b, test = "Chisq")
Analysis of Deviance Table
Model 1: dead ~ доза
Model 2: dead ~ доза * пол
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1       123     112.00
2       121     111.96  2 0.046683    0.9769
```

Немного изменив объект formula, мы можем получить ту же модель, но с иным составом коэффициентов:

```
M2a <- glm(dead ~ пол/доза - 1,
          family = binomial(link = "probit"), data = df)
summary(M2a)
Coefficients:
          Estimate Std. Error z value Pr(>|z|)
полF       -2.4033     0.3261  -7.370 1.71e-13 ***
полM       -2.3074     0.3138  -7.353 1.94e-13 ***
полF:доза   1.5431     0.1966   7.849 4.19e-15 ***
полM:доза   1.4947     0.1894   7.894 2.93e-15 ***
```

Таким образом, подогнав одну модель, мы получим два уравнения регрессии, отдельно для каждого пола:

$probit(P) = -2.40 + 1.52D$ – *модель для самок*;
$probit(P) = -2.31 + 1.49D$ – *модель для самцов*.

Рассчитаем теперь такой важный показатель токсикометрии, как среднеэффективная доза LD_{50} (то есть доза, после введения которой в генеральной популяции мышей погибнет 50% особей – аббревиатура от «*lethal dose*, 50%»). Для этого используем функцию **dose.p**() из пакета MASS. Отметим, что полученные оценки значений LD_{50} для самцов и самок практически не различаются, подтверждая сделанный выше вывод об отсутствии влияния пола на чувствительность в отношении яда:

```
library(MASS)

# На всем имеющемся материале:
dose.p(M1b, p = 0.5)
             Dose         SE
p = 0.5: 1.550653 0.05523258

# Отдельно для самок и самцов
dose.p(M2a, c(1, 3))           ## Самки
             Dose         SE
p = 0.5: 1.557488 0.0768487
```

```
dose.p(M2a, c(2, 4))           ## Самцы
             Dose         SE
p = 0.5: 1.543672 0.07934212
```

Построим теперь пробит-модель, учитывающую региональную изменчивость токсичности яда гадюк, и проверим при помощи анализа девиансов, является ли введение фактора `местооб` статистически значимым в смысле уменьшения ошибки модели:

```
M3b <- glm(dead ~ доза * местооб,
          family = binomial(link = "probit"), data = df)

anova(M1, M3b, test = "Chisq")
Analysis of Deviance Table
Model 1: dead ~ доза
Model 2: dead ~ доза + местооб
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1       123    112.003
2       111     71.045 12   40.958  4.98e-05 ***
```

Девианс остатков уменьшился с 112 до 71.0, что является статистически значимым по критерию χ^2 (p-значение равно 0.00005). Следовательно, можно сделать вывод: *яд гадюк, отловленных в разных регионах, имеет разную токсичность*.

Удобно ту же модель представить в виде набора коэффициентов индивидуальных регрессий отдельно для каждого региона (общая идентичность моделей `M3a` и `M3b` подтверждается, например, совпадением девианса остатков):

```
M3a <- glm(dead ~ местооб/доза - 1,
          family = binomial(link = "probit"), data = df)

summary(M3a)
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
местообМоск        -5.6921     2.2650  -2.513 0.011967 *
местообНжНов       -3.5051     0.8906  -3.935 8.30e-05 ***
местообНовг        -2.6839     0.6602  -4.065 4.79e-05 ***
местообПерм        -2.4000     0.5399  -4.445 8.79e-06 ***
местообСамар       -2.9431     0.8417  -3.496 0.000471 ***
местообТатар       -4.0693     0.9972  -4.081 4.49e-05 ***
местообХвал        -1.4277     0.5438  -2.625 0.008656 **
местообМоск:доза    4.2497     1.6054   2.647 0.008117 **
местообНжНов:доза   2.0651     0.4966   4.159 3.20e-05 ***
местообНовг:доза    1.6773     0.4072   4.119 3.80e-05 ***
местообПерм:доза    1.2775     0.2728   4.683 2.83e-06 ***
местообСамар:доза   2.4325     0.6588   3.692 0.000222 ***
местообТатар:доза   2.0514     0.4859   4.221 2.43e-05 ***
местообХвал:доза    1.1058     0.3763   2.938 0.003301 **
    Null deviance: 335.485  on 124  degrees of freedom
Residual deviance:  71.045  on 111  degrees of freedom
AIC: 173.65
```

Отметим также, что величина AIC-критерия после включения предиктора `местооб` уменьшилась с 190.6 до 173.6.

На рис. 129 разница между регионами проиллюстрирована в виде соответствующих линий пробит-регрессии. На этом рисунке указана также горизонтальная линия (пробит 0), относительно которой можно определить значения LD_{50} для каждого региона.

```
lf = length(levels(df$местооб))

# Линии регрессии в координатах доза-пробит
plot(df.plot$доза, df.plot$пробит, type = "l", lwd = 2,
     xlab = "Доза", ylab = "Пробит доли умерших")
for(i in 1:lf) abline(coef(M3a)[i], coef(M3a)[lf+i], col=i+1)
legend("bottomright", c("Все", levels(df$местооб)),
       col = 1:8, lwd = c(2, rep(1,7)))
```

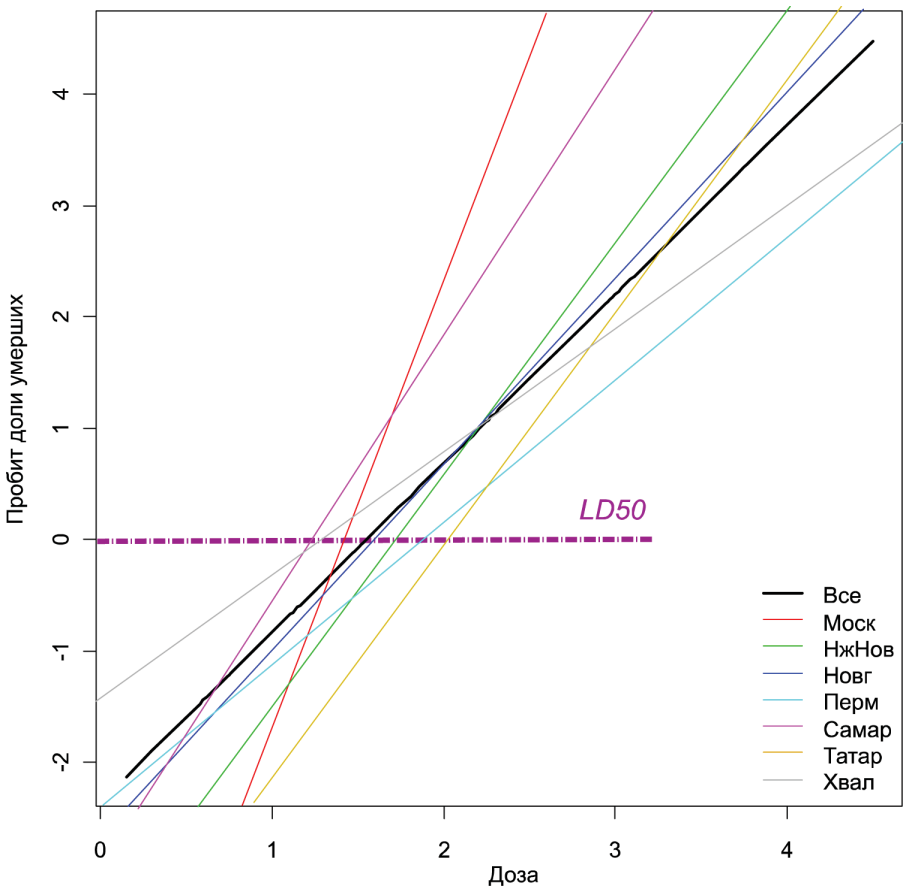


Рисунок 129

```
# Рассчитаем региональные величины LD50 для яда гадюки
df.dl = data.frame(region = levels(df$местооб),
                   LD50 = rep(0, lf), SE = rep(0, lf))
for (i in 1:lf) {
     a = dose.p(M3a, c(i, lf + i))
     df.dl[i, 2] = a[1]
df.dl[i, 3] = as.numeric(attr(a, "SE")) }

df.dl
  region     LD50         SE
1   Моск 1.339411 0.08731109
2  НжНов 1.697298 0.12916933
3   Новг 1.600140 0.13659801
4   Перм 1.878631 0.16627411
5  Самар 1.209885 0.10357006
6  Татар 1.983671 0.12654838
7   Хвал 1.291132 0.17611961
```

Заметим, что LD_{50} – это точечный показатель, не дающий полного представления о процессе развития эффекта по мере изменения дозы. Например, регионы Москва и Хвалынск имеют близкие LD_{50}, но они существенно различаются по скорости нарастания эффекта дозы. Эту скорость, в частности, отражает величина коэффициента угла наклона $\hat{\beta}_1$ (`местообМоск:доза` = 4.25 против `местообХвал:доза` = = 1.11). Читателю, интересующемуся данной темой, советуем обратить внимание на пакет `drc`, специально разработанный для выполнения анализа зависимостей типа «доза–эффект» (Ritz & Streibig, 2005, http://bit.ly/1DoIby6).

Логистическая регрессия

Другая возможность преобразования бинарного отклика в интервальную шкалу заключается в расчете так называемого «*отношения шансов*» («*odds ratio*»). Если событие происходит с вероятностью p, то шанс представляет собой отношение $O(p) = p/(1 - p)$. К сожалению, $O(p) = 0$ при $p = 0$ и $O(p) = +\infty$ при $p = 1$, то есть подобного рода трансформация является неполной, не охватывая отрицательные значения отклика. Поэтому функцию связи обычно задают в виде *логита* («*logit link function*») или логарифма отношения шансов: $g(y) = \log\left(\frac{p}{1-p}\right)$.

Обобщенная логит-линейная модель, или логистическая регрессия на m предикторов, имеет вид

$$g(y) = \log\frac{p(y)}{1-p(y)} = \beta_0 + \sum_{k=1}^{m}\beta_k x_k.$$

Оценки параметров этой модели можно трактовать следующим образом: при изменении значения предиктора на единицу значение логарифма отношения шансов зависимой переменной y изменится на величину соответствующего коэффициента β_k. Как и в случае пробит-регрессии, мы делаем предположение о том, что отклик имеет биномиальный закон распределения.

В целях облегчения интерпретации коэффициентов нам все же лучше получать не логит-значения, а непосредственно предсказанные вероятности P для каждой комбинации независимых переменных. Например, в случае с одним предиктором логит можно преобразовать в вероятность следующим образом:

$$\log\left(\frac{P}{1-P}\right)=\beta_0+\beta_1 X \;\Rightarrow\; P=\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$

Рассмотрим построение логистической регрессии в R на следующем примере. Ежемесячно с мая по октябрь 2006 г. на различных глубинах (0.8 и 4 м) оз. Нарочь (Беларусь) собирали по 15 моллюсков *Dressena polymorpha* и учитывали их зараженность личинками трематоды *Echinoparyphium recurvatum*. Необходимо ответить на вопрос: зависит ли доля зараженных моллюсков от времени отбора проб и глубины обитания?

Загружаем файл с исходными данными (доступен на сайте книги), где столбцы содержат следующие переменные: `Day` – количество дней с начала исследования, `Depth` – фактор глубины взятия проб; `Infected` и `Noninfected` – число зараженных и незараженных животных соответственно:

```
infection <- read.table(file = "dreissena_infection.txt",
                        header = TRUE, sep = "\t")
str(infection)
'data.frame':   12 obs. of  4 variables:
 $ Day        : int  0 26 56 92 124 152 0 26 56 92 ...
 $ Depth  : Factor w/ 2 levels "0.8m","4m": 1 1 1 1 1 1 2 2 2 ...
 $ Infected   : int  1 1 1 0 1 1 3 0 0 2 ...
 $ Noninfected: int  17 16 14 15 14 14 12 15 13 13 ...
```

В R логистическую регрессию можно рассчитать при помощи функции **`glm`**() двумя способами:

- отклик приводится в виде матрицы, в которой первый столбец содержит число «больных», а второй – число «здоровых» особей (так мы делали выше в примере с ядом гадюк);
- отклик приводится в виде доли «больных» экземпляров; при этом используется аргумент `weights` для указания общего числа наблюдений, на основе которого рассчитываются соответствующие доли.

Выполним расчет на основе второго варианта с использованием предварительно вычисленных долей и построим модель регрессии:

```
n.total <- infection$Infected + infection$Noninfected
prop.inf <- infection$Infected/n.total

M2 <- glm(prop.inf ~ Day + Depth, weights = n.total,
          family = binomial(link = "logit"), data = infection)

summary(M2)
Call:
glm(formula = prop.inf ~ Day + Depth, family = binomial(link = "logit"),
data = infection)
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.836627   0.651911  -5.885 3.98e-09 ***
Day          0.011039   0.004623   2.388  0.01695 *
Depth4m      1.543597   0.537679   2.871  0.00409 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to be 1)
    Null deviance: 34.394  on 11  degrees of freedom
Residual deviance: 18.146  on  9  degrees of freedom
AIC: 45.338
```

Если выполнить расчет на основе первого варианта (с использованием предварительно вычисленной матрицы численностей), то мы получим совершенно идентичные результаты (читатель может проверить это самостоятельно):

```
inf.tbl <- cbind(infection$Infected, infection$Noninfected)
M1 <- glm(inf.tbl ~ Day + Depth,
          family = binomial(link = "logit"), data = infection)
```

Проведем теперь оценку статистической значимости модели M2 по критерию χ^2 (при этом проводится последовательное включение факторов в модель):

```
anova(M2, test = "Chisq")
Analysis of Deviance Table
Model: binomial, link: logit
Response: prop.inf
Terms added sequentially (first to last)
      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                     11     34.394
Day    1   6.4053        10     27.989  0.011378 *
Depth  1   9.8424         9     18.146  0.001705 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Отметим, что использование в модели обоих предикторов Day и Depth является статистически значимым. Второй вариант тестирования заключается в анализе отличий девианса полученной логит-модели от девианса «пустой» модели без параметров:

```
M0 <- glm(prop.inf ~ 1, family = binomial(link = "logit"), data = infection)

anova(M2, M0, test = "Chisq")
Analysis of Deviance Table
Model 1: prop.inf ~ Day + Depth
Model 2: prop.inf ~ 1
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1         9     18.146
2        11     34.394 -2  -16.248 0.0002964 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Мы также с легкостью можем построить логистическую модель, используя «сырые» данные (то есть без предварительного расчета долей). Например, для рассматриваемой задачи можно использовать таблицу, каждая строка которой соответствует одной особи, а бинарная переменная EchinoPresence принимает значение 1, если животное заражено:

```
inf.raw <- read.table(file = "dreissena_infection_raw_data.txt",
                      header = TRUE, sep = "\t")

head(inf.raw)
  Day Depth Length EchinoPresence
1   0  0.8m     25              0
2   0  0.8m     25              0
3   0  0.8m     25              0
4   0  0.8m     25              0
5   0  0.8m     28              1
6   0  0.8m     23              0

M3 <- glm(EchinoPresence ~ Day + Depth,
          family = binomial(link = "logit"), data = inf.raw)

summary(M3) # с сокращениями
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.948497   0.682565  -5.785 7.26e-09 ***
Day          0.009975   0.004669   2.136  0.03264 *
Depth4m      1.754168   0.580965   3.019  0.00253 **
    Null deviance: 134.20  on 181  degrees of freedom
Residual deviance: 117.36  on 179  degrees of freedom
AIC: 123.36
```

Добавим теперь в модель дополнительно показатель длины раковины моллюсков Length и оценим по критерию χ^2 значимость различий девиансов до и после включения нового предиктора:

```
M4 <- glm(EchinoPresence ~ Length + Day + Depth,
          family = binomial(link = "logit"), data = inf.raw)

anova(M4, M3, test = "Chisq")
Analysis of Deviance Table
Model 1: EchinoPresence ~ Length + Day + Depth
Model 2: EchinoPresence ~ Day + Depth
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1       178     116.72
2       179     117.36 -1 -0.64642    0.4214
```

Поскольку статистически значимых преимуществ у модели M4 нет, целесообразно использовать более «экономную» модель M3.

Интересно, что прогнозирующая эффективность полученной логистической модели в данном примере оказалась невелика: все предъявляемые объекты обучающей выборки предсказываются моделью M3 как незараженные:

```
mp <- predict(M3, type = "response")
classified <- ifelse(mp > (1 - mp), 1, 0)

# Ошибка классификации (~ 14%):
mean(inf.raw$EchinoPresence!= classified)
[1] 0.1428571

table(Факт = inf.raw$EchinoPresence, Прогноз = classified)
     Прогноз
Факт   0   1
   0 160   0
   1  22   0
```

Здесь имеет место более чем распространенная ситуация, связанная с выраженным дисбалансом в распределении классов отклика: зараженных особей оказалось в семь раз больше, чем незараженных. Логистическая модель по умолчанию исходит из предположения о соотношении классов 50/50 и, соответственно, использует вероятность 0.5 в качестве пороговой при принятии решения о классификации того или иного объекта. Обычно для нахождения оптимальной пороговой вероятности используется ROC-анализ (от «*Receiver Operator Characteristic*» – см. Леонов, 2009, http://bit.ly/1zlSn6u; James et al., 2013; Kuhn & Johnson, 2013, а также статью, посвященную описанию пакета `pROC`, – http://bit.ly/1CTC0yh). Кроме ROC-анализа, имеются и другие подходы, о которых можно в подробностях прочитать в фундаментальных монографиях James et al. (2013) и Kuhn & Johnson (2013).

8.4. Пример использования обобщенных моделей для оценки экологической толерантности

Основной задачей моделирования окружающей среды («*habitat models*») является количественная оценка экологического отклика различных видов живых организмов на внешние и внутренние возмущения. Эта проблема актуальна не только в отношении биоты, но и для человека (например, какое количество выпитого джина является «оптимальным»?) и социума в целом.

Традиционно используемые статистические методы предполагают, что кривая зависимости показателя толерантности вида y от величины воздействующего фактора x имеет симметричную форму колоколообразной гауссианы $y = he^{-(x-\mu)^2/2\sigma^2}$ с тремя интерпретируемыми параметрами: μ – локальный оптимум на оси x, которому соответствует максимум толерантности h; σ – стандартное отклонение на шкале градиента относительно этого оптимума.

Однако в результате сложного воздействия комплекса взаимосвязанных факторов эта зависимость часто имеет ярко выраженный ассиметричный или полимодальный характер. Поэтому, приняв традиционную симметричную гауссову модель только как образец для сравнения, рассмотрим статистические методы

аппроксимации асимметричных кривых отклика различными типами моделей GLM и GAM.

Интересно проследить математическую идентичность гауссовой модели отклика и GLM в форме полинома 2-й степени, если задать распределение Пуассона при логарифмической функции связи

$$\log(y) = \beta_0 + \beta_1 x + \beta_2 x^2 \;\Leftrightarrow\; y = h\exp[-(-x - \mu)^2/2\sigma^2].$$

При этом удовлетворяются условия: $\mu = -\beta_1/2a_2$; $\sigma = (-1/2\beta_2)^{0.5}$; $h = \exp(\beta_0 - \beta_1^2/4\beta_2)$. Использование иных семейств распределений или полиномов 3-й и более высоких степеней будет подчеркивать отличия отклонений функции отклика от симметричной гауссианы.

Модели с нормально распределенным откликом

Рассмотрим модели отклика на примере распределения популяционной плотности личинок комара-звонца вида *Palpomyia sp.* на различных участках малых рек, впадающих в оз. Эльтон (комплексные исследования института ИЭВБ РАН). Подробное изложение различных аспектов анализа данных см. в статье Зинченко и др. (2014; http://bit.ly/1aBJO0N). Напомним, что с этой проблемой мы уже имели дело в разделе 7.6, когда анализировали изменчивость толерантности *Tanytarsus kharaensis* в отношении солености водотоков с использованием логистической функции.

Подготовим комплект данных для расчета пяти статистических моделей: двух (GLM и GAM) – на основе прологарифмированной численности особей и трех (GAUSS, GLM и GAM) – с использованием бинарной переменной встречаемости вида. В расчетах использовались отдельные фрагменты модуля R «Species response curves», разработанного Д. Зелены (D. Zelený) в рамках надстройки к программе JUICE и доступного для скачивания на сайте http://bit.ly/1cAgFEX).

```
library(vegan); library(mgcv)

SpecEnv <- read.delim("Elton_TolInt.txt", header = TRUE)
gr <- SpecEnv$S  # Действующий фактор – минерализация

# Логарифмируем численность вида Palpomyia sp. (см. vegan)
amblog.species <- decostand(SpecEnv$CePal_sp, "log")

# Создаем бинарную переменную встречаемости
bin.species <- SpecEnv$CePal_sp
bin.species[bin.species > 0] <- 1

data.frame(gr, Nat = SpecEnv$CePal_sp,
NatLog = amblog.species, Bin = bin.species)
         gr   Nat    NatLog Bin
9    14.375     0  0.000000   0
10   31.588    40  6.321928   1
11   27.310   390  9.607330   1
12   28.672  2860 12.481799   1
```

```
13   10.280      0  0.000000   0
14    9.746      0  0.000000   0
```

Предположим, что зависимость численности популяции от солености воды может быть линейной или описываться произвольным полиномом до 3-й степени включительно. Будем выполнять селекцию степени полинома по минимуму AIC-критерия. Для этого определим предварительно функцию, возвращающую объект «лучшей» модели:

```
model.GLM <- function (part.species, gr, family='gaussian') {
    glm.1 <- glm(part.species ~ poly(gr, 1), family=family)
    glm.2 <- glm(part.species ~ poly(gr, 2), family=family)
    glm.3 <- glm(part.species ~ poly(gr, 3), family=family)
    allAIC <- c(extractAIC(glm.1)[2], extractAIC (glm.2)[2],
                extractAIC (glm.3)[2])
    switch(c(1, 2, 3)[allAIC == min(allAIC)],
      {glm.GLM <- glm.1; GLM.deg <- 1},
      {glm.GLM <- glm.2; GLM.deg <- 2},
      {glm.GLM <- glm.3; GLM.deg <- 3})
return (glm.GLM)
}
```

Получаем первую модель (приведен сокращенный формат вывода):

```
GLM.amb <- model.GLM(amblog.species, gr)

summary(GLM.amb)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.7365     0.2678  10.218 1.90e-15 ***
poly(gr, 3)1  26.5543     2.2881  11.605  < 2e-16 ***
poly(gr, 3)2  -9.7435     2.2881  -4.258 6.38e-05 ***
poly(gr, 3)3 -17.2841     2.2881  -7.554 1.31e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
    Null deviance: 1460.06  on 72  degrees of freedom
Residual deviance:  361.25  on 69  degrees of freedom
AIC: 333.9
```

В этом случае GLM сводится к функции обычной полиномиальной регрессии

$$\ln(N) = 2.73 + 26.5S - 9.74S^2 - 17.3S^3$$

и может быть представлена традиционной кривой колоколообразной формы. Интересно, что если мы проведем расчеты с использованием функции **lm**(), то получим идентичную модель, но с другим протоколом итоговых статистик:

```
summary(lm(amblog.species ~ poly(gr,3)))
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.7365     0.2678  10.218 1.90e-15 ***
poly(gr, 3)1  26.5543     2.2881  11.605  < 2e-16 ***
```

```
poly(gr, 3)2  -9.7435     2.2881  -4.258 6.38e-05 ***
poly(gr, 3)3 -17.2841     2.2881  -7.554 1.31e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 2.288 on 69 degrees of freedom
Multiple R-squared: 0.7526,     Adjusted R-squared: 0.7418
F-statistic: 69.96 on 3 and 69 DF,  p-value: < 2.2e-16
```

Для построения модели GAM нам надо подобрать оптимальное число степеней свободы сглаживающего сплайна. Эти действия мы выполним по аналогичной схеме:

```
model.GAM <- function(part.species, gr, family='gaussian') {
    gam.3 <- gam(part.species ~ s(gr, k=3), family=family)
    gam.4 <- gam(part.species ~ s(gr, k=4), family=family)
    gam.5 <- gam(part.species ~ s(gr, k=5), family=family)
    allAIC <- c(extractAIC (gam.3)[2], extractAIC (gam.4)[2],
                extractAIC (gam.5)[2])
    switch (c(1, 2, 3)[allAIC == min(allAIC)],
      {gam.GAM <- gam.3; GAM.deg <- 3},
      {gam.GAM <- gam.4; GAM.deg <- 4},
      {gam.GAM <- gam.5; GAM.deg <- 5})
return (gam.GAM)
}

GAM.amb <- model.GAM(amblog.species, gr)

summary(GAM.amb)
Family: gaussian
Link function: identity
Formula: part.species ~ s(gr, k = 4)
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.7365     0.2614   10.47 6.77e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Approximate significance of smooth terms:
        edf Ref.df     F p-value
s(gr) 2.976      3 74.45  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
R-sq.(adj) =  0.754   Deviance explained = 76.4%
GCV score = 5.2741  Scale est. = 4.9868    n = 73
```

Полученная нами модель основана на 4 степенях свободы: одна приходится на свободный член `Intercept` (обратите внимание, что он совпал с таковым в модели GLM), а остальные 3 (точнее, 2.976) «погружены» в сплайн **s**(gr), который практически имеет смысл той же самой кубической параболы. Мы можем также убедиться в статистической значимости сглаживающего члена, «полюбоваться» на коэффициент детерминации `R-sq.(adj)`, долю объясненного девианса `Deviance explained` и прочие технические детали расчетов.

Важными параметрами жизнедеятельности любой популяции являются экологический оптимум и интервалы толерантности на шкале действующего фактора. (Из энциклопедии: «*Диапазон толерантности – минимальное и максимальное значения неблагоприятного фактора, переносимого данным организмом или экосистемой в целом*».) Однако для всех моделей, кроме гауссианы, неясен теоретический метод их оценки. На практике (Coudun & Gégout, 2006; http://bit.ly/1J8D5Y6) приняты два эвристических способа нахождения критических точек популяционной изменчивости на оси абсцисс: *а*) по значениям ординат, составляющим половину от достигнутой величины экологического максимума, и *б*) по минимальному диапазону значений фактора, ограничивающему заданную площадь распределения вероятностей под модельной кривой.

Богатый материал для содержательного анализа предоставляет визуализация построенных моделей. Определим функцию, которая возвратит нам для любой модели `M` расчетные значения ординат `fit`, а также абсциссы экологического максимума `Sopt` и точек, ограничивающих 80% площади под кривой `amplitude.range`:

```
fit.model <- function (M, axx) {
    fit <- predict(M, list(gr = axx), type = 'response')
    Sopt <- axx[fit == max(fit)][1]
      auc <- 80;
      whole.auc <- sum(fit)
      part.auc <- whole.auc * (auc/100)
      for(l in seq(max(fit)[1], min(fit)[1], length = 100))
        { if (sum(fit[fit >= l]) >= part.auc)
              {last.l <- l; break ()}
        }
      amplitude.prob <- last.l
    amplitude.range <- range(axx[fit >= amplitude.prob])
    return(list(fit, Sopt, amplitude.range))
}
```

Теперь мы можем изобразить на графике (рис. 130) кривые отклика по 100 предсказанным значениям и указать положение критических точек:

```
axx <- seq(range(gr)[1], range(gr)[2], len = 100)
l.GLM <- fit.model(GLM.amb, axx)

plot(gr, amblog.species, type = "p", pch = 20,
     xlab = "Минерализация S, г/л",
     ylab = "Численность ln(N)")
lines(axx, l.GLM[[1]], lwd = 2)
lines(axx, fit.model(GAM.amb, axx)[[1]], lwd = 2, lty = 4)
abline(v = l.GLM[[2]], lty = 2)
abline(v = l.GLM[[3]][1], lty = 2)
abline(v = l.GLM[[3]][2], lty = 2)
text(23.8, 0, "Smin", font = 4)
text(32.8, 0,"Sopt", font = 4)
text(39, 0, "Smax", font = 4)
legend("topleft", lty = c(1, 4), lwd = 2, c("GLM", "GAM"))
```

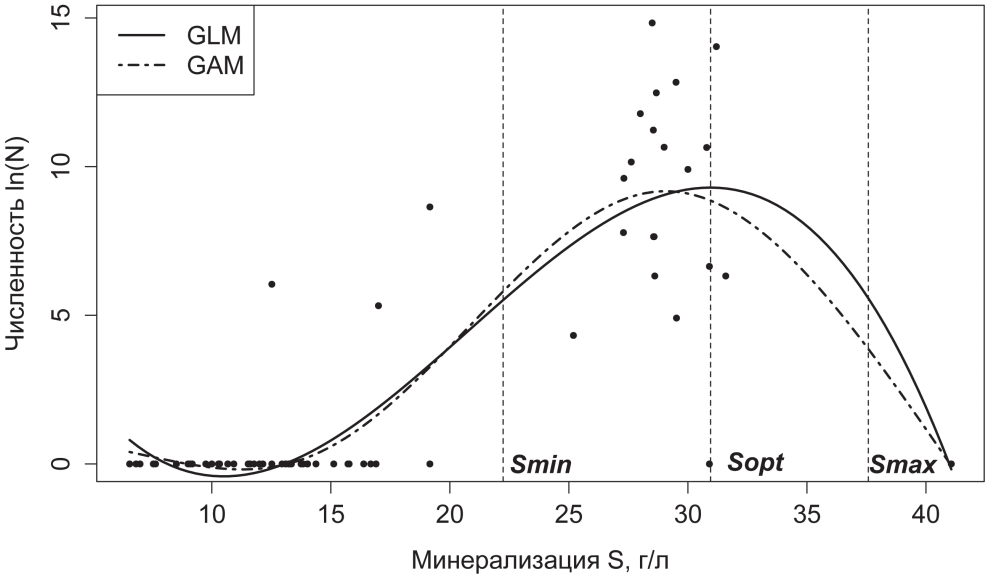


Рисунок 130

Как следует по рис. 130, в соответствии с прогнозом по GLM экологический оптимум солености воды, соответствующий максимуму популяционной плотности *Palpomyia* sp., составляет $S_{опт}$ = 31 г/л, а диапазон толерантности вида (ограничен пределами области, составляющей 80% площади под кривой отклика) находится в интервале от 22.2 до 37.6 г/л. Оптимум минерализации $S_{опт}$ = 29.2 г/л, рассчитанный по модели GAM, оказался несколько смещенным в сторону меньших значений.

Модели с бинарным откликом

Построим теперь три модели, ориентируясь лишь на вероятности встретить экземпляры *Palpomyia* sp. в водоемах с разной соленостью. Как было отмечено выше, аппроксимация частот гауссовой кривой может быть выполнена в рамках модели Пуассона при логит-трансформации вероятностей $P(x)$ и с использованием квадратичной параболы в качестве функции регрессии (ter Braak & Looman, 1986, http://bit.ly/1JSJozV):

```
GLM.gaus <- glm(bin.species ~ gr + I(gr^2), family = poisson)

summary(GLM.gaus)
Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -13.016747   4.186785  -3.109  0.00188 **
gr            0.971451   0.379070   2.563  0.01039 *
I(gr^2)      -0.018038   0.008194  -2.201  0.02771 *
```

```
---
Signif. codes:  0 C***T 0.001 C**T 0.01 C*T 0.05 C.T 0.1 C T 1
    Null deviance: 52.774  on 72  degrees of freedom
Residual deviance: 14.761  on 70  degrees of freedom
AIC: 64.761
```

Модели GLM и GAM мы получим, используя определенные выше функции **model.GLM**() и **model.GAM**(). Статистическую значимость модели оценим, используя критерий χ^2:

```
GLM.bin <- model.GLM(bin.species, gr, family = 'binomial')

anova(GLM.bin, test = "Chisq")
Analysis of Deviance Table
Model: binomial, link: logit
Response: part.species
Terms added sequentially (first to last)
            Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                           72     89.355
poly(gr, 3)  3   64.587        69     24.768 6.148e-14 ***
---
Signif. codes:  0 C***T 0.001 C**T 0.01 C*T 0.05 C.T 0.1 C T 1

GAM.bin <- model.GAM(bin.species, gr, family = 'binomial')

summary(GAM.bin)
Family: binomial
Link function: logit
Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.9288     0.8205  -2.351   0.0187 *
---
Signif. codes:  0 C***T 0.001 C**T 0.01 C*T 0.05 C.T 0.1 C T 1
Approximate significance of smooth terms:
       edf Ref.df Chi.sq  p-value
s(gr) 2.619  2.879  24.71 1.52e-05 ***
---
Signif. codes:  0 C***T 0.001 C**T 0.01 C*T 0.05 C.T 0.1 C T 1
R-sq.(adj) =  0.784   Deviance explained = 72.5%
UBRE score = -0.56439  Scale est. = 1         n = 73
```

Убедившись в статистической значимости построенных моделей, покажем на графике вид этих функций и отметим положение максимума встречаемости (рис. 131). Для модели GAUSS вектор прогнозируемых значений рассчитаем на основе специальной функции:

```
GAUS.fun <- function(p, x)
{ exp(p[1] + p[2]*x + p[3]*(x^2)) }
p.glm.gaus <- coef(GLM.gaus)
```

```
fit.GAUS <- GAUS.fun(p.glm.gaus, axx)

#  Для других моделей используем ранее определенную функцию
l.GLM <- fit.model(GLM.bin, axx)
l.GLM[[2]]
[1] 26.76949  #  Экологический оптимум для GLM

l.GAM <- fit.model(GAM.bin, axx)
l.GAM[[2]]
[1] 27.11814  #  Экологический оптимум для GAM

plot(gr, bin.species, type = "p",
     xlab = "Минерализация S, г/л", ylab = "Вероятность, p",
     ylim = c(0, 1.2), pch = 20)
lines(axx, l.GLM[[1]], col = "blue", lwd = 2)
abline(v = l.GLM[[2]], col="blue", lty = 2)
lines(axx, l.GAM[[1]], col = "green", lwd = 2)
abline(v = l.GAM[[2]], col = "green",lty = 2)
lines(axx, fit.GAUS, col = "red", lwd = 2)
legend("topleft", col = c("red", "blue", "green"), lwd = 2, c("GAUSS", "GLM", "GAM"))
```

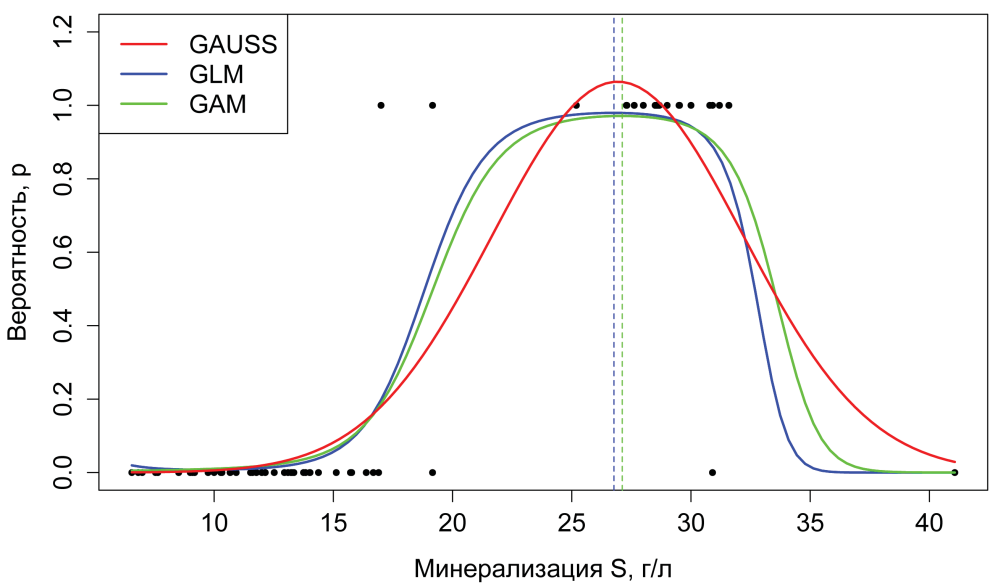


Рисунок 131

При сравнении моделей, построенных на основе нормально распределенного (`GLM.gaus`) и бинарного (`GLM.bin`) откликов, мы обнаружим, что экологический оптимум обилия вида немного сместился в область более низкой солености воды (с 31 до 27 г/л).

8.5. Ковариационный анализ

Ковариационный анализ (ANCOVA, от «*analysis of covariance*») охватывает модели, которые одновременно содержат как качественные, так и количественные переменные, называемые *ковариатами*. Другими словами, ковариационная модель является как бы синтезом регрессионного и дисперсионного анализов:

$$y_i = \sum_{j=1}^{k} f_{ij}\theta_j + \sum_{j=1}^{m} \beta_j x_i^{(j)} + \varepsilon_{ij},$$

где k – возможные типы условий эксперимента (в данном случае однофакторного), $F = \{f_1, ..., f_k\}$; m – число независимых ковариат, $X = \{x^{(1)}, ..., x^{(m)}\}$. Как это обычно делается для общих линейных моделей, индикаторные переменные f_{ij} приравниваются к 1, если j-е условие эксперимента имеет место при наблюдении y_i, и 0 в противном случае. Коэффициенты θ_i определяют эффект влияния j-го условия, β_j – значения соответствующих коэффициентов регрессии Y по $x^{(j)}$, а ε_{ij} – это независимые случайные ошибки с нулевым математическим ожиданием. При включении в модель второго, третьего и последующих факторов в правой части приведенного уравнения появятся слагаемые, отвечающие за эффекты их уровней.

Ковариационный анализ позволяет разложить общую вариацию зависимой переменной на две составляющие:

- степень случайной изменчивости отклика Y, связанную с эффектами F, то есть с влиянием категориальных предикторов и их комбинаций; если значимой взаимосвязи здесь нет, то для оценки степени воздействия ковариат на отклик достаточно воспользоваться обычным регрессионным анализом;
- долю изменчивости отклика, обусловленную влиянием непрерывных предикторов X; если воздействие таких ковариат оказывается статистически незначимым, то достаточно воспользоваться обычным дисперсионным анализом.

Основное назначение ковариационного анализа – получение оценок параметров $\theta_1, ..., \theta_k$ и $\beta_1, ..., \beta_m$, а также статистических критериев для проверки различных гипотез относительно значений этих параметров. Основные теоретические и прикладные процедуры ковариационного анализа относятся к линейным моделям, но имеются некоторые особенности касательно интерпретации результатов. Согласно приведенной выше формуле модели, коэффициенты регрессии отклика на ковариаты не зависят от качественных факторов, что геометрически можно представить в виде нескольких *параллельных* линий, соответствующих каждому уровню f_j. Для оценки статистической значимости эффекта категориальной переменной проверяется гипотеза $H_0(F)$: $\theta_1 = \theta_2 = ... = \theta_k$. Вторая проверяемая гипотеза будет состоять в том, что все регрессионные коэффициенты ковариат равны 0, то есть $H_0(X)$: $\beta_1 = \beta_2 = ... = \beta_m = 0$. При наличии *взаимодействий* между уровнями исследуемого фактора (или факторов) и тех или иных ковариат регрессионные линии больше *не будут параллельными*. Проверяемые при этом статистические гипотезы станут сложнее, и их точная формулировка будет определяться конкретными задачами исследования.

Рассмотрим особенности проведения ковариационного анализа на примере (Dalgaard, 2008). Таблица hellung из пакета ISwR содержит данные о численности conc (экз./мл) и диаметре diameter особей ресничных инфузорий *Tetrahymena*, которые выращивались в питательной среде без (glucose = 2) и с добавлением (glucose = 1) глюкозы. Необходимо ответить на вопрос: оказывает ли наличие глюкозы влияние на размер инфузорий, с учетом того, что этот размер может быть также связан с численностью популяции *Tetrahymena*? По-английски ковариату «численность популяции» могли бы назвать еще «*confounding variable*», то есть «смешивающей переменной», имея в виду то, что эффект глюкозы установить сложно, поскольку он как бы «смешивается» с эффектом численности популяции.

Изобразим данные hellung в виде диаграммы рассеяния, добавив сглаживающие кривые (и соответствующие 95%-ные доверительные области) для визуализации характера связи между conc и diameter:

```
data(hellung, package = "ISwR")
hellung$glucose = as.factor(hellung$glucose)
```

```
head(hellung)
  glucose    conc  diameter
1       1  631000      21.2
2       1  592000      21.5
3       1  563000      21.3
4       1  475000      21.0
5       1  461000      21.5
6       1  416000      21.3
```

```
library(ggplot2)
ggplot(hellung, aes(x = conc, y = diameter, color = glucose)) +
       geom_point() + geom_smooth()
```

По рис. 132 хорошо видно, что при наличии глюкозы в питательной среде диаметр клеток в среднем был существенно выше, чем в среде без глюкозы. Это наблюдение предполагает, что для описания зависимости размера инфузорий от их численности необходимы две отдельные модели. Поскольку имеет место обратная экспоненциальная зависимость между conc и diameter, то перед построением этих моделей мы сначала трансформируем данные путем логарифмирования (это также может привести к нормализации распределения остатков и стабилизировать их дисперсию – см. главу 7). Для оценки коэффициентов моделей воспользуемся хорошо известной нам функцией **lm**():

```
# без глюкозы:
(lm.nogluc <- lm(log10(diameter) ~ log10(conc),
                 data = hellung, subset = glucose == 2))
Coefficients:
(Intercept)  log10(conc)
    1.63476     -0.05968

# с глюкозой:
(lm.gluc <- lm(log10(diameter) ~ log10(conc), data = hellung, subset = glucose == 1))
```

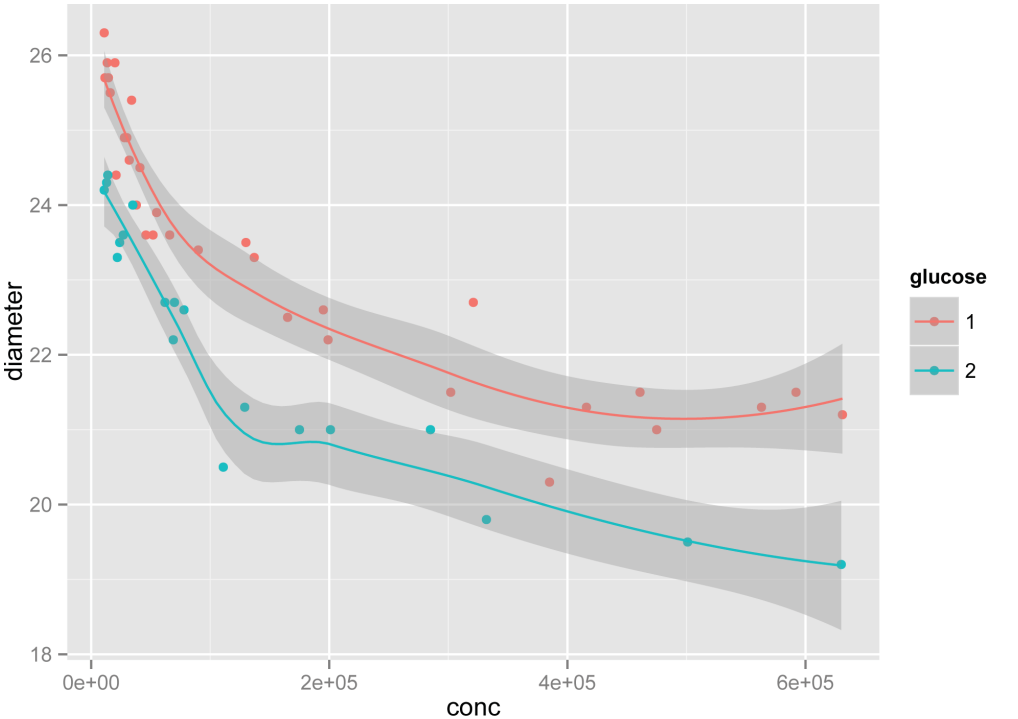


Рисунок 132

```
Coefficients:
(Intercept)  log10(conc)
     1.6313      -0.0532
```

Изобразим полученные регрессионные линии на графике, добавив для сравнения также линию регрессии, подогнанную без учета влияния глюкозы (рис. 133):

```
p_symb = c(17, 1)
plot(hellung$conc, hellung$diameter,
     pch = p_symb[as.numeric(hellung$glucose)], log = "xy",
     xlab = "cons", ylab = "diameter")
abline(lm.nogluc, col = "blue", lwd = 2)
abline(lm.gluc, col = "green", lwd = 2)
# Линия регрессии без учета влияния глюкозы:
abline(lm(log10(diameter)~ log10(conc), data = hellung), col = "red", lty = 2)
legend("topright", bty = "n", legend = c("без глюкозы",
       "с глюкозой", "обе группы"), lwd = c(2, 2, 1),
       lty = c(1, 1, 2), col = c("blue", "green", "red"))
```

Глядя на рис. 133, мы можем задаться следующими вопросами:

- Различаются ли линии, соответствующие группам «с глюкозой» и «без глюкозы», по углу наклона (то есть по степени влияния ковариаты `conc`)?

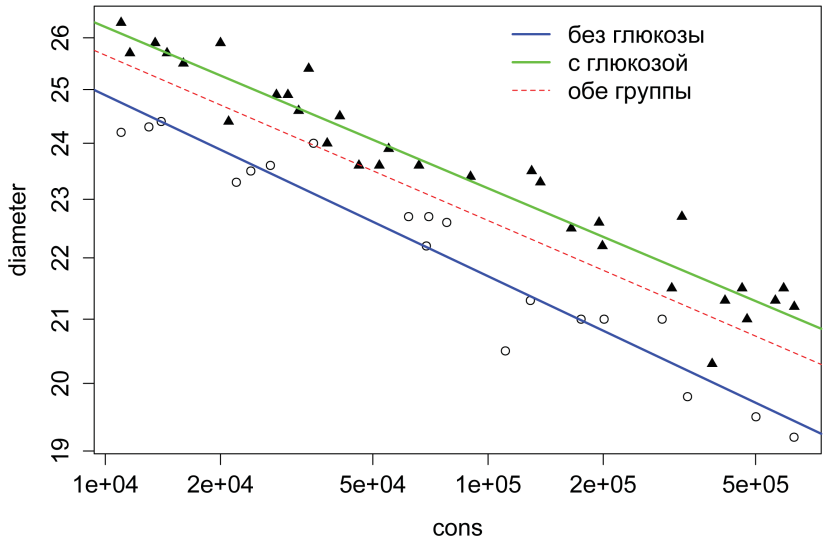


Рисунок 133

- Различаются ли группы «с глюкозой» и «без глюкозы» по среднему размеру клеток (то есть достаточно ли далеко соответствующие групповые линии регрессии отстоят от линии, полученной без учета влияния глюкозы)?

Преимущество ковариационного анализа состоит в том, что он позволяет *одновременно* ответить на оба вопроса в рамках *единой модели*. Для построения такой модели мы воспользуемся все той же функцией `lm()`. Сначала проверим статистическую значимость взаимодействия фактора `glucose` с ковариатой `cons`:

```
AN1 <- lm(log10(diameter) ~ log10(conc)*glucose, data = hellung)

summary(AN1)
Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.631344   0.013879 117.543   <2e-16 ***
log10(conc)          -0.053196   0.002807 -18.954   <2e-16 ***
glucose2              0.003418   0.023695   0.144    0.886
log10(conc):glucose2 -0.006480   0.004821  -1.344    0.185
```

Интерпретация результатов оценки параметров модели `AN1` такова:

1. Первые две строки содержат свободный член и регрессионный коэффициент для группы «с глюкозой» (выделены зеленым цветом). При численности C ожидаемое среднее значение логарифма диаметра клеток D будет выражено зависимостью

   $$\log_{10}D = 1.6313 - 0.0532 \times \log_{10}C.$$

2. Остальные две строки содержат разницу между группами по свободному члену и регрессионному коэффициенту. Другими словами, при выращивании инфузорий в среде без добавления глюкозы зависимость будет иметь вид:

$$\log_{10}D = (1.6313 + 0.0034) - (0.0532 + 0.0064)\times\log_{10}C.$$

Поскольку снижение регрессионного коэффициента для культур без глюкозы оказалось статистически незначимым ($p = 0.185$), то линии модели можно принять параллельными. С биологической точки зрения это означает, что вне зависимости от наличия глюкозы размер инфузорий одинаково тесно связан с их численностью. Упростим исходную модель для отражения этого факта:

```
AN2 <- lm(log10(diameter) ~ log10(conc) + glucose, data = hellung)

summary(AN2)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.642132   0.011417  143.83  < 2e-16 ***
log10(conc) -0.055393   0.002301  -24.07  < 2e-16 ***
glucose2    -0.028238   0.002647  -10.67 2.93e-14 ***
```

В соответствии с `summary(AN2)` имеем две ситуации:

- культура с глюкозой: $\log_{10}D = 1.6421 - 0.0554\times\log_{10}C$;
- культура без глюкозы: $\log_{10}D = (1.6421 - 0.0282) - 0.0554\times\log_{10}C$.

Таким образом, клетки в культуре без глюкозы в среднем на 6.3% мельче ($10^{-0.0282} = 0.937$), и это снижение статистически значимо. Статистическую значимость различий групп по среднему размеру клеток подтверждает также таблица дисперсионного анализа:

```
anova(AN2)
Response: log10(diameter)
            Df   Sum Sq  Mean Sq F value    Pr(>F)
log10(conc)  1 0.046890 0.046890  561.99 < 2.2e-16 ***
glucose      1 0.009494 0.009494  113.78 2.932e-14 ***
Residuals   48 0.004005 0.000083
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

В заключение подчеркнем еще раз, что с математической точки зрения модели ковариационного анализа *ничем не отличаются* от обычных линейных моделей. Особенность ковариационного анализа заключается лишь в специфике постановки исследовательских вопросов, на которые он помогает ответить (например, «оказывает ли фактор F существенное влияние на переменную Y, учитывая, что переменная Y также может быть связана с количественной переменной Z?»).

8.6. Модели со смешанными эффектами для иерархически организованных данных

Основные идеи

В разделе 6.6 нами подчеркивалось, что если при проведении многофакторного дисперсионного анализа с учетом взаимодействий число уровней категориальных переменных достигает больших значений (например, пяти и более), возникает необходимость оценки значительного количества коэффициентов модели. Было бы ошибкой механически включать в анализ весь набор возможных параметров и скрупулезно оценивать связанные с ними эффекты. Как правило, многое зависит от постановки задачи и природы анализируемых факторов.

Например, мы хотим сравнить удои молока у разных пород коров. В таблице данных имеются два фактора: порода скота F_1 и пастбища F_2, на которых производится их выпас. Можно построить модель с двумя факторами и проделать все предусмотренные протоколом дисперсионного анализа действия. Однако есть ли необходимость, наравне с интересующим нас фактором F_1, выполнять оценку соответствующих коэффициентов для фактора F_2, тратя на это часто весьма ограниченное число степеней свободы? Во-первых, у нас нет такой задачи – сравнивать между собой пастбища. Во-вторых, если порода скота имеет строго фиксированные уровни, определенные исследователем, то список пастбищ, на которых паслись коровы, мог быть выбран случайно (например, на следующий год места выпаса могли бы поменяться). По условию задачи нам было бы вполне достаточно просто оценить долю общей дисперсии надоев молока, обусловленную фактором F_2.

Независимые переменные, уровни которых надежно определены исследователем и могут быть воспроизведены при реализации повторностей, называются *фиксированными факторами*. Модели на их основе, рассмотренные в главе 6, называются моделями I типа, или моделями с *фиксированными эффектами* («*fixed effects*»). В них значение объясняемой переменной Y определяется генеральной средней μ, *дифференциальными* эффектами воздействия индивидуальных факторов, а также комбинаторными эффектами их парных, тройных и т. д., вплоть до m, взаимодействий.

Другой тип эффектов, часто вызывающий интерес исследователей, представляют *случайные эффекты* («*random effects*»). Здесь предполагается, что при реализации повторного эксперимента выборки каждый раз могут извлекаться с иным составом группировки наблюдаемых объектов. Следовательно, уровни изучаемого фактора будут формироваться случайно из некоторой совокупности возможных градаций или назначаться по субъективным соображениям. Модели II типа *со случайными эффектами* используются в ситуациях, когда исследователя интересует не значимость вкладов уровней факторов, а оценка и сравнение компонентов дисперсии распределения включенных в модель дифференциальных эффектов. Это позволяет обобщить наши заключения о природе факторов и значимости их

влияния вне связи с конкретными уровнями, заданными в плане эксперимента. Включение в модель случайного эффекта некоторого фактора позволяет выделить из общей дисперсии σ независимых остатков ε долю вариации, «в среднем» объясняемую влиянием этого фактора. Например, в рамках линейной модели II типа для двух факторов с k повторностями

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ijk}$$

случайные эффекты α_i на i уровнях воздействия фактора A, β_j на j уровнях воздействия фактора B, а также эффект их взаимодействия $(\alpha\beta)_{ij}$ должны быть независимы в совокупности и иметь нормальное распределение с нулевым средним и дисперсиями σ_α, σ_β и $\sigma_{(\alpha\beta)}$ соответственно. Если это условие выполняется, то проверяются нулевые гипотезы

H_0(A): $\sigma_\alpha = 0$,
H_0(B): $\sigma_\beta = 0$,
H_0(AB): $\sigma_{(\alpha\beta)} = 0$,

то есть дисперсия случайного фактора на всех его уровнях равна нулю и не вносит дополнительно вклада в изменчивость наблюдаемой случайной величины.

Модели III типа *со смешанными эффектами* включают как фиксированные, так и случайные факторы (Pinheiro & Bates, 2000; Singer & Willett, 2003; Demidenko, 2004, http://bit.ly/1JfJLnh; Gelman & Hill, 2006; Wood, 2006). Для простой схемы группировки по уровням линейную модель LMEM («*linear mixed-effects model*») со смешанными эффектами удобно представить в матричной форме:

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{Z}\mathbf{b} + \varepsilon,$$
$$\mathbf{b} \sim N(\mathbf{0}, \psi_\theta),\ \varepsilon \sim N(\mathbf{0}, \Lambda\sigma^2),$$

где $\mathbf{y}$ – n-мерный вектор зависимой переменной, n – число ячеек дисперсионного комплекса, $\mathbf{X}$ и $\mathbf{Z}$ – матрицы, состоящие из нулей и единиц и описывающие группировку фиксированных и случайных факторов по уровням в соответствии с планом эксперимента, β – вектор фиксированных эффектов (параметров модели). Распределение вектора случайных эффектов $\mathbf{b}$ описывается вектором нулевых средних и положительно определенной ковариационной матрицей ψ_θ, зависящей от вектора параметров θ, оценка которых является основной целью статистического вывода о природе случайных эффектов. Наконец, Λ – положительно определенная матрица простой структуры, которая обычно используется при моделировании автокорреляции остатков ε, но часто это просто единичная матрица.

Включение случайных факторов в модели с фиксированными эффектами существенно повышает обобщаемость заключений о главном результате исследования при распространении его на другие пространственные, временные или биологические уровни организации. Так, сделав вывод о влиянии одного целевого фактора (например, добавка или отсутствие удобрения), корректно было бы выполнить анализ статистической значимости остальных случайных факторов (таких как возможные места размещения пробных площадок, периоды эксперимента,

варианты видов растений и т. д.) по всем возможным их уровням. Модели со смешанными эффектами широко используются во многих био- и социологических исследованиях (Zuur et al., 2009).

Пример с морскими животными: несколько частных моделей

Рассмотрим небольшой пример, представленный в книге А. Цуура с соавторами (Zuur et al., 2009). Файлы с исходными данными и комплект скриптов R к главам книги можно скачать на сайте http://bit.ly/1DlHen2.

Вы, вероятно, любите бродить по морскому берегу и собирать мелких морских животных (ракушек, рачков, морских звезд, гребешков, крабов, ланцетиков). Именно этим занимался и голландский институт RIKZ (http://bit.ly/1yFHXDS), специалисты которого собрали данные по донным обитателям (бентосу) в девяти областях приливной зоны на Голландском побережье: `Beach` = {1, ..., 9}. Для каждой из 45 площадок сбора проб (`Sample`) была учтена высота пляжа над средним уровнем прилива `NAP` и подсчитано число обнаруженных видов бентоса `Richness`. Дополнительно был рассчитан индекс степени морских возмущений `Exposure`, учитывающий интенсивность приливной волны, наклон дна, зернистость грунта, толщину анаэробного слоя и т. д. Этот индекс был приведен к фактору с двумя уровнями: «*a*» при значениях `Exposure` < 11 и «*b*» при `Exposure` = 1. Необходимо оценить зависимость богатства видов `Richness` от высоты `NAP` и возмущения `Exposure`.

```
RIKZ <- read.table(file = "RIKZ.txt", header = TRUE)
head(RIKZ)
```

```
  Sample Richness Exposure    NAP Beach
1      1       11       10  0.045     1
2      2       10       10 -1.036     1
3      3       13       10 -1.336     1
4      4       11       10  0.616     1
5      5       10       10 -0.684     1
6      6        8        8  1.190     2
```

```
ExposureBeach <- c("a", "a", "b", "b", "a", "b", "b", "a", "a")
```

Наивный подход приведет нас к мысли описать искомую зависимость при помощи хорошо знакомой нам модели линейной регрессии с гауссовым распределением остатков:

$$R_{ij} = \alpha + \beta_1 \times \mathtt{NAP}_{ij} + \beta_2 \times \mathtt{Exposure}_i + \varepsilon_{ij},$$
$$\varepsilon_{ij} \sim N(0, \sigma_2),$$

где R_{ij} – богатство видов участка j на пляже i, $\mathtt{NAP}_{ij}$ и $\mathtt{Exposure}_i$ – предикторы модели, а ε_{ij} – случайные остатки. Однако эта модель не учитывает *эффекта вложенности* данных («*nested data*»): на каждом пляже сделано по 5 проб, и, скорее всего, число видов на участках одного пляжа будет более схожим, чем на участках различных местообитаний. Эту модель целесообразно далее не рассматривать.

Вторая мысль (более плодотворная, но все же не оптимальная) приведет нас к выполнению так называемого *двухэтапного анализа*. На первом этапе мы могли бы рассчитать значения коэффициентов регрессии β_i для каждого i-го пляжа по совокупности из 5 проб:

$$R_{ij} = \alpha + \beta_i \times \mathrm{NAP}_{ij} + \varepsilon_{ij}, \quad j = 1, ..., 5.$$

На втором шаге мы нашли бы параметры общей линейной модели, которая оценивает зависимость коэффициентов $\hat{\beta}_i$, полученных на первом этапе, от предиктора `Exposure`:

$$\hat{\beta}_i = \eta + \tau \times \mathrm{Exposure}_i + b_i, \quad i = 1, ..., 9.$$

Эта модель соответствует тривиальному однофакторному дисперсионному анализу. Код для выполнения подобного двухэтапного анализа приведен ниже.

```
Beta <- vector(length = 9)     # Первый этап
for(i in 1:9){
    tmpout <- summary(lm(Richness~NAP, subset = (Beach == i), data = RIKZ))
Beta[i] <- tmpout$coefficients[2, 1]
}

print(Beta, 3)
[1] -0.372 -4.175 -1.755 -1.249 -8.900 -1.389 -1.518 -1.893 -2.968

# Второй этап
tmp2 <- lm(Beta ~ factor(ExposureBeach), data = RIKZ)

summary(tmp2)
Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)             -3.662      1.099  -3.332   0.0126 *
factor(ExposureBeach)b   2.184      1.649   1.325   0.2268
```

Кроме огорчившей нас статистической незначимости фактора, отражающего абиотические условия, двухэтапный анализ имеет и другие недостатки:

- вариация всех данных для каждого пляжа сводится к одному параметру;
- на втором шаге мы анализируем параметры регрессии, а не наблюдаемые данные (то есть не моделируем непосредственно интересующий нас отклик);
- итоговая статистическая величина не зависит от числа исходных наблюдений (их может быть 5, 50 или 50000).

Наконец, в качестве третьей версии можно представить нашу модель как линейную зависимость богатства видов от высоты `NAP`, где свободный член изменяется для каждого пляжа Beach_i, номер которого выступает в качестве фактора с девятью фиксированными уровнями:

$$R_{ij} = (\alpha + \beta_1 \times \mathrm{Beach}_i) + \beta_2 \times \mathrm{NAP}_{ij} + \varepsilon_{ij}.$$

Проблема с этой моделью заключается в том, что хотя мы вовсе не интересуемся знанием точной природы «эффекта пляжа», нам приходится платить за включение этого члена в модель восемью степенями свободы – слишком высока цена!

Все перечисленные проблемы решаются с использованием моделей со смешанными эффектами, которые объединяют действия описанного выше двухэтапного анализа в одну простую модель:

$$R_i = \mathbf{X}_i \times \beta + \mathbf{Z}_i \times b_i.$$

Здесь R_i является оценкой богатства видов на пляже i, $i = 1, ..., 9$. Предикторы модели включают фиксированный компонент $\mathbf{X}_i \times \beta$ и случайный компонент $\mathbf{Z}_i \times b_i$. Даже для этого небольшого примера есть несколько вариантов структуры матриц $\mathbf{X}_i$ и $\mathbf{Z}_i$, которые мы обсудим ниже. Пока нам важно только то, что эти матрицы имеют размерности $n_i \times p$ и $n_i \times q$ соответственно, где n_i – число наблюдений R_i, p – число предикторов в $\mathbf{X}_i$ и q – число объясняющих переменных в $\mathbf{Z}_i$. Мы рассмотрим последовательно следующие три варианта:

а) модель зависимости между R_i и NAP_i со случайным свободным членом;
б) модель со случайными свободным членом и коэффициентом угла наклона;
в) различные модели со смешанными эффектами, включающими все предикторы.

Модель со случайным свободным членом использует местоположение пляжа Beach_i как случайный фактор, однако предполагает, что весь эффект его влияния сконцентрирован в сдвиге свободного члена:

$$R_i = NAP_i \times \beta + b_i.$$

В развернутой форме эта модель имеет вид:

$$\begin{pmatrix} R_{i1} \\ R_{i2} \\ R_{i3} \\ R_{i4} \\ R_{i5} \end{pmatrix} = \begin{pmatrix} 1\ NAP_{i1} \\ 1\ NAP_{i2} \\ 1\ NAP_{i3} \\ 1\ NAP_{i4} \\ 1\ NAP_{i5} \end{pmatrix} \times \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \times b_i + \begin{pmatrix} \varepsilon_{i1} \\ \varepsilon_{i2} \\ \varepsilon_{i3} \\ \varepsilon_{i4} \\ \varepsilon_{i5} \end{pmatrix},$$

то есть матрица случайных эффектов $\mathbf{Z}_i$ из приведенного выше выражения представляет собой единичный вектор.

Для подгонки модели воспользуемся функцией **`lme`**`()` из пакета `nlme`, которая, в отличие от **`lm`**`()`, позволяет задать случайные эффекты. В частности, параметр `~1|fBeach` определяет случайный свободный член модели (справа от знака `|` указывается предиктор, в соответствии с уровнями которого сгруппированы данные):

```
library(nlme)

RIKZ$fBeach <- factor(RIKZ$Beach)
Mlme1 <- lme(Richness ~ NAP, random = ~1|fBeach, data = RIKZ)
```

```
summary(Mlme1)
Linear mixed-effects model fit by REML
 Data: RIKZ
       AIC     BIC    logLik
  247.4802 254.525 -119.7401
Random effects:
 Formula: ~1 | fBeach
        (Intercept) Residual
StdDev:    2.944065  3.05977
Fixed effects: Richness ~ NAP
               Value Std.Error DF   t-value p-value
(Intercept)  6.581893 1.0957618 35  6.006682       0
NAP         -2.568400 0.4947246 35 -5.191574       0
```

В первой части блока выведенных результатов приведены информационные критерии AIC и BIC, а также достигнутый максимум функции правдоподобия logLik. Эти показатели могут потребоваться нам для селекции оптимальной модели. Вторая и третья части содержат анализ случайных и фиксированных эффектов. Часть, связанная с Fixed effects, определяет зависимость числа видов от высоты пляжа как $6.58 - 2.57 \times \text{NAP}_i$, причем в зависимости от i к свободному члену этого уравнения добавляется величина случайного эффекта b_i. Стандартное отклонение этих случайных приращений свободного члена (StdDev: 2.94) определено выше в части, связанной с Random effects, и почти совпадает со стандартным отклонением для остатков Residual.

Приведем график, иллюстрирующий геометрический смысл этой модели:

```
F0 <- fitted(Mlme1, level = 0)
F1 <- fitted(Mlme1, level = 1)
I <- order(RIKZ$NAP); NAPs <- sort(RIKZ$NAP)
plot(NAPs, F0[I], lwd = 2, col = "blue" , type = "l",
     ylim = c(0, 22), ylab = "Число видов", xlab = "NAP")
for(i in 1:9){
    x1 <- RIKZ$NAP[RIKZ$Beach == i]
    y1 <- F1[RIKZ$Beach == i]; K <- order(x1)
    lines(sort(x1), y1[K])
}
text(RIKZ$NAP, RIKZ$Richness, RIKZ$Beach, cex = 0.9)
```

Синяя линия на рис. 134 отображает центральную тенденцию зависимости числа видов от высоты расположения пляжа относительно уровня прибоя. К свободному члену этой зависимости добавлены (с учетом знака) случайные эффекты местоположения каждого пляжа, вызывающие параллельные сдвиги остальных 9 прямых. Цифрами показаны координаты точек для каждого из 9 пляжей.

Модель со случайными свободным членом и коэффициентом угла наклона также использует местоположение пляжа Beach_i в качестве случайного фактора, однако включает дополнительное предположение о том, что и степень зависимости между числом видов R_i и высотой NAP_i варьирует от пляжа к пляжу. Иными словами, для каждого значения фактора Beach_i корректируются не только свободный член,

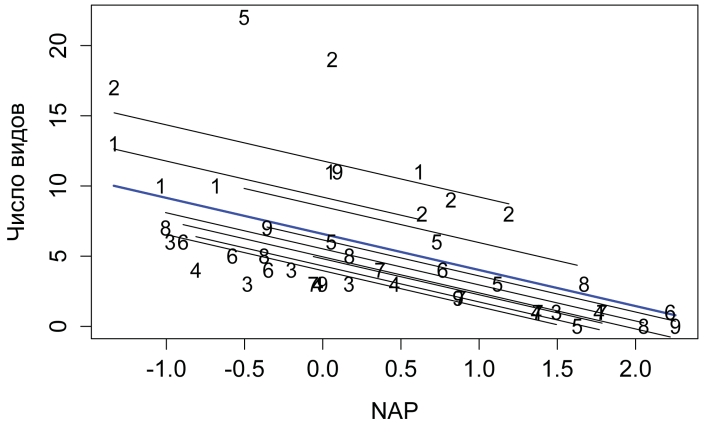


Рисунок 134

но и регрессионный коэффициент (рис. 135). Математически эту модель можно представить следующим образом:

$$\begin{pmatrix} R_{i1} \\ R_{i2} \\ R_{i3} \\ R_{i4} \\ R_{i5} \end{pmatrix} = \begin{pmatrix} 1\ NAP_{i1} \\ 1\ NAP_{i2} \\ 1\ NAP_{i3} \\ 1\ NAP_{i4} \\ 1\ NAP_{i5} \end{pmatrix} \times \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \begin{pmatrix} 1\ NAP_{i1} \\ 1\ NAP_{i2} \\ 1\ NAP_{i3} \\ 1\ NAP_{i4} \\ 1\ NAP_{i5} \end{pmatrix} \times \begin{pmatrix} b_{i1} \\ b_{i2} \end{pmatrix} + \begin{pmatrix} \varepsilon_{i1} \\ \varepsilon_{i2} \\ \varepsilon_{i3} \\ \varepsilon_{i4} \\ \varepsilon_{i5} \end{pmatrix},$$

то есть в матрицу $\mathbf{Z}_i$, определяющую конфигурацию случайных эффектов, дополнительно включается набор значений NAP_i. Это выражается в спецификации аргумента `random` функции **`lme`**() в виде `~1+NAP|fBeach`:

```
Mlme2 <- lme(Richness ~ NAP, method = "REML", random = ~1 + NAP | fBeach, data = RIKZ)

summary(Mlme2)
Random effects:  Formula: ~1 + NAP | fBeach
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev   Corr
(Intercept) 3.549100 (Intr)
NAP         1.715015 -0.99
Residual    2.702785
Fixed effects:  Richness ~ NAP
                Value Std.Error DF   t-value p-value
(Intercept)  6.588729 1.2647708 35  5.209425   0e+00
NAP         -2.830029 0.7229514 35 -3.914549   4e-04
 Correlation:
    (Intr)
NAP -0.819

#  Код для графика на рис. 135 в целом идентичен представленному выше
```

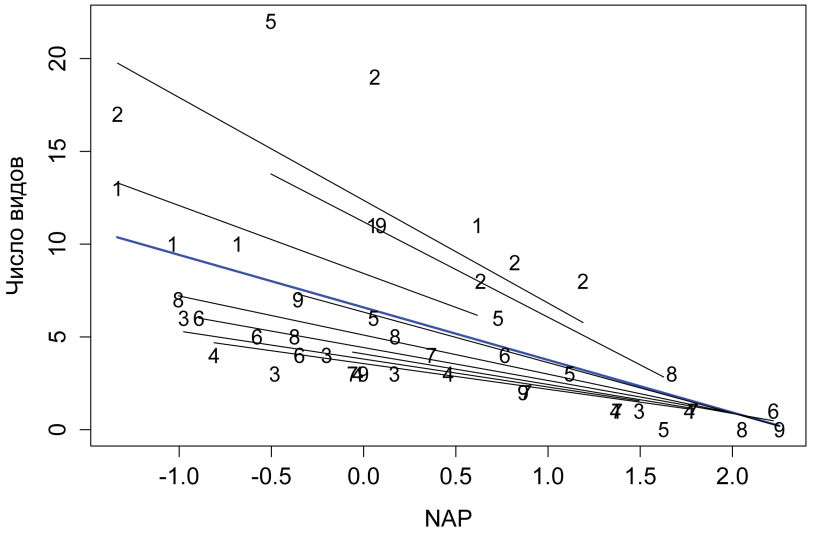


Рисунок 135

Изменению подвергся блок выводимых результатов для случайных эффектов, который стал включать уже два члена. Первый, как и в предыдущей модели, определяет оценку стандартного отклонения случайных изменений свободного члена (`Intercept`) (b_1). Второй характеризует изменчивость угла наклона (b_2) прямых, описывающих зависимость R_i от NAP_i на отдельных пляжах. Можно отметить, что вариация, сконцентрированная в случайном эффекте b_1, в четыре раза превышает дисперсию, заключенную в b_2, при высоком коэффициенте корреляции между ними (`-0.99`).

Обратите внимание также на аббревиатуру `REML`, которая обозначает метод оценки параметров моделей со смешанными эффектами. Речь идет об используемом по умолчанию «*методе максимального правдоподобия с ограничениями*» («*restricted maximum likelihood*»). Относительно возможного вопроса о том, чем он отличается от обычного метода максимального правдоподобия (кстати, его также можно использовать, если задать параметр `method = "ML"`), большинство книг на эту тему приводят несколько страниц не слишком дружелюбных формул в духе матричной алгебры либо вовсе избегают деталей и представляют REML как способ «корректировки степеней свободы». Ограничимся этим вторым подходом и мы.

Модели с использованием всех предикторов. Преобразуем индекс интенсивности абиотических возмущений `Exposure` в фактор `fExp` и выполним построение линейной модели, в фиксированной части которой учтем как дифференциальные эффекты, так и эффект парного взаимодействия `NAP` и `fExp`:

```
RIKZ$fExp <- RIKZ$Exposure
RIKZ$fExp[RIKZ$fExp == 8] <- 10
RIKZ$fExp <- factor(RIKZ$fExp, levels = c(10, 11))
```

```
Mlme4 <- lme(Richness ~1 + NAP * fExp,
             random = ~1 + NAP | fBeach, data = RIKZ)

summary(Mlme4)
Random effects:  Formula: ~1 + NAP | fBeach
            StdDev   Corr
(Intercept) 2.421276 (Intr)
NAP         1.507835 -0.801
Residual    2.607753

Fixed effects: Richness ~ 1 + NAP * fExp
                Value Std.Error DF   t-value p-value
(Intercept)  9.118945 1.2242357 34  7.448684  0.0000
NAP         -3.879203 0.8816476 34 -4.399947  0.0001
fExp11      -5.534743 1.8510032  7 -2.990132  0.0202
NAP:fExp11   2.429496 1.3327641 34  1.822900  0.0771
 Correlation:
           (Intr) NAP    fExp11
NAP        -0.637
fExp11     -0.661  0.421
NAP:fExp11  0.421 -0.662 -0.658
```

Общий смысл представленной модели заключается в следующем. Как всегда при построении регрессионных моделей, имеется неопределенность в отношении истинной формы зависимости видового богатства R_i от высоты пляжа $\texttt{NAP}_i$ и интенсивности приливных возмущений $\texttt{fExp}_i$ (фиксированные факторы). Некоторую часть этой неопределенности мы можем объяснить *вариацией между местообитаниями*. Объявив для этого в качестве случайного эффекта fBeach, мы оцениваем вклад этого фактора в виде стандартного отклонения соответствующего нормального распределения StdDev. Иными словами, местообитание перестает быть самостоятельным предиктором модели, но, внося свой вклад в общую дисперсию, наряду с необъясненными остатками ε ограничивает возможность совершения ошибки 1-го рода (отклонение верной нулевой гипотезы).

Здесь, конечно, естественен вопрос: является ли найденная модель оптимальной? Рассчитаем еще две модели: в первой из них исключим член NAP:fExp11, оценивающий парное взаимодействие (поскольку по *t*-критерию он оказался незначимым):

```
Mlme5 <- lme(Richness ~1 + NAP + fExp,
             random = ~1 + NAP | fBeach, data = RIKZ)

summary(Mlme5)
Fixed effects: Richness ~ 1 + NAP + fExp
                Value Std.Error DF   t-value p-value
(Intercept)  8.407714  1.183419 35  7.104595  0.0000
NAP         -2.808422  0.759642 35 -3.697034  0.0007
fExp11      -3.704917  1.517669  7 -2.441189  0.0447
```

```
 Correlation:
       (Intr) NAP
NAP    -0.500
fExp11 -0.573 -0.024
```

Вторая модель возвращает нас к модели со случайным свободным членом:

```
Mlme6 <- lme(Richness ~1 + NAP + fExp, data = RIKZ,
             random = ~1 | fBeach, method = "REML")

summary(Mlme6)
Random effects:  Formula: ~1 | fBeach
        (Intercept) Residual
StdDev:    1.907175 3.059089
```

Для выбора оптимальной модели можно использовать разные подходы. Так, ниже приведена таблица со значениями AIC-критерия для всех построенных моделей (первая из перечисленных моделей, ранее не обсуждавшаяся нами, вообще не включает абиотических факторов, а показывает лишь изменчивость числа видов морских организмов для различных пляжей). Согласно этим значениям AIC, оптимальной является модель `Mlme4`. Следует подчеркнуть, однако, что некоторые из перечисленных моделей имеют разную структуру фиксированных эффектов, в связи с чем сравнение их AIC-значений нужно выполнять с большой осторожностью. Подробнее об этой проблеме, а также о способах ее решения можно узнать в замечательной книге Zuur et al. (2009).

	Фиксированный эффект	Случайный эффект	DF	AIC
1	`Richness ~ 1`	`~1 | fBeach`	3	267.11
2	`Richness ~ NAP`	`~1 | fBeach`	4	247.48
3	`Richness ~ NAP`	`~1 + NAP | fBeach`	6	244.38
4	`Richness ~ 1 + NAP * fExp`	`~1 + NAP | fBeach`	8	**237.13**
5	`Richness ~ 1 + NAP + fExp`	`~1 + NAP | fBeach`	7	240.53
6	`Richness ~ 1 + NAP + fExp`	`~1 | fBeach`	5	240.55

Стоит также упомянуть проблему проверки того, являются ли построенные нами модели статистически значимыми в целом. Ответ на этот вопрос для моделей со смешанными эффектами сводится к применению различных методов рандомизации, бутстрепа и марковских цепей Монте-Карло (MCMC). Некоторые примеры их использования приведены в книге Шитиков, Розенберг (2014).

8.7. Индуктивные модели (метод группового учета аргументов)

В конце 60-х годов группа украинских ученых во главе с академиком А. Г. Ивахненко разработала базовую концепцию индуктивного моделирования, которая легла в основу целой серии эволюционных алгоритмов искусственного интеллекта.

В отличие от классического *дедуктивного* пути («от предполагаемых механизмов изучаемого процесса к верификации конкретной математической модели»), при *индуктивном* подходе сама конечная модель, построенная в ходе ее постепенной адаптации к исходному набору данных, является объективным результатом исследования (см. статью В. С. Степашко на сайте http://bit.ly/1HIrOzi). При этом используются следующие *принципы самоорганизации*:

1. Определяется класс моделей, в рамках которого возможен последовательный переход от простейших вариантов к наиболее сложным (то есть индуктивный процесс). Здесь могут использоваться самые различные типы моделей: линейная или нелинейная регрессия, ряды Фурье, кластеры объектов, наборы логических правил и прочее.
2. Задаются механизм эволюции моделей и критерии их отбора, благодаря которым каждая отдельная «мутация» оценивается с точки зрения полезности для улучшения качества конечного результата. При этом наиболее целесообразный путь постепенного усложнения самоорганизующейся модели основан на принципах *неокончательности решений* Д. Габора и *массовой селекции* А. Г. Ивахненко, которые заключаются в необходимости сохранения достаточной «свободы выбора» из некоторого подмножества лучших решений на каждом шаге адаптации.
3. Формирование модели оптимальной сложности, описывающей скрытые в зашумленных экспериментальных данных закономерности, осуществляется с использованием принципа *внешнего дополнения*. Только внешние критерии, основанные на новой информации, являются: *а*) своеобразным фильтром, отбрасывающим «мусор» в исходных данных, и *б*) средством, позволяющим блокировать переусложнение искомой модели, которое, по словам Ивахненко, «*так же вредно, как и ее недоусложнение*».

Эти принципы легли в основу такого направления анализа данных, как *метод группового учета аргументов* – МГУА («*Group Method of Data Handling*», GMDH). При синтезе моделей МГУА используются различные процессы самоорганизации: комбинаторные (*COMBI* и *MULTI*), многорядные (*MIA*) и релаксационные (*RIA*) схемы, алгоритмы гибридизации, основанные на конечных стохастических автоматах, и т. д. На сайте www.gmdh.net представлено для ознакомления большое число монографий и журнальных публикаций (1968–2004 гг.), описывающих теоретические и практические аспекты метода.

Рассматриваемый нами ниже многорядный итерационный алгоритм МГУА (*MIA*) является одним из обобщений регрессионного анализа, где некоторая сложная зависимость случайной величины y от m предикторов $y = F(x_1, x_2, ..., x_m)$ заменяется последовательностью рядов «частных описаний», являющихся простыми функциями двух переменных.

Первый ряд селекции (рис. 136) формируется с использованием s моделей регрессии, основанных на всех парных комбинациях исходных переменных (x_i, x_j), то есть $u_1 = f(x_1, x_2)$, $u_2 = f(x_1, x_3)$, ..., $u_s = f(x_{m-1}, x_m)$. Рассчитанные по этим моделям прогнозируемые значения u_1, ..., u_s используются для построения частных описа-

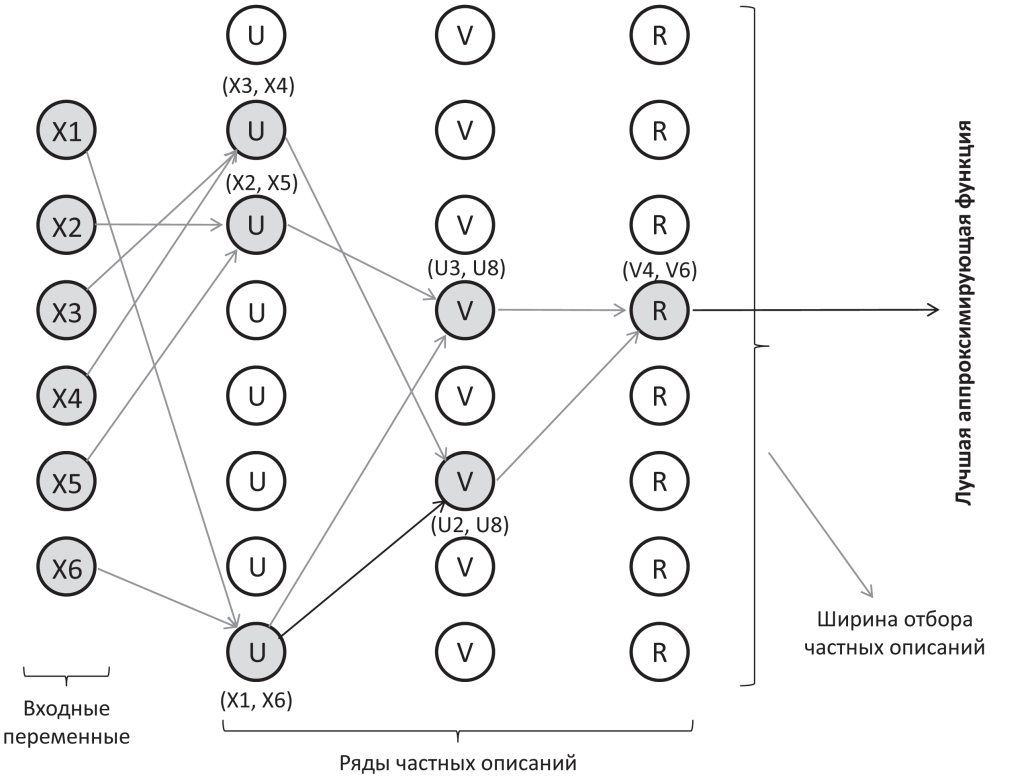


Рисунок 136

ний во втором ряду селекции – $v_1 = f(u_1, u_2)$, $v_2 = f(u_1, u_3)$, ..., $v_p = f(u_{s-1}, u_s)$. Этот итеративный процесс продолжается далее, то есть формируются третий и последующие ряды селекции.

Результирующая сложность такой многорядной модели зависит от трех факторов: вида опорной функции f для каждого частного описания, количества рядов селекции и ширины отбора частных описаний. Отметим предварительно ключевую роль здесь критерия самоорганизации L, адекватный выбор которого призван обеспечить:

а) ранжирование частных описаний внутри каждого ряда, чтобы пропустить для дальнейшего усложнения модели только лучшие решения, адекватные по отношению к новым данным и устойчивые к «мусору»;

б) остановку итерационного процесса, если на текущем шаге селекции достигнуто оптимальное значение критерия L_{opt} (Васильев, Ланге, 2002, http://bit.ly/1HIrOzi).

В оригинальном описании алгоритма исходную выборку предварительно разделяют случайным образом на две статистически однородные части: *обучающую* и *проверочную*. В качестве внешнего критерия селекции можно использовать стан-

дартную ошибку частных описаний на проверочной выборке, что обеспечивает несмещенность получаемого решения. Нами в представленном ниже скрипте предпринята попытка использовать для этого ошибку s_{CV} при кросс-проверке.

Каждое частное описание обычно задается линейной или полиномиальной функцией f двух переменных вида

$$y_k = \beta_0 + \beta_1 x_1 + \beta_2 x_j$$

или

$$y_k = \beta_0 + \beta_1 x_i + \beta_2 x_j + \beta_3 x_i x_j + \beta_4 x_i^2 + \beta_5 x_j^2; \quad 0 < i < m, 0 < j < m.$$

Коэффициенты такого регрессионного уравнения можно легко оценить методом наименьших квадратов даже по небольшому числу наблюдений в обучающей выборке.

Усложнение модели происходит только за счет многорядности, поскольку на каждом шаге количество используемых коэффициентов β_i, как минимум, удваивается (если формируется достаточно много рядов, то число аргументов модели может насчитывать миллионы). На практике число шагов ограничивают значением m, хотя минимум критерия селекции обычно достигается достаточно быстро.

Ширину отбора частных описаний обычно (но не всегда) также принимают равной исходному числу предикторов m, то есть, например, при $m = 7$ из 21 модели, полученной на первом шаге, во второй ряд отбирается только 7 частных описаний, наилучших по используемому критерию. По завершении итерационного процесса (то есть когда величина L_{opt} стабилизируется) последний ряд селекции содержит m конечных моделей оптимальной сложности.

Рассмотрим реализацию многорядного алгоритма на примере. Используем для этого таблицу `state.x77` из базового дистрибутива R, которая содержит 8 показателей по каждому из 50 штатов США: численность населения (`Population`), доход на душу населения (`Income`), процент неграмотных (`Illiteracy`), число убийств и смертей по неосторожности на 100 000 жителей (`Murder`), процент выпускников средней школы (`HS Grad`), число дней в году с температурой ниже 0° (`Frost`), площадь штата (`Area`) и средняя продолжительность жизни (`Life Exp`). Последний признак будем считать зависимой переменной y и построим для нее модель регрессии, включающую остальные семь предикторов.

Поскольку названия некоторых переменных таблицы содержат пробелы (что по синтаксису языка R нежелательно), предварительно выполним их коррекцию. Определим также список формул моделей, которые мы будем использовать для подгонки частных описаний:

```
X <- state.x77[, c(1:3, 5:8, 4)]
colnames(X)[8] = "Life.Exp"
colnames(X)[5] = "HS.Grad"
M <- ncol(X); N <- nrow(X)

names(X)
[1] "Population" "Income"    "Illiteracy" "Murder"    "HS.Grad"
[6] "Frost"      "Area"      "Life.Exp"
```

```
FormulList <- c(
"Y~X1","Y~X2", "Y~X1+X2", "Y~X1+X2+I(X1*X2)",
"Y~X1+X2+I(X1*X1)","Y~X1+X2+I(X2*X2)",
"Y~X1+X2+I(X1*X2)+I(X1*X1)","Y~X1+X2+I(X1*X2)+I(X2*X2)",
"Y~X1+X2+I(X1*X2)+I(X1*X1)+I(X2*X2)")

as.data.frame(FormulList)
                          FormulList
1                              Y~X1
2                              Y~X2
3                            Y~X1+X2
4                   Y~X1+X2+I(X1*X2)
5                   Y~X1+X2+I(X1*X1)
6                   Y~X1+X2+I(X2*X2)
7          Y~X1+X2+I(X1*X2)+I(X1*X1)
8          Y~X1+X2+I(X1*X2)+I(X2*X2)
9 Y~X1+X2+I(X1*X2)+I(X1*X1)+I(X2*X2)
```

Для проведения перекрестной проверки моделей, составляющих частные описания, будем использовать функцию **cvLm**() из пакета cvTools. Параметр cost = = rtmspe этой функции означает, что мы будем использовать в качестве критерия самоорганизации *L* усеченное («*trimmed*») стандартное отклонение ошибки предсказания на проверочных совокупностях (trim = 0.1 – пропорция «урезания»). Другой параметр – folds – в нашем случае будет определять, что исходная выборка объемом N = 50 случайно разбивается на пять групп численностью по 10 наблюдений каждая (K = 10), после чего последовательно по четырем группам вычисляются параметры модели, а оставшаяся пятая группа используется для выполнения предсказаний (см. раздел 7.5). Вся эта процедура повторяется R = 5, то есть фактически нами для каждой тестируемой модели рассчитывается R*N/K = 25 значений ошибки кросс-проверки s_{CV}, среднее из которых после цензурирования (то есть отбрасывания 10% экстремальных значений) возвращается функцией **cvLm**().

Определим также две собственные функции. Первая – **NEML**() – для каждой пары векторов предикторов (X1, X2) и отклика Y рассчитывает значения критерия AIC для всех линейных моделей по списку из FormulList, после чего наиболее оптимальная из них подвергается перекрестной проверке описанным выше способом. Вторая функция – **PRED.NEML**() – просто возвращает при тех же входных аргументах модельный объект с номером Iopt:

```
library(cvTools)
set.seed(1234)
folds <- cvFolds(N, K = 10, R = 5)

NEML <- function (Y, X1, X2) {
allAIC <- sapply(1:length(FormulList), function(k)
          AIC(lm(as.formula(FormulList[k]))))
Iopt <- which.min(allAIC)
Mopt <- lm(as.formula(FormulList[Iopt]))
```

```
cvFitLm <- cvLm(Mopt, cost = rtmspe, folds = folds, trim = 0.1)
return (c(Iopt, cvFitLm$cv))
}

PRED.NEML <- function (Y, X1, X2, Iopt) {
Mopt <- lm(as.formula(FormulList[Iopt]))
return (Mopt)
}
```

Теперь мы можем реализовать вычисления по многорядному алгоритму МГУА:

```
eps <- 0.001 # Некоторое малое число для оценки стабилизации L
AllModel <- list() # Список для хранения коэффициентов

for (h in 1:(M-1)) {      # Наращивание рядов селекции
  res1 <- matrix(0, ncol=6, nrow=(M-1)*(M-2)/2)
  colnames(res1) <- c("H", "#", "I", "J", "Nmod", "CVcrit")
  a <- 0
  for (i in 1:(M - 2)) {     # Перебор всех пар предикторов
     for (j in (i + 1):(M - 1))   {
         a <- a + 1
         res1[a, 1:4] = c(h, 0, i, j)
         res1[a, 5:6] = NEML(X[, M], X[, i], X[, j])
     }
   }

#  Получили таблицу частных описаний для одного ряда селекции.
#  Сортируем ее по величине критерия самоорганизации:
   res1 <- res1[order(res1[, 6]),]
   res1[, 2] <- 1:nrow(res1)
   if(h == 1) {                  # Первый шаг селекции
      CVglob <- res1[1, 6]        # Лучшая модель в ряду
      result <- res1[1:(M - 1), ]
   } else {
# Если шаг селекции не первый, проверяем сходимость процесса:
      if((CVglob - res1[1, 6]) < eps) break
      CVglob <- res1[1, 6]
      result <- rbind(result, res1[1:(M - 1), ])
   }
   listMod1 <- lapply(1:(M - 1), function(k) PRED.NEML(X[, M],
               X[, res1[k, 3]], X[, res1[k, 4]], res1[k, 5]))
# Для (M-1)- лучших моделей извлекаем коэффициенты и прогнозы
    listCoef1 <- lapply(listMod1, function(fit) fit$coef)
    XP <-  sapply(listMod1, function(fit) fit$fit)
    AllModel <- c(AllModel, listCoef1)
# Заменяем исходную таблицу на подогнанные значения Y (!!!)
    X[, 1:(M - 1)] <- XP
}
```

```
result  # Получаем результаты моделирования
      H  # I J Nmod    CVcrit
 [1,] 1  1 1 4    4 0.5837048
 [2,] 1  2 4 7    6 0.6270402
 [3,] 1  3 3 4    2 0.6451471
 [4,] 1  4 2 4    5 0.6585327
 [5,] 1  5 4 5    3 0.6616672
 [6,] 1  6 4 6    3 0.6627902
 [7,] 1  7 3 5    4 0.8189247
 [8,] 2  1 1 2    1 0.5603183
 [9,] 2  2 1 3    1 0.5603183
[10,] 2  3 1 6    1 0.5603183
[11,] 2  4 2 3    1 0.5814782
[12,] 2  5 1 7    3 0.5819404
[13,] 2  6 2 7    3 0.5867959
[14,] 2  7 1 5    8 0.5915364
[15,] 3  1 1 7    2 0.5464488
[16,] 3  2 2 7    2 0.5464488
[17,] 3  3 3 7    2 0.5464488
[18,] 3  4 4 7    2 0.5464488
[19,] 3  5 5 7    2 0.5464488
[20,] 3  6 6 7    2 0.5464488
[21,] 3  7 1 6    3 0.5542008
```

Нетрудно заметить, что последний (третий) шаг селекции (H = 3) в некотором смысле является формальностью, поскольку шесть его наилучших (L_{opt} = 0.546) моделей, как будет показано ниже, представляют собой тождество $y = v_7$. Выведем последовательно информацию о коэффициентах частных описаний во втором и первом рядах:

```
AllModel[[15]]  # X2 = V7
  (Intercept)           X2
1.339446e-13 1.000000e+00

AllModel[[14]]  # X1 = U1  X2 = U5
(Intercept)          X1          X2  I(X1 * X2)  I(X2 * X2)
229.7523966  47.4951225 -53.0944657  -0.6636776   0.7110299

AllModel[[1]]   #  X1 = Population   X2 = Murder
  (Intercept)            X1            X2    I(X1 * X2)
 7.337297e+01 -1.065221e-04 -3.695612e-01  1.856252e-05

AllModel[[5]]   #  X1 = Murder    X2 = HS.Grad
(Intercept)          X1          X2
 70.2970840  -0.2370901   0.0438873

sd(X[,1] - X[,M]) # Стандартное отклонение разностей (не остатков!!!)
0.7019943
```

Запишем уравнения модели в привычном для нас виде:

$$\texttt{Life.Exp} = 229.7 + 47.5u_1 - 53.1u_5 - 0.663u_1u_5 + 0.711u_5^2,$$

где

$$u_1 = 73.3 - 0.000106\texttt{Population} - 0.369\texttt{Murder} + 1.85\times10^{-5}\texttt{Population}\times\texttt{Murder};$$
$$u_5 = 70.3 - 0.237\texttt{Murder} + 0.0438\texttt{HS.Grad}.$$

Беглый анализ модели говорит о том, что продолжительность жизни зависит от уровня образования и обратно пропорциональна криминальной обстановке. Для полноты картины выполним построение обычной линейной модели, на основании которой мы в целом можем прийти к аналогичным выводам:

```
fitLm <- lm(Life.Exp ~ ., data = X)

summary(fitLm)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.094e+01  1.748e+00  40.586  < 2e-16 ***
Population   5.180e-05  2.919e-05   1.775   0.0832 .
Income      -2.180e-05  2.444e-04  -0.089   0.9293
Illiteracy   3.382e-02  3.663e-01   0.092   0.9269
Murder      -3.011e-01  4.662e-02  -6.459 8.68e-08 ***
HS.Grad      4.893e-02  2.332e-02   2.098   0.0420 *
Frost       -5.735e-03  3.143e-03  -1.825   0.0752 .
Area        -7.383e-08  1.668e-06  -0.044   0.9649
---
Residual standard error: 0.7448 on 42 degrees of freedom
Multiple R-squared: 0.7362,     Adjusted R-squared: 0.6922
F-statistic: 16.74 on 7 and 42 DF,  p-value: 2.534e-10
# Кросс-проверка полной линейной модели:
cvLm(fitLm, cost = rtmspe, folds = folds, trim = 0.1)
10-fold CV results:
       CV
0.7040126
```

В качестве резюме можно отметить, что применение МГУА оправдано в следующих случаях: *а*) когда отсутствует или почти отсутствует априорная информация о структуре модели и распределении ее параметров, *б*) объем имеющейся выборки наблюдений существенно мал, и имеет место большое число коррелирующих между собой предикторов, *в*) моделируемый процесс имеет объективно сложный характер (например, временные ряды с нестационарным трендом).

8.8. Моделирование структурными уравнениями

Предметом большинства исследований, требующих статистического моделирования, является анализ причинно-следственных связей в изучаемой системе. При этом в ряде общественных или естественных отраслей науки (экономика, социо-

логия, экология и прочие) постановка активного эксперимента не всегда бывает возможна, и поэтому в качестве альтернативы используются специфические статистические процедуры обработки неэкспериментальных данных. В частности, вводятся гипотетические конструкты, не доступные прямому наблюдению, но имеющие тесную эмпирическую связь с независимыми переменными и оказывающие решающее скрытое воздействие на моделируемый отклик. Совместный анализ наблюдаемых и латентных переменных осуществляется при помощи *моделирования структурными уравнениями* («*structural equation modeling*»), которое называют еще многопараметрическим комплексным анализом («*multitrait multimethod analysis*»), путевым анализом («*path analysis*»), линейным причинным анализом («*linear causal analysis*») или методом одновременных уравнений («*simultaneous equations*»).

Обычные уравнения регрессии дают информацию о степени простой связи между переменными x и y в форме утверждения «при изменении x изменяется и y», тогда как структурные уравнения отражают более высокий уровень абстракции, на котором в центре внимания оказываются множественные причинные связи между переменными. Сама модель представляет собой *систему одновременных* (совместных, или взаимозависимых) *уравнений*, где одни и те же переменные одновременно рассматриваются как зависимые в одних уравнениях и независимые в других:

$$\begin{cases} y_1 = b_{12}y_2 + b_{13}y_3 + ... + b_{1m}y_m \quad + a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n + \varepsilon_1, \\ y_2 = b_{21}y_1 + b_{23}y_3 + ... + b_{2m}y_m \quad + a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n + \varepsilon_2, \\ y_3 = b_{31}y_1 + b_{32}y_2 + ... + b_{3m}y_m \quad + a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n + \varepsilon_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_m = b_{m1}y_1 + b_{m2}y_2 + ... + b_{m,m-1}y_{m-1} + a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n + \varepsilon_n. \end{cases}$$

Как следствие изменение какой-то одной переменной не может происходить без неизменности прочих и отражается практически во всей системе взаимосвязанных признаков.

Система одновременных уравнений (или *структурная форма* модели) обычно содержит эндогенные и экзогенные переменные. *Эндогенные переменные* – это зависимые переменные, число которых равно числу уравнений в системе (выше были обозначены как y). *Экзогенные переменные* (обозначены как x) – это переменные, влияющие на эндогенные переменные, но не зависящие от них.

Рассмотрим простой пример и алгоритм его алгебраического решения (он называется еще *косвенным* методом наименьших квадратов). Необходимо построить модель потребления свинины на душу населения y_1 (кг) в зависимости от цены на нее y_2 (руб./кг), располагаемого дохода x_1 (руб.) и расходов по обработке мяса x_2 (% от цены). При этом известно, что:

- потребление свинины пропорционально ее цене и располагаемому доходу, причем потребление падает с ростом цен;
- цена растет с ростом потребления свинины и ростом стоимости ее переработки.

Учитывая эти закономерности, модель можно записать в следующем виде (u_t и v_t – остатки, обусловленные неучтенными факторами):

$$\begin{cases} y_{1t} = a_{12}y_{2t} + b_{11}x_{1t} + u_t \\ y_{2t} = a_{21}y_{1t} + b_{22}x_{2t} + v_t \\ a_{12} < 0 \end{cases}$$

Для оценки регрессионных коэффициентов a_{ij} и b_{ij} структурная форма модели преобразуется в так называемую *приведенную форму*, которая в данном случае имет вид:

$$\begin{cases} y_1 = \delta_{11}x_1 + \delta_{12}x_2 + u_1, \\ y_2 = \delta_{21}x_1 + \delta_{22}x_2 + u_2. \end{cases}$$

Мы получили неодновременную систему уравнений, каждое из которых можно решить методом наименьших квадратов.

На втором этапе находят нелинейную связь между коэффициентами обеих форм модели. В нашем случае достаточно просто подставить одно уравнение в другое:

$$a_{12} = \delta_{12}/\delta_{22}; \quad b_{11} = \delta_{11} - a_{12}\delta_{21}; \quad a_{21} = \delta_{21}/\delta_{11}; \quad b_{22} = \delta_{22} - a_{21}\delta_{12}.$$

Создадим исходные данные для модели и выполним оценку ее параметров с использованием R:

```
Ddf <- as.data.frame(matrix(
  c(-3, 0.6, -200, 3, -1, -0.4, -200, -1,2, -0.2, 0, -1, -1,
    0.6, 100, 6,3, -0.6, 300, -7),
    nrow = 5, ncol = 4, byrow = TRUE,
    dimnames = list(c("1990", "1991", "1992", "1993", "1994"),
                    c("Y1", "Y2", "X1", "X2"))))

Ddf # Данные представлены в виде абсолютных отклонений от среднего
     Y1   Y2   X1 X2
1990 -3  0.6 -200  3
1991 -1 -0.4 -200 -1
1992  2 -0.2    0 -1
1993 -1  0.6  100  6
1994  3 -0.6  300 -7

# Находим коэффициенты регрессии для приведенной модели:
(E1 <- lm(Y1 ~ X1 + X2, data=Ddf)$coef)
  (Intercept)            X1            X2
 6.580070e-17  6.093636e-03 -2.648135e-01

(E2 <- lm(Y2 ~ X1 + X2, data=Ddf)$coef)
  (Intercept)            X1            X2
-2.981894e-17  2.940746e-04  1.120702e-01
```

```
# Пересчитываем коэффициенты для структурной модели
Coef <- matrix(NA, nrow = 2, ncol = 2)
Coef[1,1] = E1[3]/E2[3] ; Coef[1,2] = E1[2]- Coef[1,1]*E2[2]
Coef[2,1] = E2[2]/E1[2] ; Coef[2,2] = E2[3]- Coef[2,1]*E1[3]
Coef
           [,1]        [,2]
[1,] -2.3629243 0.006788512
[2,]  0.0482593 0.124849940
```

Таким образом, решенная система структурных уравнений имеет вид:

$$\begin{cases} y_{1t} = -2.363 y_{2t} + 0.0067 x_{1t} + u_t \\ y_{2t} = 0.048 y_{1t} + 0.125 x_{2t} + v_t \end{cases}$$

Естественно, в сложных случаях такая подстановка неосуществима, и поэтому более распространенным методом решения систем структурных уравнений является *двухшаговый метод* наименьших квадратов («*two-stage least squares*», 2SLS), основанный на использовании так называемых «инструментальных» переменных. Переменные (z_{1t}, z_{2t}, ..., z_{kt}) называются *инструментальными*, если они коррелируют с независимыми переменными x_{it} и не коррелируют с остальными случайными возмущениями (при $n \to \infty$). Тогда процедура

$$\tilde{\tilde{a}} = (z^T x)^{-1} z^T \vec{y}$$

обеспечивает состоятельные оценки параметров модели. Подробно проблемам инструментальных переменных посвящен выпуск журнала «Квантиль», № 2, 2007 (см. quantile.ru).

В двухшаговом МНК используется следующая процедура:

Шаг 1. Строятся модели регрессии каждой эндогенной переменной на **Z** и получают расчетные значения этих переменных обычным методом наименьших квадратов.

Шаг 2. Оцениваются параметры структурной формы уравнения модели: для этого в правую часть вместо значений эндогенных переменных подставляются их оценки, рассчитанные на предыдущем шаге.

В среде R методы моделирования структурными уравнениями реализованы в нескольких пакетах. Для оценивания коэффициентов рассматриваемого примера двухшаговым методом наименьших квадратов используем функцию **tsls**() из пакета sem. Вызов этой функции отличается от построения обычной линейной модели **lm**() включением нового параметра instruments, определяющего список инструментальных переменных:

```
library(sem)
summary(tsls(Y1 ~ Y2 + X1 + X2, # 1-е уравнение системы
       instruments ~ X1 + X2), data = Ddf)
 2SLS Estimates
Model Formula: Y1 ~ Y2 + X1
Instruments: ~X1 + X2
```

```
              Estimate Std. Error    t value Pr(>|t|)
(Intercept) -2.623e-17   0.548006 -4.787e-17   1.0000
Y2          -2.363e+00   1.254677 -1.883e+00   0.2004
X1           6.789e-03   0.003096  2.193e+00   0.1596
```

```
summary(tsls(Y2 ~ Y1 + X2, # 2-е уравнение системы
       instruments ~ X1 + X2, data = Ddf))
 2SLS Estimates
Model Formula: Y2 ~ Y1 + X1
Instruments: ~X1 + X2
              Estimate Std. Error    t value Pr(>|t|)
(Intercept) -1.110e-17    0.15648 -7.095e-17   1.0000
Y1           4.826e-02    0.15217  3.171e-01   0.7812
X2           1.248e-01    0.06867  1.818e+00   0.2107
```

Были получены те же коэффициенты системы структурных уравнений, что и выше.

Следует отметить, что классификация переменных на эндогенные и экзогенные условна и зависит от содержательной концепции принятой модели. В общем случае все переменные – измеряемые и латентные – взаимодействуют друг с другом, однако характер этих взаимодействий устанавливается исследователем на этапе формулирования гипотез. Любые две переменные могут детерминировать друг друга (то есть составлять причинно-следственную пару): детерминирующая переменная будет называться независимой, а детерминируемая – зависимой. Например, в качестве экзогенных переменных могут рассматриваться значения эндогенных переменных за предшествующий период времени (*лаговые переменные*).

Рассмотрим еще один пример, традиционно приводимый при обсуждении моделей с одновременными уравнениями. Нобелевский лауреат в области экономики Л. Клейн предложил следующую простую модель макроэкономики США, содержащую 3 уравнения и 3 балансовых тождества. Тремя экзогенными переменными, которые задаются вне модели, являются:

- G_t – государственные расходы (без заработной платы) в период t;
- T_t – непрямые налоги на бизнес и поступления от экспорта;
- A_t – учитываемый период времени (текущий год t – 1931).

Придавая этим переменным планируемые значения, можно получать с помощью модели прогнозные значения 9 эндогенных переменных:

- C_t – объем валового потребления;
- I_t – объем валовых инвестиций;
- W_t^p и W_t^g – уровень заработной платы (в частном и государственном секторах);
- K_t – валовой объем основных капитальных фондов;
- P_t – прибыль частных компаний;
- X_t – равновесие между спросом и предложением.

Дополнительными экзогенными переменными являются лаговые значения P_{t-1}, K_{t-1} и X_{t-1}, то есть прибыль, основной капитал и увеличение предложения над

спросом за предыдущий год. Основное предположение модели состоит в том, что влияние экзогенных и лаговых переменных условно независимо и они не вносят дополнительной ошибки в оценки значений эндогенных переменных.

Система из трех структурных уравнений включает коэффициенты β_{ij} при экзогенных переменных, α_{ij} – при эндогенных переменных и ошибки ε_i («структурные флуктуации»), имеющие тот же смысл, что и ошибки обычной регрессии:

$$C_t = \alpha_{10} + \alpha_{11} P_t + \beta_{11} P_{t-1} + \alpha_{11}(W_t^p + W_t^g) + \varepsilon_{1t} \quad : \quad \text{потребительская функция;}$$
$$I_t = \alpha_{20} + \alpha_{21} P_t + \beta_{21} P_{t-1} + \beta_{22} K_{t-1} + \varepsilon_{2t} \quad : \quad \text{инвестиционная функция;}$$
$$W^p{}_t = \alpha_{30} + \alpha_{31} X_t + \beta_{31} A_t + \beta_{32} X_{t-1} + \varepsilon_{3t} \quad : \quad \text{динамика заработной платы.}$$

Балансовыми тождествами, которые не включают структурные коэффициенты и ошибки модели, являются:

$$X_t = C_t + I_t + G_t;$$
$$P_t = X_t - T_t - W_t^p;$$
$$K_t = K_{t-1} + I_t.$$

Подгонка модели Клейна осуществлялась с использованием временного ряда данных с 1921 по 1941 г. Эти данные представлены в таблице `Klein` из пакета `sem`. Выполним формирование лаговых переменных (заметим, что для 1920 г. эти значения не определены из-за отсутствия значений переменных P_0 и X_0).

```
Klein$P.lag <- with(Klein, c(NA, P[-length(P)]))
Klein$X.lag <- with(Klein, c(NA, X[-length(X)]))
Klein$A <- Klein$Year - 1931

head(Klein)
  Year    C    P   Wp    I K.lag    X  Wg   G   T P.lag X.lag   A
1 1920 39.8 12.7 28.8  2.7 180.1 44.9 2.2 2.4 3.4    NA    NA -11
2 1921 41.9 12.4 25.5 -0.2 182.8 45.6 2.7 3.9 7.7  12.7  44.9 -10
3 1922 45.0 16.9 29.3  1.9 182.6 50.1 2.9 3.2 3.9  12.4  45.6  -9
4 1923 49.2 18.4 34.1  5.2 184.5 57.2 2.9 2.8 4.7  16.9  50.1  -8
5 1924 50.6 19.4 33.9  3.0 189.7 57.1 3.1 3.5 3.8  18.4  57.2  -7
6 1925 52.6 20.1 35.4  5.1 192.7 61.0 3.2 3.3 5.5  19.4  57.1  -6

# Подгонка структурных моделей Клейна методом 2SLS.
# Определим формулу для списка инструментальных переменных:
InstFor <- as.formula(~ G + T + Wg + A + P.lag + K.lag + X.lag)
eqn.1 <- tsls(C ~ P + P.lag + I(Wp + Wg), InstFor, data = Klein)

summary(eqn.1)  # Модель 1 потребительской функции
            Estimate Std. Error t value  Pr(>|t|)
(Intercept)  16.5548    1.46798 11.2772 2.587e-09
P             0.0173    0.13120  0.1319 8.966e-01
P.lag         0.2162    0.11922  1.8137 8.741e-02
I(Wp + Wg)    0.8102    0.04474 18.1107 1.505e-12
```

Вызывает подозрение низкая статистическая значимость коэффициента α_{11} при P. Поэтому выполним проверку адекватности уравнения, сравнив остаточные дисперсии модели eqn.1 и нулевой модели с одним свободным членом:

```
anova(eqn.1, tsls(C ~ 1, InstFor, data = Klein))
Analysis of Variance
Model 1:  C ~ P + P.lag + I(Wp + Wg)
Model 2:  C ~ 1
Instruments: ~G + T + Wg + A + P.lag + K.lag + X.lag
        Res.Df    RSS Df Sum of Sq      F    Pr(>F)
Model 1     17  21.93
Model 2     20 941.43  3     919.5 237.65 4.485e-14 ***

# Аналогичным образом выполним подгонку и проверку
#  адекватности остальных двух уравнений
eqn.2 <- tsls(I ~ P + P.lag + K.lag, InstFor, data = Klein)
eqn.3 <- tsls(Wp ~ X + X.lag + A, InstFor, data = Klein)
```

Полученные прогнозные значения эндогенных переменных могут дать информацию для проведения государством рациональной стабилизационной политики.

Важным средством представления и интерпретации структурной модели является отображение ее в виде графа, который часто называют *диаграммой пути* («*path diagram*»; Blau & Duncan, 1967, http://bit.ly/1GbDPeL). Модели, которые могут быть представлены графом пути, называют *рекурсивными*, и к ним предъявляются некоторые естественные требования: отсутствие взаимонаправленных связей или обратных петель, взаимная независимость коэффициентов модели и др.

Рассмотрим процесс построения диаграмм пути, заодно показав возможности другого известного пакета – lavaan, включающего функции конфирматорного факторного анализа («*confirmatory factor analysis*», CFA), построения структурных уравнений и моделирования латентного роста («*latent growth curve models*»). Обратимся к еще одному интересному примеру (Bollen, 1989, http://bit.ly/1K05C2Q), который посвящен анализу связи между степенью индустриализации страны и уровнем демократизации политической жизни. Таблица PoliticalDemocracy с исходными данными, включенная в пакет lavaan, содержит сведения по 75 странам в виде набора из 11 переменных за два фиксированных периода – 1960 и 1965 гг. (указаны в скобках за обозначениями переменных), в том числе:

- x_1 (1960) – валовой национальный продукт (ВНП) на душу населения;
- x_2 (1960) – потребление индустриальной энергии на душу населения;
- x_3 (1960) – процент рабочей силы, занятой в промышленности;
- y_1 (1960), y_5 (1965) – экспертная оценка свободы прессы;
- y_2 (1960), y_6 (1965) – свобода политической оппозиции;
- y_3 (1960), y_7 (1965) – справедливость выборов;
- y_4 (1960), y_8 (1965) – эффективность избранного законодательного органа.

Первым шагом является определение переменных структурной модели:

```
library(lavaan)
head(PoliticalDemocracy)
     y1       y2       y3       y4       y5       y6       y7       y8       x1       x2       x3
1  2.50 0.000000 3.333333 0.000000 1.250000 0.000000 3.726360 3.333333 4.442651 3.637586 2.557615
2  1.25 0.000000 3.333333 0.000000 6.250000 1.100000 6.666666 0.736999 5.384495 5.062595 3.568079
3  7.50 8.800000 9.999998 9.199991 8.750000 8.094061 9.999998 8.211809 5.961005 6.255750 5.224433
4  8.90 8.800000 9.999998 9.199991 8.907948 8.127979 9.999998 4.615086 6.285998 7.567863 6.267495
5 10.00 3.333333 9.999998 6.666666 7.500000 3.333333 9.999998 6.666666 5.863631 6.818924 4.573679
6  7.50 3.333333 6.666666 6.666666 6.250000 1.100000 6.666666 0.368500 5.533389 5.135798 3.892270

model <- '
# вычисляемые компоненты
ind60 =~ x1 + x2 + x3
dem60 =~ y1 + y2 + y3 + y4
dem65 =~ y5 + y6 + y7 + y8

# регрессия
dem60 ~ ind60
dem65 ~ ind60 + dem60

# корреляции остатков
y1 ~~ y5
y2 ~~ y4 + y6
y3 ~~ y7
y4 ~~ y8
y6 ~~ y8'
```

Спецификация модели `model`, принятая в рамках пакета `lavaan`, позволяет в явной и наглядной форме представить ее структуру. Как и в обычных формулах среды R, знак «~» означает оператор регрессии. Слева от него представлены зависимые эндогенные переменные, которыми в данном примере являются `dem60` и `dem65` - уровень демократизации общества в 1960 и 1965 гг. соответственно. Справа могут быть как экзогенные (наблюдаемые) переменные, так и другие латентные компоненты, в частности уровень индустриализации страны в 1960 г. – `ind60`.

Другой блок уравнений определяет способ вычисления латентных переменных. Оператор «=~» имеет смысл утверждения «рассчитывается как». Справа от него представлен список независимых переменных.

Наконец, третий блок задает схему корреляционных отношений между экзогенными переменными, которые связываются между собой оператором «~~» (двойная тильда). В этом примере разумно предположить, что одни и те же показатели демократизации за оба сравниваемых периода времени имеют значимую ковариацию остатков. В спецификацию также включена информация о возможности связи между степенью свободы для оппозиции и эффективностью выбираемого парламента.

Функция **`sem`**() выполняет оценку параметров модели `model`. По умолчанию для оптимизации искомых коэффициентов используется метод максимального прав-

доподобия, но доступны также и различные версии метода наименьших квадратов. Результаты можно получить при помощи стандартной функции **summary**() или в более развернутом виде при помощи **parameterEstimates**():

```
fit <- sem(model, data=PoliticalDemocracy)
parameterEstimates(fit)
```

	lhs	op	rhs	est	se	z	pvalue	ci.lower	ci.upper
1	ind60	=~	x1	1.000	0.000	NA	NA	1.000	1.000
2	ind60	=~	x2	2.180	0.139	15.742	0.000	1.909	2.452
3	ind60	=~	x3	1.819	0.152	11.967	0.000	1.521	2.116
4	dem60	=~	y1	1.000	0.000	NA	NA	1.000	1.000
5	dem60	=~	y2	1.257	0.182	6.889	0.000	0.899	1.614
6	dem60	=~	y3	1.058	0.151	6.987	0.000	0.761	1.354
7	dem60	=~	y4	1.265	0.145	8.722	0.000	0.981	1.549
8	dem65	=~	y5	1.000	0.000	NA	NA	1.000	1.000
9	dem65	=~	y6	1.186	0.169	7.024	0.000	0.855	1.517
10	dem65	=~	y7	1.280	0.160	8.002	0.000	0.966	1.593
11	dem65	=~	y8	1.266	0.158	8.007	0.000	0.956	1.576
12	dem60	~	ind60	1.483	0.399	3.715	0.000	0.701	2.265
13	dem65	~	ind60	0.572	0.221	2.586	0.010	0.139	1.006
14	dem65	~	dem60	0.837	0.098	8.514	0.000	0.645	1.030
15	y1	~~	y5	0.624	0.358	1.741	0.082	-0.079	1.326
16	y2	~~	y4	1.313	0.702	1.871	0.061	-0.063	2.689
17	y2	~~	y6	2.153	0.734	2.934	0.003	0.715	3.591
18	y3	~~	y7	0.795	0.608	1.308	0.191	-0.396	1.986
19	y4	~~	y8	0.348	0.442	0.787	0.431	-0.519	1.215
20	y6	~~	y8	1.356	0.568	2.386	0.017	0.242	2.470
21	x1	~~	x1	0.082	0.019	4.184	0.000	0.043	0.120
22	x2	~~	x2	0.120	0.070	1.718	0.086	-0.017	0.256
23	x3	~~	x3	0.467	0.090	5.177	0.000	0.290	0.643
24	y1	~~	y1	1.891	0.444	4.256	0.000	1.020	2.762
25	y2	~~	y2	7.373	1.374	5.366	0.000	4.680	10.066
26	y3	~~	y3	5.067	0.952	5.324	0.000	3.202	6.933
27	y4	~~	y4	3.148	0.739	4.261	0.000	1.700	4.596
28	y5	~~	y5	2.351	0.480	4.895	0.000	1.410	3.292
29	y6	~~	y6	4.954	0.914	5.419	0.000	3.162	6.746
30	y7	~~	y7	3.431	0.713	4.814	0.000	2.034	4.829
31	y8	~~	y8	3.254	0.695	4.685	0.000	1.893	4.615
32	ind60	~~	ind60	0.448	0.087	5.173	0.000	0.279	0.618
33	dem60	~~	dem60	3.956	0.921	4.295	0.000	2.151	5.762
34	dem65	~~	dem65	0.172	0.215	0.803	0.422	-0.249	0.593

Используя функцию **fitMeasures**(), можно выполнить серию проверок на адекватность модели различными методами и получить 36 тестовых показателей. Ограничимся получением результатов тестирования по критерию χ^2 и величине корня из среднеквадратичной ошибки аппроксимации (RMSEA, от «*root mean square error of approximation*») с выводом соответствующих им *p*-значений, а также знакомых нам информационных критериев (AIC, BIC) и индекса качества подгонки

тестируемой модели по сравнению с «максимально насыщенной» (CFI, от «*comparative fit index*»):

```
fitMeasures(fit, c("chisq", "df", "pvalue", "aic", "bic",
                   "cfi", "rmsea", "rmsea.pvalue"))
```

```
   chisq       df   pvalue      aic      bic
  38.125   35.000    0.329 3157.582 3229.424
     cfi    rmsea rmsea.pvalue
   0.995    0.035        0.611
```

Другие подробности анализа этих данных см. в статье Rosseel (2012, http://bit.ly/1cS0OSn).

Изучение длинных списков коэффициентов системы линейных уравнений множественной регрессии и ковариационных соотношений является весьма утомительным занятием (даже для нашего скромного примера число оцененных параметров составляет 34). Анализ модели становится нагляднее, если полученные результаты отобразить на диаграмме пути. Выполнить визуализацию графов структурных моделей можно с использованием функции **semPaths**() из пакета semPlot. Многочисленные графические параметры функции позволяют регулировать объем включаемой информации, цвет, форму и другие атрибуты создаваемого изображения:

```
# классическая форма диаграммы
semPaths(fit, "model", "est", style = "lisrel")
```

На рис. 137 латентные переменные обозначены кругами, а наблюдаемые – прямоугольниками, причинно-следственная пара соединяется односторонней стрелкой, исходящей от независимой к зависимой переменной. Ковариационные связи обозначаются двухсторонними стрелками. Над стрелками проставлены значения коэффициентов модели.

Вид другой диаграммы (рис. 138) может понравиться, например, влюбленным аналитикам:

```
# Привет от св. Валентина
semPaths(fit, "std", "hide",
       sizeLat = 15, shapeLat = "star",
       shapeMan = "heart",
       col = list(man = "pink", lat = "yellow"),
       residuals = FALSE,
       borders = FALSE,edge.color = "purple",
       XKCD = TRUE, edge.width = 2,
       rotation = 2, layout = "tree2",
       fixedStyle = 1, mar = c(1, 3, 1, 3))
```

Для читателя, интересующегося применением структурных моделей в биологии и экологии, рекомендуем статью Grace & Pugesek (1997, http://bit.ly/1D6WdRa), в которой авторы предприняли попытку прогноза видового богатства растительности в штате Луизиана.

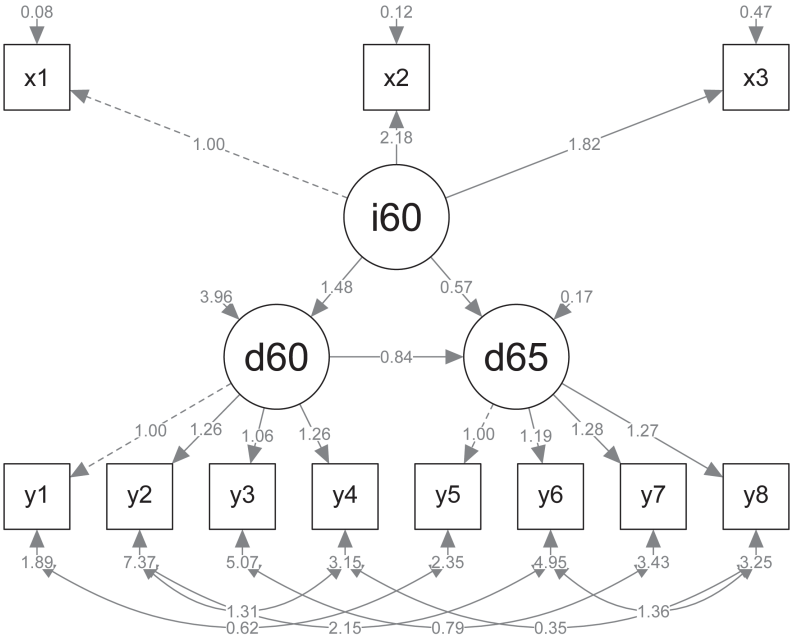


Рисунок 137

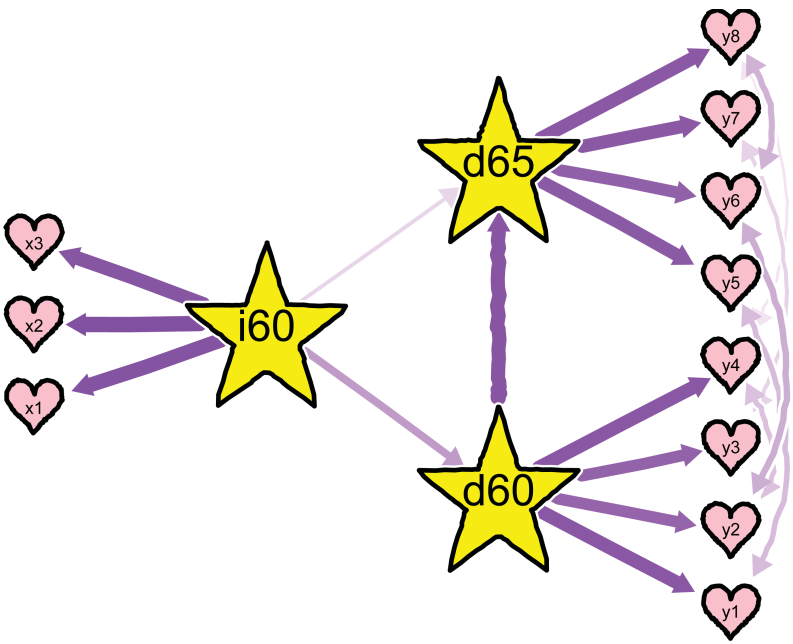


Рисунок 138

Глава 9

Пространственный анализ и создание картограмм

Изучение расположения и взаимосвязи объектов в пространстве играет очень важную роль при обработке результатов наблюдений. Можно даже сказать категоричнее – большинство социальных или экологических исследований является частным случаем анализа пространственных структур. На современном этапе пространственно-временные технологии моделирования систем в различных масштабах рассматриваются как основной путь интегрального анализа и обобщения всего массива данных о состоянии природы и общества.

Статистическая среда R имеет весьма продвинутый инструментарий пространственного анализа, обеспечивающий геостатистическое моделирование, визуализацию, связь с современными ГИС-системами и интернет-ресурсами. Со списком основных пакетов можно ознакомиться в соответствующем разделе на сайте http://cran.r-project.org. Некоторые возможности их использования мы рассмотрим в настоящей главе. Следует подчеркнуть, что пространственный анализ с помощью R – это огромная тема, изложение которой легко может занять по объему несколько отдельных книг. Наша цель – привести лишь некоторые примеры, связанные либо с нашими собственными исследованиями, либо с исследованиями наших коллег.

9.1. Простая карта: использование растрового рисунка и расчет расстояний

Рассмотрим простой пример нанесения точек на карту. В шести областях европейской части России был проведен отлов обыкновенной гадюки *Vipera berus* (исследования лаборатории герпетологии и токсинологии ИЭВБ РАН, http://bit.ly/1PTCoXR). У каждого из отловленных экземпляров измеряли морфологические признаки (всего 10 показателей) и в лабораторных условиях определяли протеолитическую активность яда.

Создадим растровый рисунок в формате JPG путем выкопировки нужного фрагмента из любого географического атласа, которые во множестве представлены в Интернете. Пусть размеры рисунка $X_{ras} = 1015$ и $Y_{ras} = 635$ пикселей. Откроем в среде R новое графическое окно с использованием функций **readJPEG**() и **rasterImage**(). Укажем координаты мест отлова змей (условно совмещая их

с точками областных центров) при помощи функции **locator**(). Тут важно сделать в точности 6 кликов мышью, сохраняя выбранную последовательность прохода по регионам:

```
library(jpeg)
image <- readJPEG("maps.yandex.ru.JPG")
Xras = 1015; Yras = 635
Reg <- c("Вологда", "Пермь", "Казань", "Самара", "Пенза", "Липецк")

par(mar = c(0, 0, 0, 0))
plot(1, xlim = c(0, Xras), ylim = c(0, Yras), xlab = "", ylab = "")
lim <- par()

rasterImage(image, lim$usr[1], lim$usr[3], lim$usr[2], lim$usr[4])

xy <- locator(6)

xy

$x
[1] 152.4807 959.6825 607.1545 660.7189 402.8627 138.7782
$y
[1] 619.51773 500.95917 301.65812  88.04829  86.00417  33.87928
```

Предположим, что по результатам статистического анализа важным для нас показателем является протеолитическая активность яда PA. Нанесем на нашу карту круги, диаметр которых пропорционален величине этого признака (рис. 139). Чтобы были видны надписи на карте, сделаем эти круги полупрозрачными (параметр alfa = 0.6):

```
PA <- c(21.7, 19.1, 16.4, 16.7, 9.4, 6.2)
mycex = ((PA - min(PA))^2/max(PA) + 2)
colpts = rgb(0.2, 0.5, 0.4, alpha = 0.6)
points(xy$x, xy$y, cex = mycex, col = 1, pch = 21, bg = colpts)
```

Очевидно, что активность яда гадюк увеличивается в направлении северо-востока.

Использование географических расстояний в статистическом анализе

Найденные нами координаты точек на карте могут быть использованы для оценки расстояний между ними:

```
df <- data.frame(xy)
dist(df)
         1        2        3        4        5
2 816.8005
3 554.1790 406.5527
```

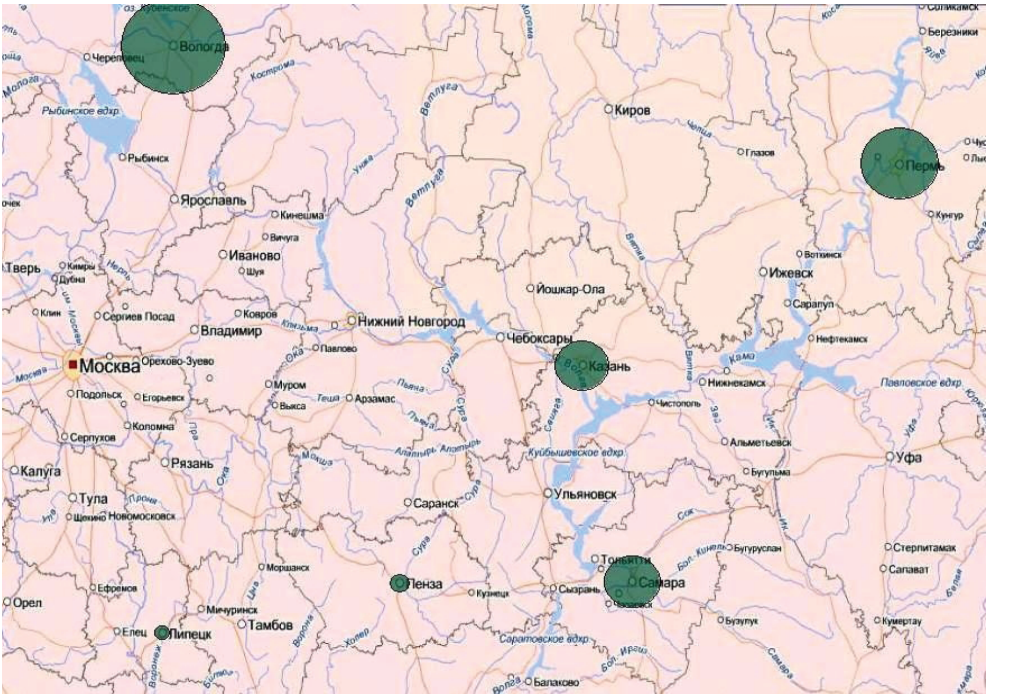


Рисунок 139

```
4 736.2293 511.4361 221.5197
5 589.8756 695.6555 296.9448 257.8642
6 582.7054 943.9776 537.4254 524.4364 268.6027
```

Мы получили матрицу относительных («пиксельных») дистанций, которые, безусловно, неточны хотя бы потому, что не учитывают неравные размеры сторон пикселя. Однако имеющейся точности вполне достаточно для реализации некоторых аспектов статистического анализа.

Предположим, что имеются две матрицы расстояний **A** и **B**, относящиеся к одному и тому же фиксированному набору объектов (например, одна матрица может содержать генетические расстояния между некоторыми видами, найденные молекулярно-биологическими методами, а другая – географические расстояния между условными центрами ареалов тех же видов). Тест Мантеля («*Mantel's test*», или «*quadratic assignment*») проверяет гипотезу H_0 от том, что расстояния между объектами в матрице **A** независимы от расстояний между теми же самыми объектами в матрице **B**. Тест оценивает, по сути, силу связи между многомерными структурами данных, вычисляя коэффициент линейной корреляции двух матриц, составленных на основе меры расстояния любой природы.

Если найти сумму элементов в матрице **Z** = **AB**, которая является произведением обеих сравниваемых матриц расстояний **A** и **B**, исключая при этом элементы на

главной диагонали, то получим Z-статистику Мантеля, то есть $Z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{ij} b_{ij}$. Стандартизованная статистика Мантеля изменяется от +1 до –1 и соответствует коэффициенту корреляции Пирсона r между матрицами расстояний **A** и **B**.

С использованием теста Мантеля можно, например, оценить обоснованность следующих содержательных гипотез:

- связана ли статистически значимой зависимостью плотность изучаемых популяций с географическим расстоянием между их местообитаниями;
- имеется ли взаимосвязь между двумя группами объектов, обитающих в одних и тех же пунктах (например, между числом врачей-хирургов и гангстеров);
- имеются ли изменения в структуре сообщества до и после внесения некоторого возмущения.

Пусть, например, мы имеем матрицу евклидовых расстояний между 6 изученными регионами, оцененную с использованием 10 средних стандартизированных морфологических показателей гадюки *Vipera berus* (процедуру ее расчета мы здесь не приводим):

```
Rmat <- matrix(c(


0,          0.87763393,  1.37359643,  1.07033575,  1.26727796,  1.62654352,
0.87763393, 0,           1.03383327,  1.01546645,  1.12793326,  1.6236099,
1.37359643, 1.03383327,  0,           1.22476184,  1.09088194,  1.39409792,
1.07033575, 1.01546645,  1.22476184,  0,           1.18241739,  1.87482882,
1.26727796, 1.12793326,  1.09088194,  1.18241739,  0,           1.57430208,
1.62654352, 1.6236099,   1.39409792,  1.87482882,  1.57430208,  0
), nrow = 6)

Rdist <- as.dist(Rmat)
```

Выполним тест Мантеля с использованием функции **mantel**() из пакета vegan:

```
library(vegan)
mantel(Rdist, dist(df))
Mantel statistic based on Pearson's product-moment correlation
Call:
mantel(xdis = Rdist, ydis = dist(df))
Mantel statistic r: 0.02597
      Significance: 0.471
Empirical upper confidence limits of r:
  90%   95% 97.5%   99%
0.469 0.518 0.557 0.618
Based on 999 permutations
```

Статистика Мантеля оказалась незначимой, что заставляет предположить два возможных объяснения: либо гадюки с разными морфометрическими признаками равновероятно распределены по изученной территории, либо эта зависимость

имеет существенно нелинейный характер (то есть нельзя подобрать гиперплоскость, гладко аппроксимирующую наблюдаемые данные).

Поскольку есть все основания предположить второе, то используем один из геометрических подходов, применяемых в биогеографии и генетике, который позволяет выявить особенности изменения структуры объектов в реальных пространственных координатах («*isolation by distance*», IBD). В нашем случае речь идет об *идентификации «барьеров»*, то есть о выделении границ, относительно которых параметры изучаемых сообществ претерпевают резкие характерные изменения.

Одной из базовых структур вычислительной геометрии является триангуляция Делоне, которая отображает множество точек на плоскости сетью треугольников (каждое подпространство является изолированным, границы являются непересекающимися, сумма длин всех ребер минимальна среди всех возможных триангуляций). После построения сети, объединяющей окрестности всех географических точек, каждое звено сети может быть связано со значением расстояния матрицы дистанций, полученной по данным наблюдений (на рис. 140 – это числа в квадрате). Алгоритм *максимума различий Монмоньера* (Monmonier, 1973, http://bit.ly/1PAPrw3) используется для идентификации границы области, где расстояния между смежными парами изучаемых объектов являются наибольшими.

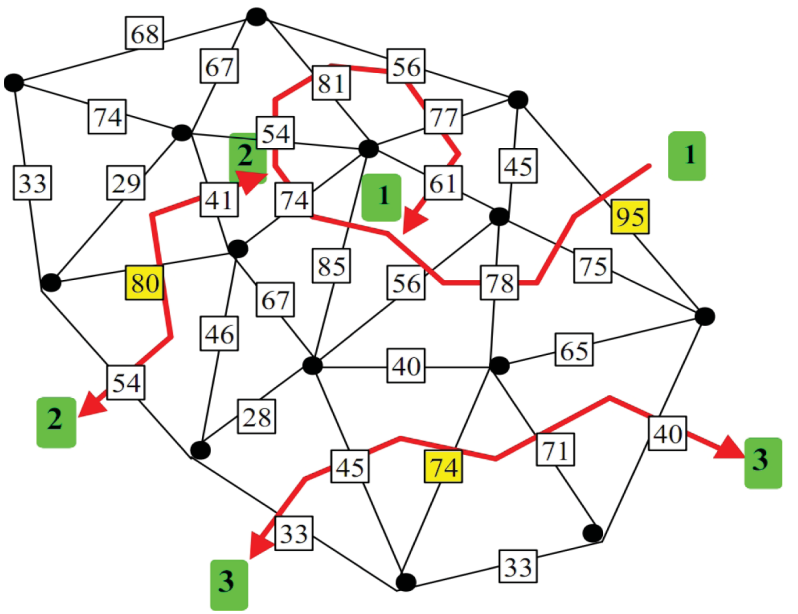


Рисунок 140

Перебор обычно начинается с краевого звена с максимальным значением расстояния и продолжается через прилегающие звенья, пока формируемая граница не достигнет пределов триангуляции или замкнется на себя, формируя контур

вокруг совокупности точек. Оценить статистическую значимость вычисленных барьеров можно с использованием рандомизационного теста – подробности см. в статье Manni et al. (2004, http://bit.ly/1IWis1t) и руководстве к программе Barrier на сайте http://bit.ly/1PAQOeh.

Получим для имеющихся географических точек сеть триангуляции Делоне с использованием функции **chooseCN**() из пакета adegenet. Функция **monmonier**() позволяет нам найти две максимальные границы (nrun = 2), разделяющие наиболее различающиеся между собой популяции гадюк согласно матрице наблюдений Rdist. Полученный рисунок (рис. 141) выведем в PDF-файл:

```
library(adegenet)
cn <- chooseCN(df, type = 1)
mon <- monmonier(df, Rdist, cn, threshold = 1.2, nrun = 2)

pdf("Result.pdf")
plot(mon, method = "greylevel", add.arr = TRUE, col = "red")
points(df, pch = 21, cex = 1.2, bg = "green")
text(df, Reg, adj = c(0, 0))
graphics.off()

shell.exec("Result.pdf") # Открываем файл со схемой
```

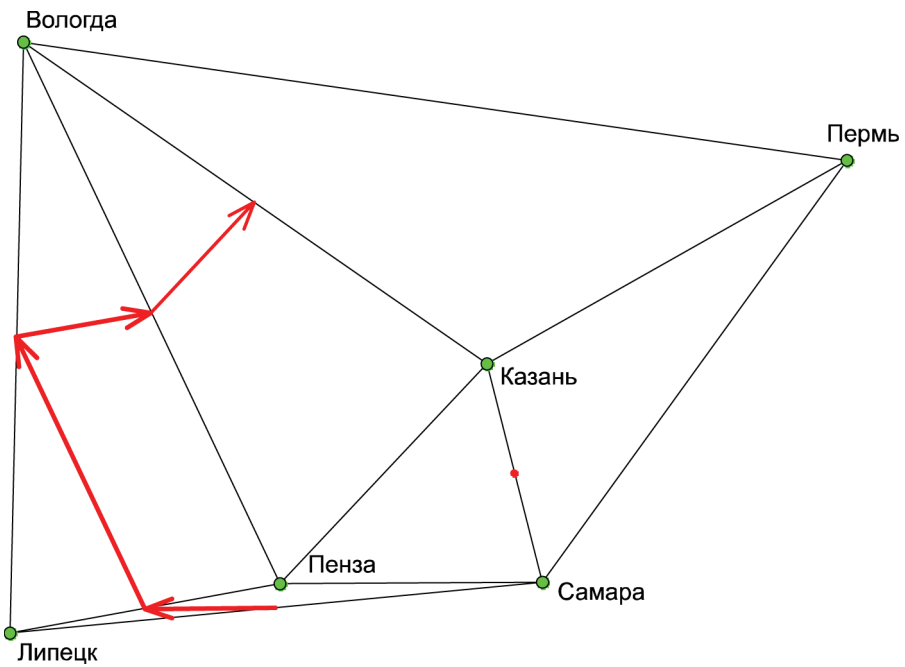


Рисунок 141

Первый основной барьер отделяет от других регионов липецкую популяцию. Вторая граница проходит на северо-восток, подчеркивая тем самым специфичную морфологию вологодских змей. К сожалению, приведенный пример не слишком характерен из-за своей простоты, поэтому в полной мере со смыслом алгоритма и примерами его работы можно ознакомиться по приведенным выше источникам или книге Шитиков и др. (2011, http://bit.ly/1GuE3jT).

Расчет расстояния между объектами по их географическим координатам

Приведенный выше пример оценивает приближенные относительные расстояния и непригоден, когда, например, прикидываешь запас бензина для автомобильной поездки. Более точно расстояния между точками на географической карте можно рассчитать по их координатам широты и долготы на земной поверхности.

Предположим, что по каким-то причинам нас интересуют четыре местоположения на территории... острова Мадагаскар (мы всегда хотели побывать на этом острове – отсюда и выбор примера), показанные ниже в виде точек красного цвета. Приведенная на рис. 142 карта была построена при помощи следующего кода:

```
library(sp)
library(maptools)
library(mapproj)
library(ggplot2)
load(url("http://biogeo.ucdavis.edu/data/gadm2/R/MDG_adm0.RData"))

madagascar <- fortify(gadm)
m <- ggplot() + geom_map(data = madagascar,
                         aes(map_id = id),
                         map = madagascar,
                         fill = "white", color = "black") +
   expand_limits(x = madagascar$long, y = madagascar$lat) +
   coord_map("mercator") +
   xlab("Долгота") + ylab("Широта") + theme_bw()
mp <- m + geom_point(data = data.frame(Lon = c(45.0, 47.2,
         45.5, 48.5), Lat = c(-25.0, -21.0, -17.0, -15.0)),
         aes(Lon, Lat), color = "red", size = 3)
print(mp)
```

Для начала рассчитаем расстояние между двумя наиболее удаленными точками. Расчет расстояний между географическими объектами по их координатам возможен при помощи нескольких пакетов R. Мы воспользуемся пакетом geosphere, в состав которого входит большой набор функций, реализующих методы пространственной тригонометрии. В частности, в этом пакете имеется несколько функций для расчета кратчайшего расстояния между двумя точками на поверхности Земли разными методами: **distCosine**(), **distVincentySphere**(), **distVincentyEllipsoid**(), **distHaversine**(), **distMeeus**(). Применим функцию **distHaversine**(), которая рас-

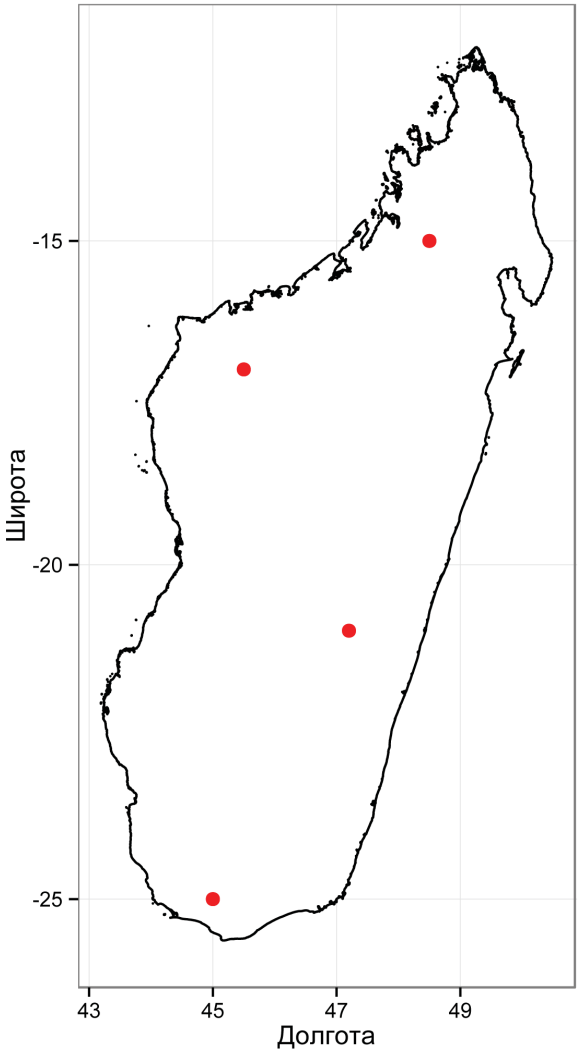


Рисунок 142

считывает расстояние по широко используемой формуле гаверсинуса (http://bit.ly/1E4fB5o). Эта функция имеет следующий синтаксис:

```
distHaversine(p1, p2, r = 6378137),
```

где `p1` – координаты точки (или точек), подаваемые либо в виде вектора из двух значений (долгота, широта), либо в виде матрицы с двумя столбцами (первый столбец – долгота, второй – широта); `p2` – то же, что и `p1`; `r` – радиус Земли, выраженный в метрах. Для нашего примера получаем:

```
library(geosphere)
# делим на 1000 для выражения расстояния в км:
distHaversine(c(45.0, -25.0), c(48.5, -15.0))/1000
[1] 1171.7
```

Следует отметить, что полученный результат основан на предположении, что Земля имеет форму шара. Однако известно, что это не совсем верно – Земля представляет собой эллипсоид. При расчете расстояния между существенно удаленными объектами это обстоятельство может сказаться на точности результата. В таких случаях стоит воспользоваться функцией **distVincentyEllipsoid**().

Теперь несколько усложним задачу и предположим, что требуется рассчитать расстояния между всеми имеющимися парами точек. Необходимость такого рода расчетов часто возникает, например, в экологии при анализе пространственного распределения особей того или иного вида, когда требуется вычислить среднее расстояние между особями в целом. Сохраним координаты имеющихся точек в виде матрицы:

```
coords <- cbind(c(45.0, 47.2, 45.5, 48.5),
                c(-25.0, -21.0, -17.0, -15.0))
coords
     [,1] [,2]
[1,] 45.0  -25
[2,] 47.2  -21
[3,] 45.5  -17
[4,] 48.5  -15
```

Теперь мы можем легко рассчитать все пары расстояний, воспользовавшись базовой функцией **apply**():

```
Dist <- apply(coords, 1, FUN =
    function(eachPoint) distHaversine(eachPoint, coords)/1000)

Dist
          [,1]     [,2]     [,3]      [,4]
[1,]    0.0000 499.0594 892.0671 1171.6513
[2,]  499.0594   0.0000 479.8662  681.9332
[3,]  892.0671 479.8662   0.0000  390.6508
[4,] 1171.6513 681.9332 390.6508    0.0000
```

Как видим, в итоговой квадратной матрице информация о расстояниях между точками дублируется по обе стороны от главной диагонали. Хорошо ли это, зависит от поставленной задачи. Например, при выполнении кластерного анализа многие соответствующие функции без проблем примут такую матрицу. Однако при необходимости вычислить среднее расстояние (или какие-либо другие статистики) потребуются только те значения, которые расположены сверху или снизу от главной диагонали матрицы. Эти значения можно легко извлечь при помощи базовых функций **lower.tri**() или **upper.tri**():

```
Dist <- Dist[lower.tri(Dist)]
Dist
[1]  499.0594  892.0671 1171.6513  479.8662  681.9332  390.6508
```

Рассчитать среднее расстояние теперь очень легко:

```
mean(Dist)
[1] 514.4
```

9.2. Анализ пространственного размещения точек

Под *пространственным анализом* («*spatial analysis*») понимают исследование таких ситуаций, когда случайная переменная Z связана с некоторым изучаемым показателем, изменяющимся в пространстве и характеризующим динамику изучаемой системы, а остальные независимые переменные определяют пространственно-временное положение объекта или точки наблюдения (время t, x-y-координаты, высоту h и т. д).

При изучении пространственных структур принято выделять два направления: *геостатистический анализ* и *анализ пространственного размещения точек*, – которые отличаются методом получения исходных данных. В первом случае исследователь для анализа пространственного распределения какого-то показателя (например, содержания тяжелых металлов в верхнем слое почвы) *сам планирует* точки, в которых будут проводиться наблюдения. Полученные выборочные значения пространственной переменной $z(x)$ рассматриваются как возможные реализации случайной функции $Z(x)$. Теоретические аспекты геостатистического анализа и примеры его реализации в R см. в книге Савельев и др. (2012), представленной на сайте сообщества «Гис-Лаб» (http://bit.ly/1brV8NN).

Во втором случае также анализируются свойства определенного участка пространства по выборочным данным из разных его точек, однако местоположение этих точек устанавливается не по воле исследователя, а *случайно назначается* неким природным «механизмом» (например, число яиц в гнезде можно подсчитать только там, где есть гнездо). Иными словами, изучаемое явление трактуется как не зависящий от наблюдателя процесс, генерирующий поток дискретных событий. О свойствах этого процесса мы можем судить по характеру размещения на координатной сетке ансамбля точек, соответствующих произошедшим событиям (Савельев и др., 2014, http://bit.ly/1HDFLhH).

Описание процесса, генерирующего пространственные данные, обычно сводится к отнесению конкретной совокупности точек на изучаемом участке пространства к одному из возможных типов размещения: *а*) регулярное (детерминированное равномерное или стохастическое равномерное); *б*) групповое, или агрегированное (детерминированное неравномерное); *в*) неоднородное случайное и, наконец, *г*) полностью случайное размещение («*complete spatial randomness*», CSR). Проверка гипотезы о наличии CSR обычно проводится путем имитации данных на основе *однородного процесса Пуассона*, то есть проверяется условие о том, что количество точек, попавших в любую из непересекающихся областей пространства Ω, явля-

ется независимой *случайной величиной*. Другое условие однородности удовлетворяется, если вероятность того, что в некоторую область A, $A \subset \Omega$, попадет ровно n точек, не зависит от положения области A на плоскости или от ее формы.

Рассмотрим пример, заимствованный из семинарского занятия Н. Чижиковой (2012) на конференции «Открытые ГИС!» (http://gisconf.ru). Текст доклада «*Введение в разведочный анализ точечных образов с помощью пакета* `spatstat` *R*» можно скачать в разделе «Мастер-классы» на странице http://bit.ly/1HDGLlP, а скрипты и исходные данные для презентации находятся в архиве по адресу http://bit.ly/1HDNL3P.

Таблица с данными содержит пространственные координаты точек `x` и `y`, где были обнаружены цветковые растения *Adonis vernalis L.* (адонис весенний), количество цветков `blossoms`, а также категориальная переменная `age` (возраст), которая принимает значения `gene` (генеративный) и `pre`. С помощью функции **`ppp`**`()` из пакета `spatstat` создадим объект типа `ppp` («*point pattern*»), который содержит всю необходимую информацию для последующего анализа:

```
library(spatstat)
# Загружаем текстовый файл с данными:
plot_13 <- read.table(file = "13.txt", header = T, sep = "\t")
# Вычислим размеры площадки по осям X и Y:
xmin <- floor(min(plot_13$x))
xmax <- ceiling(max(plot_13$x))
ymin <- floor(min(plot_13$y))
ymax <- ceiling(max(plot_13$y))

# Создадим объект класса ppp:
ppp_object <- ppp(x = plot_13$x, y = plot_13$y,
                  marks = data.frame(age = plot_13$age,
                  blossoms = plot_13$blossoms),       # метки
         window = owin(c(xmin, xmax), c(ymin, ymax),  # "окно"
         unitname = c("metre", "metres")))   # единицы измерения
```

Простейший способ показать графически характер распределения точек – это нанести их на сетку квадратов. Выбрать оптимальные размеры ячейки сетки при площади всего участка A = 10×10 м^2 и общей численности наблюдаемых точек N = 126 можно, например, с использованием эмпирического правила «большого пальца»:

Сторона квадрата $L = (2{\times}A\ /\ N)^{0.5} = (2{\times}10{\times}10\ /\ 126)^{0.5} = 1.17$ м.

Эмпирические частоты в каждом k-ом квадрате (представлены на рис. 143 ниже) при справедливости гипотезы о случайном размещении точек будут хорошо описываться распределением Пуассона $Np_k = N(\lambda^k e^{-\lambda})/k!$. Здесь несмещенной оценкой интенсивности процесса λ является среднее число точек на единицу площади: $\lambda = n_{pois}(k)N/A$. На рис. 143 кружочками показаны цветущие (`age = gene`), а треугольниками – несозревшие растения (`age = pre`).

```
# площадь участка:
A <- area.owin(ppp_object$window)
# число точек:
N <- ppp_object$n
# сторона одной ячейки, в которую попадают в среднем 2 точки:
L <- sqrt(2*A/N)
# сколько ячеек уместится в сторону площадки 10×10 м?
quadnum <- round(10/L)
# построим сетку ячеек:
ppp_quadrats<-quadrats(ppp_object, nx = quadnum, ny = quadnum)
# подсчитаем число точек в ячейках, округлим до 1 десятичного знака:
quadcount <- quadratcount(ppp_object, tess = ppp_quadrats)
quadcount <- round(quadcount, 1)

# ----- изобразим точки и их число в каждом квадрате ------
# вычислим центры ячеек:
xgrid <- quadrats(ppp_object, nx = quadnum, ny = quadnum)$xgrid
ygrid <- quadrats(ppp_object, nx = quadnum, ny = quadnum)$ygrid
image(xgrid, ygrid,
      t(quadcount[order(1:quadnum, decreasing = T), ]),
      col = colorRampPalette(c("white", "green"))(15),
      axes = F, asp = 1, xlab = "", ylab = "",
      main = "Число точек в квадратах")
plot(quadcount, add = T, cex = 0.7)
axis(2, las = 1, pos = 0, at = seq(0, 10, 2), cex.axis = 1)
axis(1, las = 1, pos = 0, at = seq(0, 10, 2), cex.axis = 1)
axis(3, pos = 10, at = seq(0, 10, 2), labels = F, cex.axis = 1)
axis(4, pos = 10, at = seq(0, 10, 2), labels = F, cex.axis = 1)
#добавим точки:
plot(ppp_object, which.marks = "age", chars = c(19, 24), cex = 0.7, add = T)
```

Проверку гипотезы о равенстве эмпирических и теоретических вероятностей H_0: $p = p_{\text{pois}}$ можно выполнить, например, при помощи критерия согласия χ^2:

```
quadrat_test_result <- quadrat.test(ppp_object, nx = 9, ny = 9)
round(quadrat_test_result$expected, 2) # Частоты Пуассона
[1] 1.81 1.81 … 1.81 1.81

quadrat_test_result
        Chi-squared test of CSR using quadrat counts
data:  ppp_object
X-squared = 127.9592, df = 80, p-value = 0.0005321
Quadrats: 9 by 9 grid of tiles
```

Второе проверяемое предположение заключается в оценке *однородности* (стационарности) пуассоновского процесса: число точек в любых двух непересекающихся частях области исследования является случайной и независимой перемен-

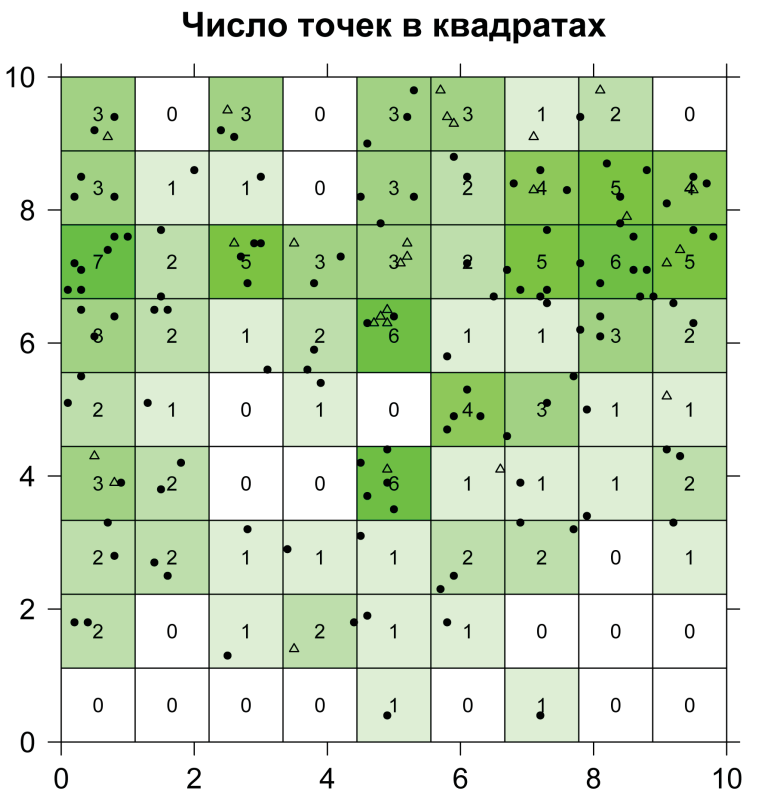


Рисунок 143

ной. Пространственная версия теста Колмогорова-Смирнова позволяет проверить, равномерно ли распределено число точек вдоль пространственных осей x или y:

```
kstest(ppp_object, "x")  # Тест Колмогорова-Смирнова по оси X
        Spatial Kolmogorov-Smirnov test of CSR
data:  covariate 'x' evaluated at points of 'ppp_object'
D = 0.0783, p-value = 0.3285
alternative hypothesis: two-sided

kstest(ppp_object, "y")  # Тест Колмогорова-Смирнова по оси Y
        Spatial Kolmogorov-Smirnov test of CSR
data:  covariate 'y' evaluated at points of 'ppp_object'
D = 0.2451, p-value = 4.296e-08
```

Выполненные расчеты D-статистики позволяют сделать очевидный вывод, что размещение популяций изучаемых растений равномерно распределено относительно оси x, но не является стационарным относительно оси y. Отметим, однако, что тест на стационарность имеет смысл, только если в ходе проверки CSR при

критерии теста χ^2 нулевая гипотеза не отклоняется (в нашем случае выполнение теста на стационарность было не вполне оправданным).

В разделе 8.1 мы рассматривали сглаживание точек на плоскости ядерными функциями. Те же модели мы можем использовать для трехмерного (а в общем случае и многомерного) сглаживания. Эти процедуры лежат в основе построения карт локальных плотностей, с помощью которых осуществляется визуальный ГИС-анализ пространственной неоднородности распределения разнообразных переменных.

Выполним построение поверхности интерполяции плотности точек с помощью радиальных гауссовых ядерных функций. Как и при аппроксимации кривых, скользящее «окно» будет сканировать всю пространственную область и суммировать вклады точек в окрестности, связанной с текущим положением курсора. Гладкость полученной поверхности интерполяции зависит от ширины окна (bandwidth), которую можно предварительно оценить по эмпирическим формулам или найти автоматически в ходе кросс-проверки. Интенсивность процесса, генерирующего размещение точек gene из рассматриваемого нами примера, представлена на рис. 144.

```
# извлечем данные для генеративных особей:
gene_ppp <- ppp_object[ppp_object$marks$age=="gene"]
# вычислим размер скользящего окна, bandwidth по правилу Silverman:
sigma <- (sd(gene_ppp$x) + sd(gene_ppp$y))/2
iqr <- (IQR(gene_ppp$x) + IQR(gene_ppp$y))/2

( bandwidth <- 0.9*min(sigma, iqr)*gene_ppp$n^(-1/5) )
[1] 0.9102268

gene_intensity <- density.ppp(gene_ppp, sigma = bandwidth)
plot(gene_intensity, main = "Плотность gene-особей, экз/кв.м")
points(gene_ppp, pch = 19, cex = 0.6)
```

Важной дополнительной возможностью является построение карт пространственно-обусловленного относительного риска (например, чтобы обозначить области, где число заболевших особей превысило какой-то процент). Так, на рис. 145 представлена карта распределения доминирующих возрастных групп нашего растения, позволяющая легко выделить фрагменты территории, где $p = n_{\text{pre}} / n_{\text{gene}} > 0.5$, то есть число экземпляров n_{pre} превышает число генеративных экземпляров n_{gene}:

```
# выделим коды возрастов:
for_relrisk_example <- ppp_object # создадим копию ppp объекта
marks(for_relrisk_example) <- ppp_object$marks$age
# вычислим вероятность появления особей возраста pre:
p <- relrisk(for_relrisk_example, 0.5)

plot(p, main = "Доля группы pre", col = colorRampPalette(
     c("antiquewhite", "aquamarine3", "navyblue"))(100))
# добавим изолинии:
contour(p, nlevels = 5, lwd = seq(from=0.1, to=3, length.out=5), add = T)
```

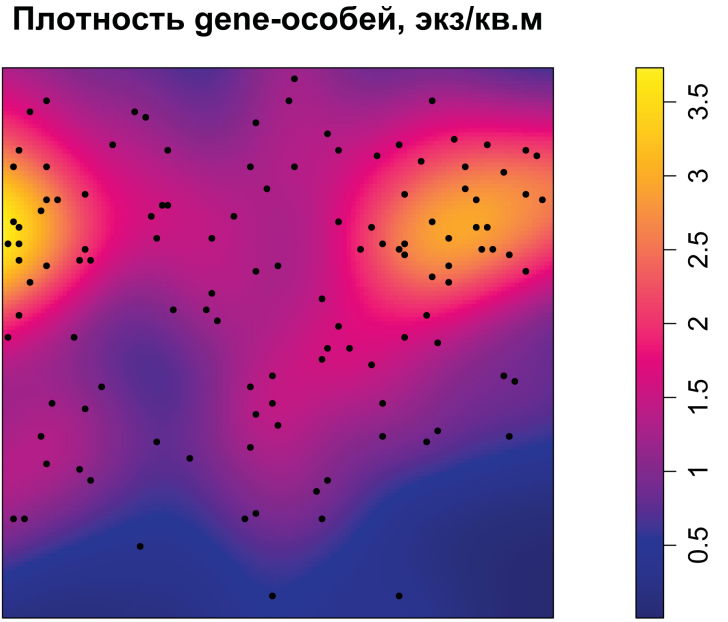


Рисунок 144

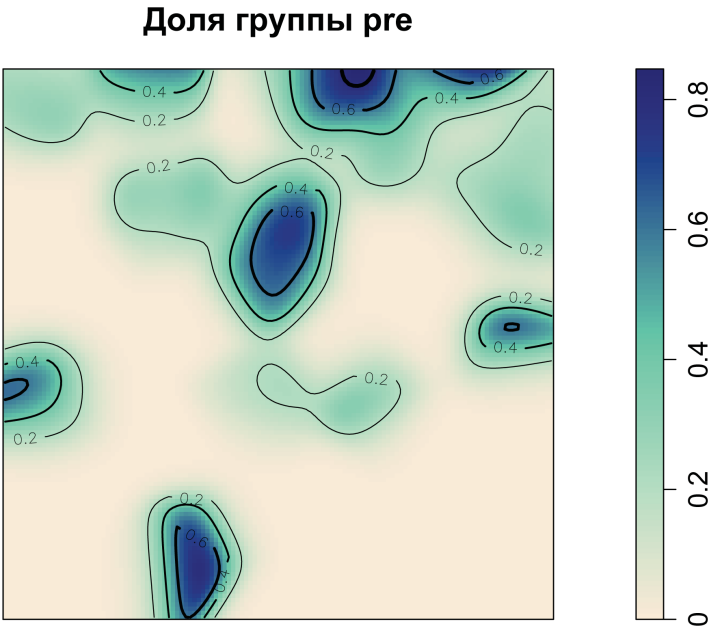


Рисунок 145

9.3. Использование сервисов картографической системы Google Maps

Признанным лидером среди современных картографических систем являются *карты Google* (Google Maps; http://maps.google.ru). Система содержит детализированные спутниковые изображения для обширных регионов любых стран и предоставляет интерактивный доступ к этим картам онлайн. Разработано также большое количество дополнительных сервисов и инструментов (Google Earth, Google Mars, разнообразные погодные и транспортные сервисы, один из самых мощных API).

Многие из этих сервисов реализованы в пакете `googleVis` (http://bit.ly/1Gv5l9U), в состав которого входит также ряд функций, обеспечивающих взаимодействие R с интерфейсом программирования приложений Google Visualization API (http://bit.ly/1DSWrPl). При помощи этого пакета можно сгенерировать html-код карт, который легко будет вставить на страницу веб-сайта.

В общем виде синтаксис функции **`gvisMap`**`()`, создающей карты Google Maps, выглядит следующим образом:

```
gvisMap(data, locationvar = " ", tipvar = " ", options = list(), chartid),
```

где:

- `data` – таблица данных, которая должна содержать как минимум два столбца – с географическими координатами точек (`locationvar`) и с текстом всплывающих подсказок для каждой из этих точек (`tipvar`);
- `locationvar` – имя столбца, содержащего географические координаты точек. Координаты задаются в формате `"широта:долгота"` (см. пример ниже). Вместо географических координат можно также указывать [максимально полный] почтовый адрес, однако авторы пакета `googleVis` рекомендуют этот способ не применять;
- `tipvar` – имя столбца, содержащего текст всплывающих подсказок для каждой точки;
- `options` – список опций, определяющих внешний вид карты;
- `chartid` – текстовая переменная, при помощи которой карте можно присвоить пользовательское имя. По умолчанию данная настройка отключена (в этом случае ID карты генерируется случайным образом автоматически).

Внешний вид карты настраивается при помощи следующих опций:

- `enableScrollWheel` – логическая переменная, по умолчанию принимающая значение `FALSE`. При `enableScrollWheel = TRUE` появится возможность изменять масштаб карты, вращая колесо мыши;
- `showTip` – логическая переменная, по умолчанию равная `FALSE`. Активация этой настройки (`showTip = TRUE`) позволит видеть описание точки при наведении на нее курсора;
- `ShowLine` – позволяет включать (`TRUE`) или выключать (`FALSE`) линию, соединяющую точки на карте;

- `lineColor` – текстовая переменная, задающая цвет вышеупомянутой линии (`"red"`, `"green"` и т. п.; возможно также использование html-цветов, например `"#800000"`);
- `lineWidth` – ширина линии. По умолчанию составляет 10 пикселей. Очевидно, что эта настройка имеет смысл только при `ShowLine = TRUE`;
- `mapType` – текстовая переменная, задающая тип карты. Возможные значения: `"normal"` (обычная схематичная карта), `"terrain"` (режим «Земля»), `"satellite"` («спутник») или `"hybrid"` («гибрид»);
- `useMapTypeControl` – логическая переменная, по умолчанию равная `FALSE`. Значение `TRUE` активизирует меню, позволяющее пользователю выбирать тип карты (см. предыдущий пункт);
- `zoomLevel` – количественная переменная, принимающая целые значения от `0` (виден весь мир) до `19` (максимальное приближение).

Функция **`gvisMap`**`()` возвращает объект класса `gvis`, который представляет собой список, состоящий как минимум из следующих трех элементов: `type` (тип карты), `chartid` (см. выше) и `html`. Именно последний элемент интересует нас больше всего, поскольку он содержит html-код карты.

Работу функции `gvisMap()` мы проиллюстрируем на примере данных из статьи Mastitsky & Makarevich (2007, http://bit.ly/1GtyX4L), посвященной распространению в Беларуси чужеродных видов бокоплавов (http://bit.ly/1ECU3Ai). В ходе выполненного обследования белорусской части реки Днепр был, в частности, обнаружен рачок *Dikerogammarus haemobaphes* – выходец из нижнего течения рек Понто-Каспийского бассейна. Этот вид был отмечен на трех обследованных створах реки. Для начала создадим таблицу (назовем ее `Dikero` – по названию вида), в которой будут храниться координаты створов и их названия:

```
Dikero <- data.frame(Coords = c("52.2:30.6", "53.1:30.1", "53.7:30.3"),
         Location = c("д. Холмеч", "г. Рогачев", "д. Стайки"))

Dikero
     Coords   Location
1 52.2:30.6  д. Холмеч
2 53.1:30.1  г. Рогачев
3 53.7:30.3  д. Стайки
```

Отобразим координаты этих створов на Google-карте. Используя функцию **`gvisMap`**`()`, создадим объект с каким-либо произвольным именем (например, `Map`), в котором будут храниться html-код и другие элементы карты:

```
Map <- gvisMap(Dikero, "Coords", "Location",
               Options = list(showTip = TRUE, mapType = 'normal',
               enableScrollWheel = TRUE),
               chartid = "Dikerogammarus")
```

Команда **`plot`**`(Map)` приведет к открытию браузера, в котором появится созданная Google-карта (естественно, при этом компьютер должен быть подключен к сети Интернет; рис. 146).

```
plot(Map)
```

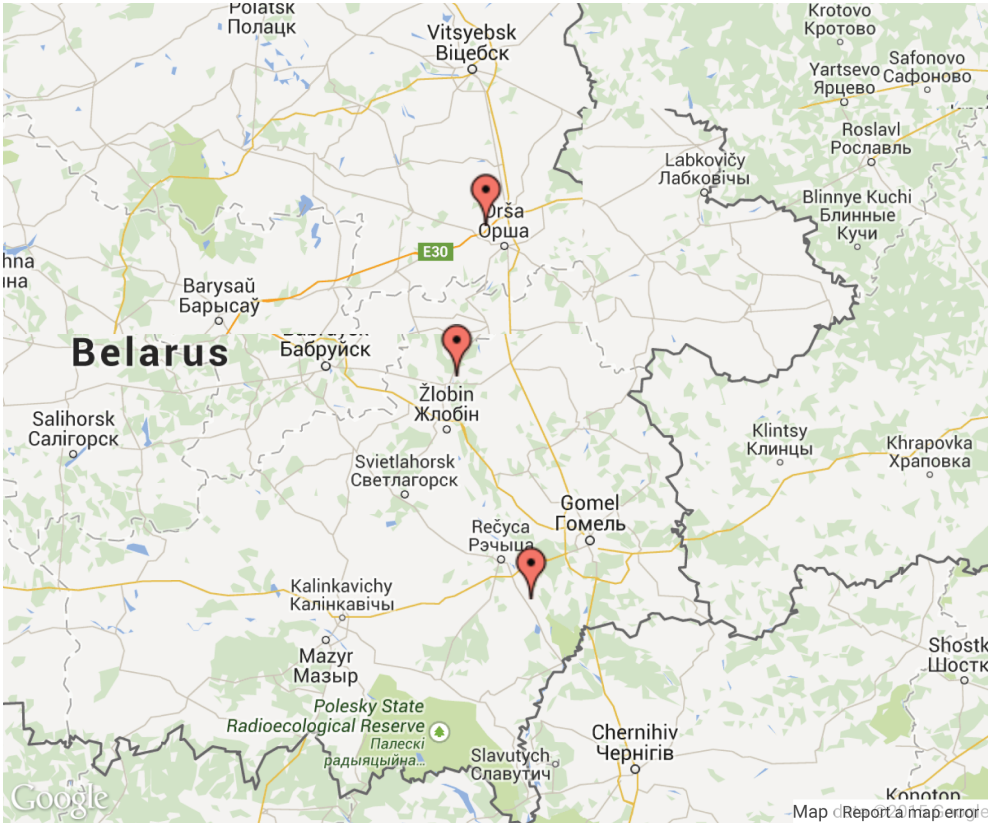


Рисунок 146

На рис. 146 мы привели лишь копию экрана браузера, но читатель может увидеть интерактивный вариант карты, посетив страницу http://bit.ly/1z0L2yr.

Для извлечения html-кода карты из объекта `Map` необходимо выполнить следующую команду:

```
print(Map, tag = "chart")
```

Код будет выведен непосредственно в консоль R, откуда его можно скопировать и вставить в свою веб-страницу.

В 2006 г. выступление профессора Ханса Рослинга (http://www.gapminder.org) на конференции «Технологии, развлечения, дизайн» («*Technology Entertainment Design*», TED; http://bit.ly/1kUuqgl) произвело своего рода революцию в подходах к интерактивной визуализации многомерных данных. Речь идет о «*motion charts*» («*анимированные диаграммы*»), которые профессор Рослинг использовал в своей презентации для анализа социально-экономической ситуации в мире за последние 50 лет.

В состав библиотеки googleVis входят функции для создания не только карт Google, но и разнообразных анимированных диаграмм. Рассмотрим пример создания такого графика на основе таблицы данных Fruits, входящей в состав библиотеки googleVis. Эта таблица содержит динамику продаж трех видов фруктов (Apples, Bananas, Oranges) в двух регионах (East и West). Предположим, мы хотим визуализировать зависимость затрат Expenses от объема продаж Sales для всех видов фруктов Fruit в разных регионах Location и для разных периодов времени Year.

Для создания графика с элеменами управления, позволяющими выбирать разные варианты отображения данных, используем функцию **gvisMotionChart**(), которая имеет четыре основных аргумента:

```
gvisMotionChart(data, idvar = "id", timevar = "date", chartid),
```

где data – имя таблицы данных; idvar – имя номинальной переменной, для которой строится график; timevar – имя переменной, задающей ось времени, а chartid – необязательный аргумент, позволяющий присвоить графику уникальное имя.

В нашем случае команда выглядит следующим образом:

```
data(Fruits)
M <- gvisMotionChart(Fruits, idvar = "Fruit", timevar = "Year")
```

Объект M является списком из трех списков (проверьте командой str(M)). Первые два элемента этого списка содержат информацию о типе графика (MotionChart) и его уникальное имя (chartid). Третий элемент наиболее интересен, поскольку он содержит html-код графика. Этот третий элемент, в свою очередь, включает четыре других элемента: header («шапка»), chart – непосредственно код графика, caption – подпись графика и footer («футер»). html-код графика можно легко извлечь командой print(M, tag = "chart"), а затем скопировать и вставить на страницу веб-сайта.

В тексте этой книги мы, естественно, не можем показать процесс анимации и ограничились только копией окна браузера (рис. 147). Увидеть интерактивный вариант графика можно на странице http://bit.ly/1z0NuVv.

На полученном графике имеется несколько возможностей для визуализации зависимостей между различными переменными. Например, мы можем: *а)* выбрать для отображения необходимый период времени Year путем перемещения «слайдера» в нижней части графика, *б)* раскрасить желаемым цветом «пузыри» в соответствии с регионом Location, *в)* прикрепить названия фруктов к каждому «пузырю» (меню Select), *г)* изменить стиль графика с «пузырей» на столбиковую или линейную диаграмму и т. д.

При помощи функций пакета googleVis можно создавать и множество других типов графиков, а также объединять их на одной странице. Для ознакомления с имеющимися возможностями выполните команду **demo**(googleVis).

9.4. Создание картограмм при помощи R

В этом разделе мы покажем, как при помощи R можно создать *картограммы* («*choropleth maps*»). На картограммах отдельные территориальные единицы залиты цветом, интенсивность которого соответствует величине отображаемой

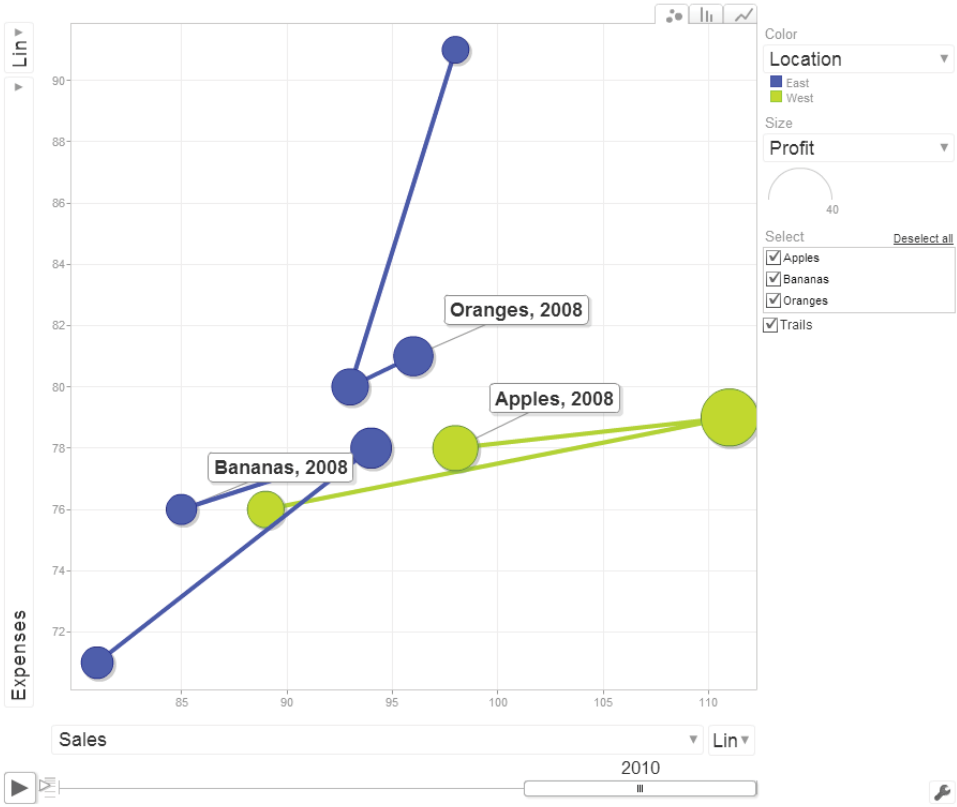


Рисунок 147

количественной переменной (плотность населения, уровень дохода, процент проголосовавших за определенного кандидата на выборах и т. п.). Процедуру построения картограммы мы без комментариев привели в разделе 1.2, когда описывали приемы работы с командной консолью R. Теперь настало время объяснить все подробнее.

Шейп-файлы

Основой для создания картограмм являются так называемые «шейп-файлы» («*shapefile*») для разных административно-территориальных единиц. Шейп-файл представляет собой распространенный формат хранения информации о геометрическом положении и определенных атрибутах географических объектов. Несмотря на свое название, в действительности шейп-файл – это не один, а набор из нескольких файлов с одинаковым именем, но разными расширениями. Обязательными при этом являются файлы с расширениями `.shp`, `.dbf` и `.shx`. Мы не будем описывать здесь, что содержится в каждом из этих файлов, так как об этом

можно найти много информации в Сети – см., например, статью в Википедии (http://bit.ly/1OSxqXP) и официальное техническое описание формата (http://arcg.is/1o0k7Lr). Для наших целей достаточным будет представление о шейп-файле как о файле, по данным из которого можно воспроизвести контурную карту страны и/или отдельных ее территориальных единиц.

Естественно, встает вопрос о том, где можно раздобыть шейп-файлы для той или иной интересующей нас территории. Одним из бесплатных (для некоммерческого использования) источников является «*Глобальная база данных административных областей*» (*Global Administrative Areas,* http://gadm.org). В этой базе данных доступны шейп-файлы трех разных уровней пространственного разрешения: страна, области и районы.

Необходимые нам шейп-файлы можно получить на сайте GADM двумя разными путями. Во-первых, мы можем сделать это «вручную», зайдя на сайт GADM и выполнив следующие шаги.

1. В главном меню сайта выбираем опцию *Download*.
2. На открывшейся странице из выпадающих меню выбираем желаемую страну (*Country*) и формат скачиваемых файлов (*File Format*). В меню *File Format* доступно несколько опций, включая нужный нам шейп-файл (*Shapefile*). Стоит отметить, что специалисты, работающие над созданием и поддержкой GADM, отлично осведомлены о распространенности и возможностях R, и, как следствие, в качестве одной из опций в меню *File Format* присутствует *R (SpatialPolygonsDataFrame)* – файл «рабочего пространства» R, который уже содержит шейп-файл в необходимом формате и может быть непосредственно загружен в программу. Описание того, как это сделать, приведено ниже. Пока же выберем опцию *Shapefile*:
3. Все последующие картограммы мы будем строить на примере Республики

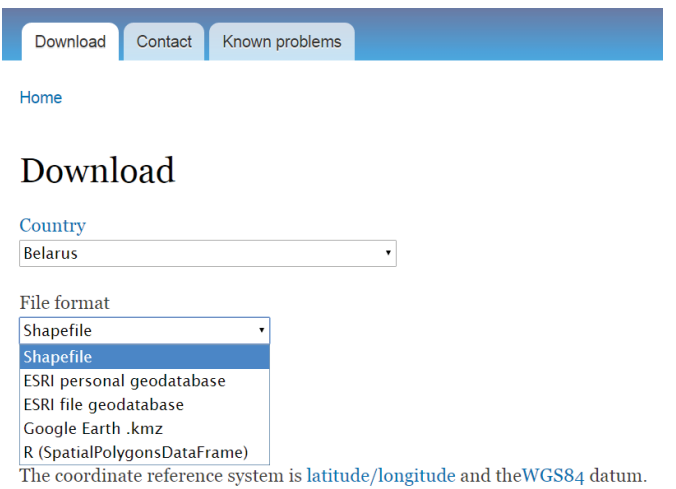


Рисунок 148

Беларусь. На открывшейся странице увидим ссылку *download* для скачивания zip-архива с нужными нам шейп-файлами для всех трех уровней пространственного разрешения. Сохраним этот архив (`BLR_adm.zip`) в удобном месте (лучше всего в текущей рабочей папке R).

4. Разархивируем `BLR_adm.zip` в текущую рабочую папку R. В результате в этой папке появятся следующие файлы (рис. 149):

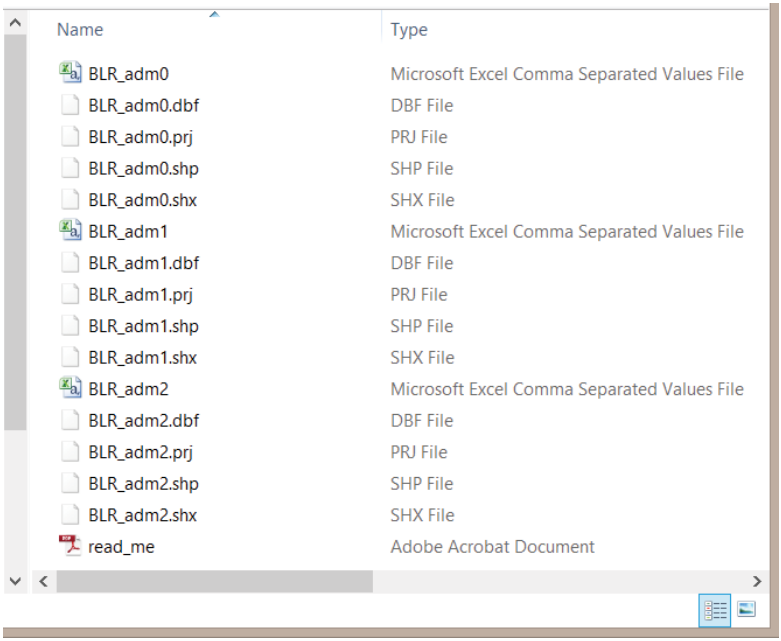


Рисунок 149

Все перечисленные файлы важны и должны находиться *в одной директории*. Обратите внимание на цифры в именах файлов – они помогают понять, какому уровню пространственного разрешения соответствует каждый файл. Так, файлы, в названии которых имеется 0, описывают контуры страны, 1 – контуры страны и областей, а 2 – контуры страны, областей и районов.

Для загрузки шейп-файлов в операционную среду R можно воспользоваться функцией **`readShapePoly`**`()` из пакета `maptools`. Так, шейп-файл для первого уровня пространственного разрешения (страна, разделенная на области) можно загрузить следующим образом:

```
library(maptools)
Regions <- readShapePoly("BLR_adm1.shp")
```

Второй способ получения шейп-файлов с сайта GADM состоит в их загрузке по URL непосредственно из среды R. Обычно этот способ предпочтительнее, и ниже

мы будем предполагать, что им вы и воспользовались. Так, для загрузки шейп-файла с контурами страны и областей нужно выполнить следующую команду:

```
load(url("http://biogeo.ucdavis.edu/data/gadm2/R/BLR_adm1.RData"))
```

В результате в рабочей среде R появится объект с именем gadm. Во избежание путаницы в дальнейшем стоит сохранить этот объект, присвоив ему какое-либо более описательное имя, например Regions, а исходный объект затем просто удалить:

```
Regions <- gadm
rm(gadm)
```

С точки зрения своего внутреннего устройства и представления в R, созданный нами объект Regions принадлежат к объектам класса S4 (подробнее об этом классе, а также о многих других особенностях программиррвания на R можно узнать из замечательной книги Хэдли Уикхэма «*Advanced R*», http://adv-r.had.co.nz). Вкратце скажем, что такие объекты имеют сложную структуру, напоминающую структуру обычных списков R. Отдельные компоненты S4-объектов называются *слотами* («*slots*») и могут сами являться сложными объектами любого другого класса. В отличие от обычных списков, обращение к слотам S4-объектов происходит при помощи оператора @, а не $. Кроме того, к слотам можно обращаться только по их именам, тогда как отдельные элементы списков мы можем индексировать также по их порядковым номерам. Имена слотов, входящих в состав Regions, легко выяснить при помощи базовой R-функции **slotNames**():

```
slotNames(Regions)
[1] "data"        "polygons"    "plotOrder"
    "bbox"        "proj4string"
```

Наибольший интерес для нас представляют слоты data и polygons. Первый включает обычную таблицу данных, содержимое которой (идентификационные номера отдельных территориальных единиц, названия этих единиц и прочее) можно просмотреть при помощи команды Regions@data. В свою очередь, слот polygons содержит список с координатами точек, задающих контуры соответствующих территориальных единиц в формате системы координат WGS84 (http://bit.ly/1OpWsCP), а также некоторые другие атрибуты:

```
str(Regions@polygons)
List of 6
 $ :Formal class 'Polygons' [package "sp"] with 5 slots
  .. ..@ Polygons :List of 1
  .. .. ..$ :Formal class 'Polygon' [package "sp"] with 5 slots
  .. .. .. .. ..@ labpt  : num [1:2] 25.5 52.4
  .. .. .. .. ..@ area   : num 4.3
  .. .. .. .. ..@ hole   : logi FALSE
  .. .. .. .. ..@ ringDir: int 1
  .. .. .. .. ..@ coords : num [1:2100, 1:2] 25.9 25.9 26 26 26 ...
  .. ..@ plotOrder: int 1
  .. ..@ labpt    : num [1:2] 25.5 52.4
```

```
  .. ..@ ID       : chr "0"
  .. ..@ area     : num 4.3
# остальная часть выведенной информации опущена для экономии места
```

Разобравшись со структурой шейп-файлов и их представлением в R, приступим собственно к построению картограмм. Как это часто случается, в R имеется несколько способов достижения одной и той же цели. Не являются исключением и картограммы.

Функция spplot() из пакета sp

Пакет sp содержит функции, реализующие целый ряд методов и классов для пространственных данных (импорт, преобразование и экспорт данных, графические методы и т. п.). В частности, в нем имеется функция **spplot()**, которая позволяет отображать на графике данные, заключенные в шейп-файле. В основе этой функции лежат высокоуровневые команды lattice – одного из популярных графических пакетов R.

Для начала построим картограмму Беларуси, на которой каждая область будет залита своим цветом (рис. 150; цвет в этом примере служит просто для идентификации области – никакой другой информационной нагрузки он не несет). Посмотрим еще раз на названия областей, которые следуют по алфавиту (приведены оригинальные белорусские названия в написании латинскими буквами):

```
(Regions@data$NAME_1 <- factor(Regions@data$NAME_1))
[1] Brest     Homyel'   Hrodna    Mahilyow  Minsk     Vitsyebsk
Levels: Brest Homyel' Hrodna Mahilyow Minsk Vitsyebsk
```

Переменная NAME_1, как видим, является фактором с 6 уровнями. Каждому из этих уровней при построении карты необходимо будет присвоить уникальный цвет. Хорошо отличимые друг от друга цвета можно подобрать, воспользовавшись одной из стандартных палитр R, например **rainbow()**:

```
library(sp)
spplot(Regions, "NAME_1", # отображаемая переменная
    scales = list(draw = T), # отображение координатных осей
    col.regions = rainbow(n = 6) ) # опция, задающая заливку цветом
```

Можно попробовать и другие стандартные палитры R, например **terrain.colors()** (см раздел 1.2) или **heat.colors()**. Стоит обратить внимание и на специально подобранные для создания карт палитры из пакета RColorBrewer (см. также http://colorbrewer2.org). Помимо использования одной из палитр RColorBrewer, в приведенном ниже примере добавлена опция par.settings, при помощи которой отключаются координатные оси вокруг карты (рис. 151):

```
library(RColorBrewer)
spplot(Regions, "NAME_1",
       col.regions = brewer.pal(6, "Set3"),
       par.settings = list(axis.line = list(col = NA)))
```

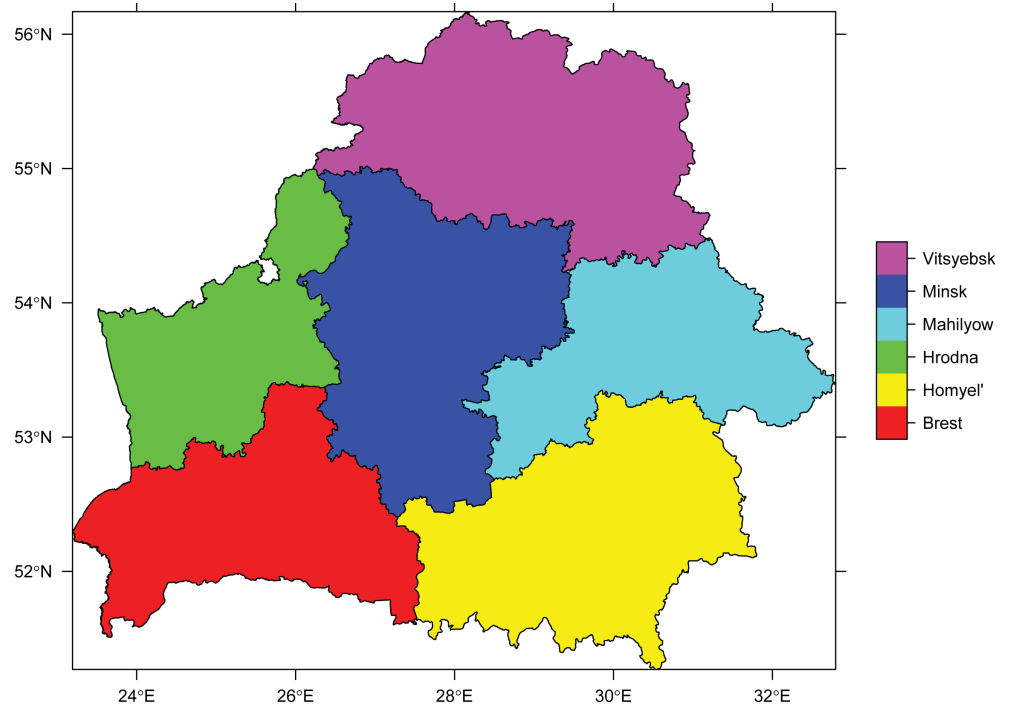


Рисунок 150

Теперь придадим цвету смысл – построим картограмму, на которой цветовая шкала будет соответствовать числу мужчин, проживающих в каждой области. Данные взяты с сайта http://belstat.gov.by и представляют собой результаты переписи населения Беларуси за 2009 г. Значения численности мужчин достаточно добавить в таблицу данных, уже хранящуюся в слоте `data` объекта `Regions` (см. выше):

```
# Порядок чисел важен! Он должен совпадать с порядком,
# в котором в таблице data перечислены соответствующие области:
Regions@data$Population = c(186716, 152169, 131817,
                            105417, 253427, 135348)
```

Перед построением картограммы нам необходимо определиться с цветовой шкалой, то есть какими именно цветами шкала должна быть представлена и на сколько градаций разбита. Сделать это можно разными способами в зависимости от стоящей задачи, а также от разброса значений отображаемой на картограмме переменной. Воспользуемся одним из таких способов – определением цветовой палитры при помощи базовой функции **`colorRampPalette`**`()`. В качестве аргументов этой функции следует указать «опорные» цвета, между которыми требуется произвести интерполяцию (то есть плавный переход, например, от красного к желтому). Функция **`colorRampPalette`**`()` возвращает другую функцию, которая и вы-

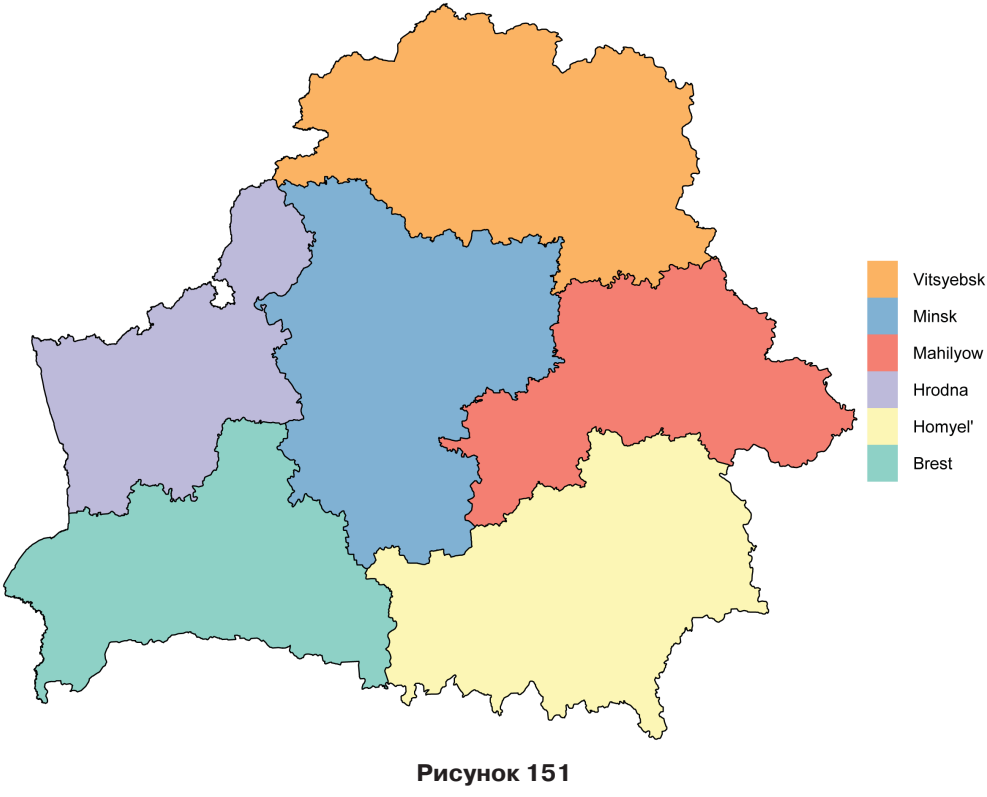


Рисунок 151

полняет такую интерполяцию автоматически. Для нашего примера мы создадим цветовую шкалу, в которой будет происходить плавный переход от цвета `seagreen` к `whitesmoke` (для ознакомления со списком стандартных цветов R выполните команду **`colors`**`()`):

```
mypalette <- colorRampPalette(c("seagreen", "whitesmoke"))
# Объект mypalette - это функция:
mypalette
function (n)
{
    x <- ramp(seq.int(0, 1, length.out = n))
    rgb(x[, 1L], x[, 2L], x[, 3L], maxColorValue = 255)
}
```

Теперь функцию **`mypalette`**`()` мы можем подать на функцию **`spplot`**`()`. При этом в качестве (единственного) аргумента **`mypalette`**`()` следует указать общее *количество* опорных и промежуточных цветов (`n`), которое должно присутствовать в шкале. Чем больше разброс значений отображаемой переменной, тем выше должно быть это число – так будет достигаться плавный переход между цветами. В нашем слу-

чае разброс значений численности мужчин в отдельных областях невелик, и поэтому выберем n = 20. В приведенной ниже команде для демонстрации еще одной опции функции **spplot**() добавлен аргумент col со значением "transparent" (прозрачный) – с его помощью отключается отображение линий, ограничивающих каждую область (рис. 152):

```
spplot(Regions, "Population",
       col.regions = mypalette(20), # определение цветовой шкалы
       col = "transparent", # отключение контурных линий на карте
       par.settings = list(axis.line = list(col = NA)))
```

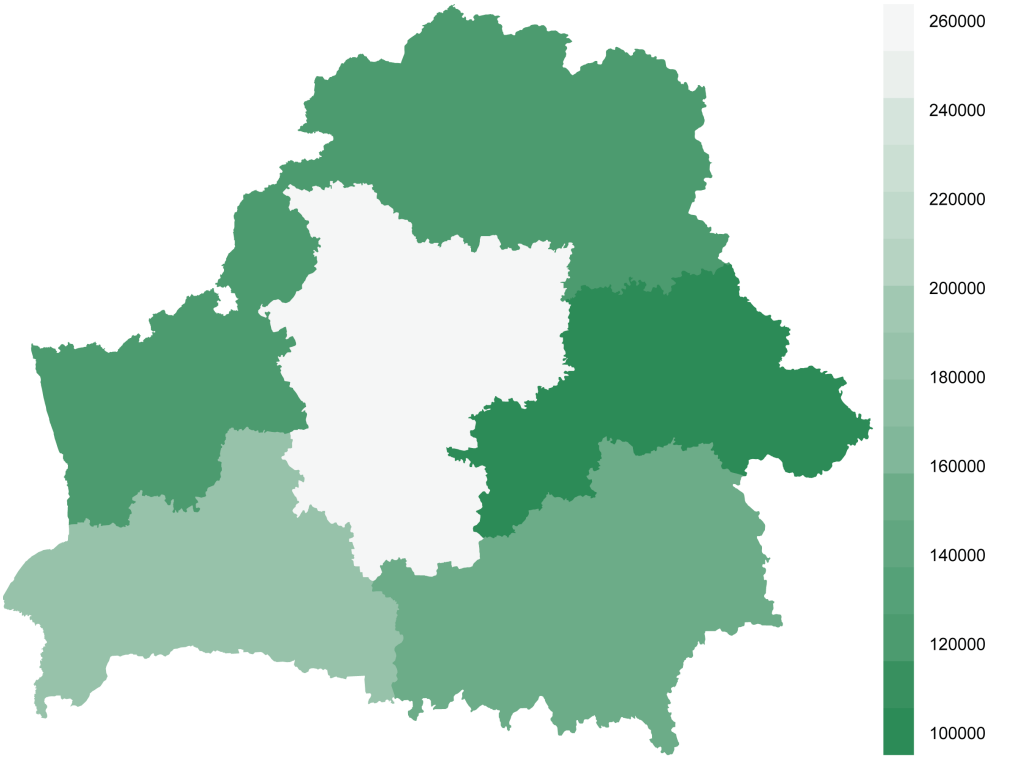


Рисунок 152

Как видим, наибольшая численность мужчин в Беларуси имеет место в Минской области и в двух южных областях – Брестской и Гомельской.

Рассмотренный принцип добавления пользовательских данных в слот data шейп-файла с последующим их отображением на картограмме можно применить и к более дробным территориальным единицам – например, районам. Загрузим соответствующий шейп-файл с сайта GADM и сохраним его в виде объекта с именем Counties:

```
load(url("http://biogeo.ucdavis.edu/data/gadm2/R/BLR_adm2.RData"))
Counties <- gadm; rm(gadm)
```

Вместо использования реальных данных воспользуемся генератором случайных чисел и создадим вектор Fake из нормально распределенных значений. Этот вектор будет добавлен в слот data объекта Counties. Длина вектора Fake будет соответствовать числу районов в Беларуси:

```
set.seed(1234) # для воспроизводимости результата
Counties@data$Fake <- rnorm(dim(Counties@data)[1], 100, 35)
```

Для создаваемой картограммы применим еще один способ определения цветовой шкалы. Будем использовать цвета из пакета RColorBrewer, в частности из «красно-голубой» палитры RdBu. При работе с количественными переменными возникает проблема с тем, чтобы разбить их значения на отдельные промежутки, которым можно сопоставить соответствующие цвета. В R такое разбиение не составляет труда, причем есть несколько способов это сделать. Мы воспользуется возможностями пакета classInt. Функция **classIntervals**() разбивает имеющуюся совокупность чисел на классы с определенным шагом, задаваемым при помощи аргумента style. Мы постараемся разбить наш вектор Fake на 6 классов одинакового размера:

```
require(classInt)
brks.eq = classIntervals(Counties$Fake, n = 6, style = "equal")
```

Просмотрим результат разбиения этого вектора на 6 классов и построим карту (рис. 153):

```
brks.eq
style: equal
  one of 167,549,733 possible partitions of this variable
  into 6 classes
[17.90058,46.45293) [46.45293,75.00528) [75.00528,103.5576)
                  4                   31                   43
  [103.5576,132.11)   [132.11,160.6623) [160.6623,189.2147]
                 24                   11                    5

# Задаем палитру из 6 цветов:
plotcol <- brewer.pal(6,"RdBu")
# Рисуем карту:
spplot(Counties, "Fake",
       col.regions = plotcol,
       at = brks.eq$brks, # задает границы классов
       par.settings = list(axis.line = list(col = NA)))
```

Создание картограмм при помощи пакета ggplot2

Пакет ggplot2 – пожалуй, самая популярная коллекция графических функций для R. Следует сразу оговориться, что синтаксис команд ggplot2 является своего рода мини-языком программирования. Изложение основ ggplot2 не является

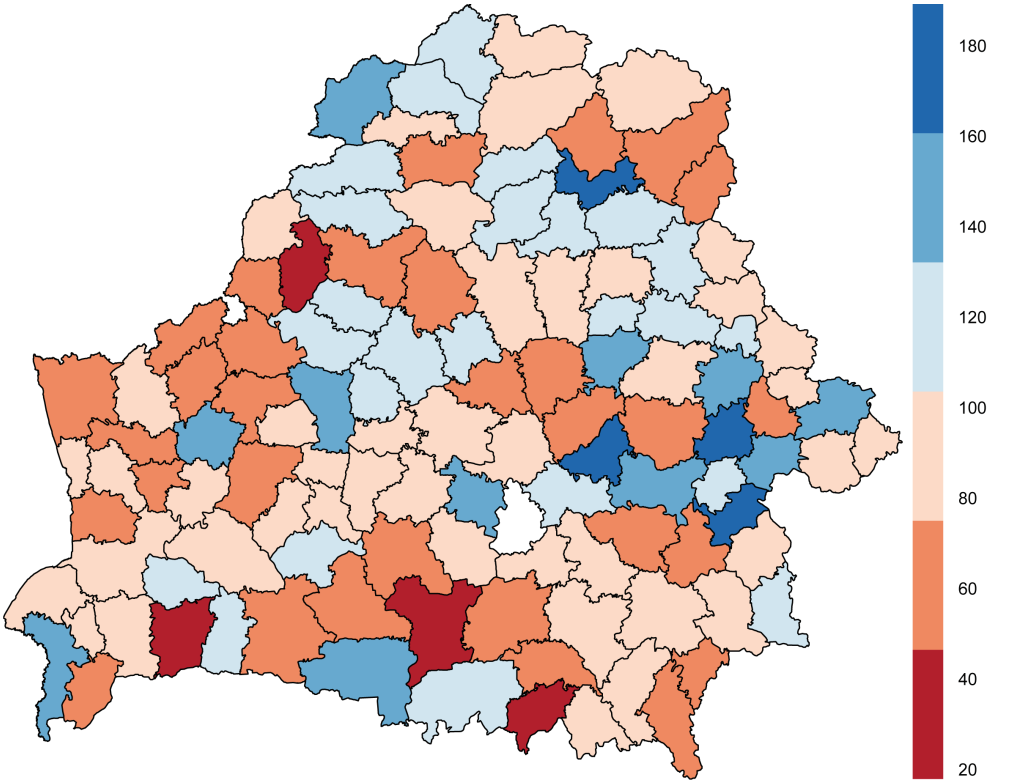


Рисунок 153

целью данного раздела – для этого существуют соответствующие руководства, из которых особое внимание стоит обратить на официальную документацию пакета (http://bit.ly/1GvmXT3), а также на книги Wickham (2009, http://bit.ly/1IXz1dc) и Chang (2013, http://oreil.ly/1b1Sg9Z). В приведенных ниже примерах даются лишь краткие пояснения к коду, необходимые для его понимания.

Напомним, что выше мы загрузили шейп-файл с информацией по административно-территориальным единицам Беларуси на уровне районов и создали объект `Counties`, который относится к объектам класса `S4`. Однако единственный формат данных, который принимают функции пакета `ggplot2`, – это «таблица данных». Соответственно, сначала нам необходимо преобразовать объект `Counties` в таблицу данных. Сделать это позволяет функция **`fortify`**`()` из пакета `ggplot2`:

```
library(ggplot2); library(rgeos); library(maptools)
counties <- fortify(Counties, region = "NAME_2")
# аргумент region = "NAME_2" указывает на переменную (NAME_2)
# с идентификаторами административных единиц - районов
str(counties)
'data.frame': 42659 obs. of  7 variables:
```

```
 $ long : num  29.1 29.1 29.1 29.1 29.1 ...
 $ lat  : num  52.9 52.9 52.9 52.9 52.8 ...
 $ order: int  1 2 3 4 5 6 7 8 9 10 ...
 $ hole : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ piece: Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
 $ group: Factor w/ 118 levels "Aktsyabar.1",..: 1 1 1 1 1 1 1 1 1 1
 $ id   : chr  "Aktsyabar" "Aktsyabar" "Aktsyabar" "Aktsyabar" ...
```

Теперь мы можем построить прототип нашей будущей картограммы при помощи следующей составной команды:

```
ggplot() + geom_map(data = counties,
                    aes(map_id = id),
                    map = counties, color = "gray70") +
  expand_limits(x = counties$long, y = counties$lat) +
  coord_map("polyconic")
```

Создание статистических графиков средствами ggplot2 напоминает послойное создание изображений в программе *Adobe Photoshop*: вначале создается пустой слой графика, поверх которого далее добавляются другие слои, содержащие определенные графические элементы (в частности, изображение карты). Рассмотрим отдельные части приведенной выше команды подробнее:

- **ggplot**(): инициирует создание пустого слоя;
- **geom_map**(): создает слой с картой, внешний вид которой задается при помощи соответствующих аргументов. Так, при помощи аргумента data мы указываем таблицу данных, в которой содержатся подлежащие отображению на карте данные. Поскольку на первом этапе никаких таких данных у нас нет, в качестве «заглушки» мы просто указали таблицу counties. Функция **aes**() позволяет настроить «эстетические параметры» (от англ. «*aesthetics*») создаваемого слоя и, в частности, указать, какая переменная в таблице с данными содержит идентификаторы административно-территориальных единиц (районов). Аргумент map служит для указания таблицы с координатами соответствующих полигонов на карте (в нашем случае это counties – таблица, которую мы создали при помощи функции **fortify**() из объекта Counties;
- **expand_limits**(): в отличие от других функций ggplot2, которые предназначены для создания новых слоев с графическими элементами, функция **geom_map**() не способна автоматически определять оптимальные пределы значений *x*- и *y*-координат. Сделать это ей помогает функция **expand_limits**();
- **coord_map**(): задает тип картографической проекции (http://bit.ly/1Dx2tmK). В приведенном примере использован тип polyconic – *поликоническая проекция*. Доступны также многие другие типы, реализованные в пакете mapproj (подробнее см. справочный файл, доступный по команде ?mapproject).

Для иллюстрации принципа создания картограмм добавим к структуре Counties@data столбец fake_data$Value со случайно сгенерированными данными для каждого района Беларуси:

```
fake_data <- as.data.frame(Counties@data)
fake_data$Value <- rnorm(nrow(fake_data))
```

Теперь отобразим данные из столбца `Value` таблицы `fake_data` на картограмме:

```
ggplot() + geom_map(data = fake_data,
                    aes(map_id = NAME_2, fill = Value),
                    map = counties) +
  expand_limits(x = counties$long, y = counties$lat) +
  coord_map("polyconic")
```

Результатом приведенной команды будет картограмма с автоматически выбранной цветовой шкалой (рис. 154). Может возникнуть необходимость изменить эту цветовую шкалу. Так, в рассматриваемом примере отображаемые на карте данные имеют среднее значение 0. Имеет смысл определенным образом выделить это значение на цветовой шкале, например при помощи белого цвета. Для значений же, отклоняющихся от 0 в отрицательную и в положительную стороны, можно выбрать, например, оттенки синего и красного соответственно.

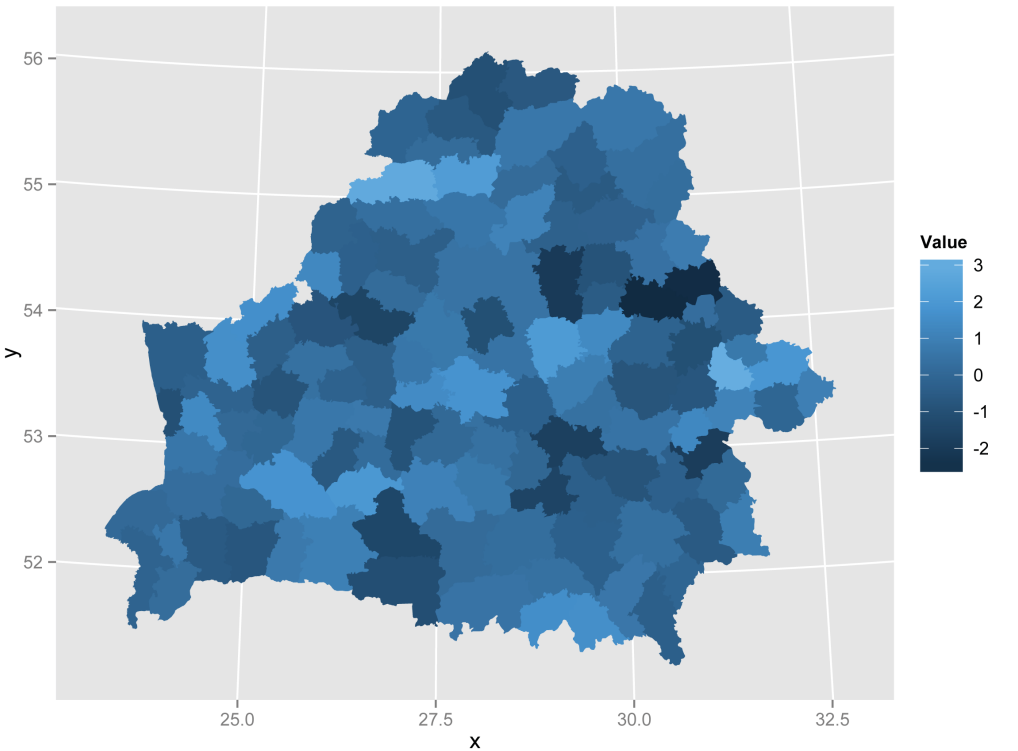


Рисунок 154

Для создания подобных расходящихся («*diverging*») цветовых шкал в пакете `ggplot2` служит функция **`scale_fill_gradient2()`**, которая работает примерно следующим образом (рис. 155):

```
library(scales) # для функции muted (см. ниже)
ggplot() + geom_map(data = fake_data,
                    aes(map_id = NAME_2, fill = Value),
                    colour = "gray",
                    map = counties) +
  expand_limits(x = counties$long, y = counties$lat) +
  scale_fill_gradient2(low = muted("blue"),
                       midpoint = 0,
                       mid = "white",
                       high = muted("red"),
                       limits = c(min(fake_data$Value),
                                  max(fake_data$Value))) +
  coord_map("polyconic")
```

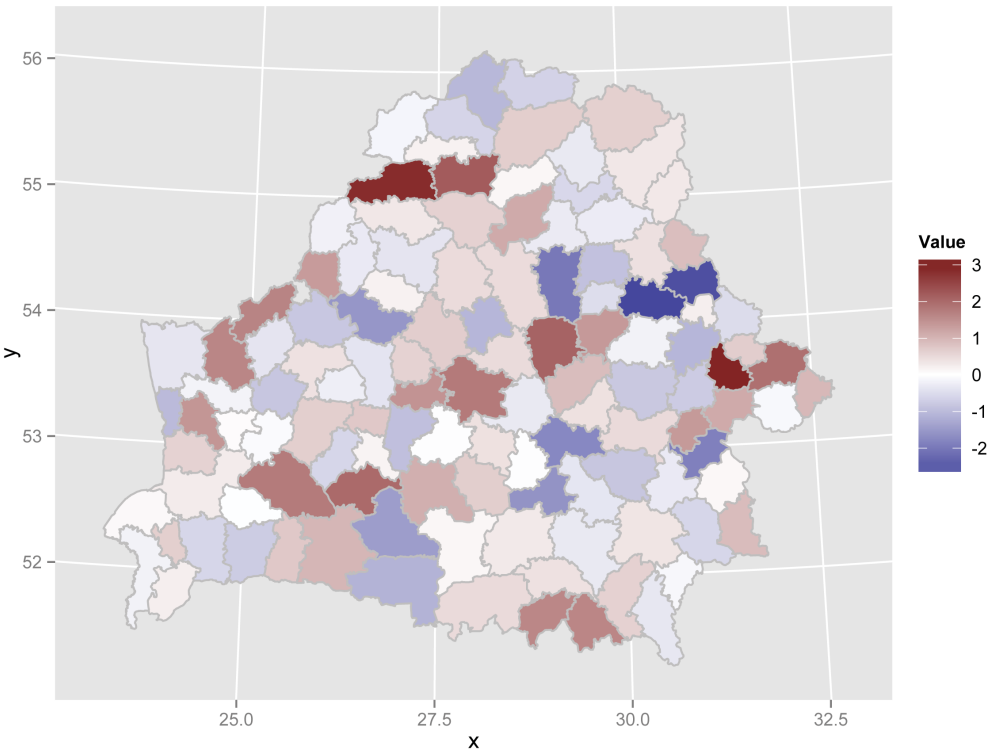


Рисунок 155

Обратите внимание на аргумент `colour = "gray"` функции **`geom_map`**`()` – он добавлен для обводки границ районов серым цветом.

При необходимости мы можем выполнить тонкую настройку и других деталей внешнего вида графика. В частности, при помощи функции **`theme`**`()` («тема», или «шаблон») можно отключить отображение серого фона, координатной сетки, а также осей и их подписей:

```
ggplot() + geom_map(data = fake_data,
                    aes(map_id = NAME_2, fill = Value),
                    colour = "gray",
                    map = counties) +
  expand_limits(x = counties$long, y = counties$lat) +
  scale_fill_gradient2(low = muted("blue"),
                       midpoint = 0,
                       mid = "white",
                       high = muted("red"),
                       limits = c(min(fake_data$Value),
                                  max(fake_data$Value))) +
  coord_map("polyconic") +
  theme(
    axis.line = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    plot.background = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank()  )
```

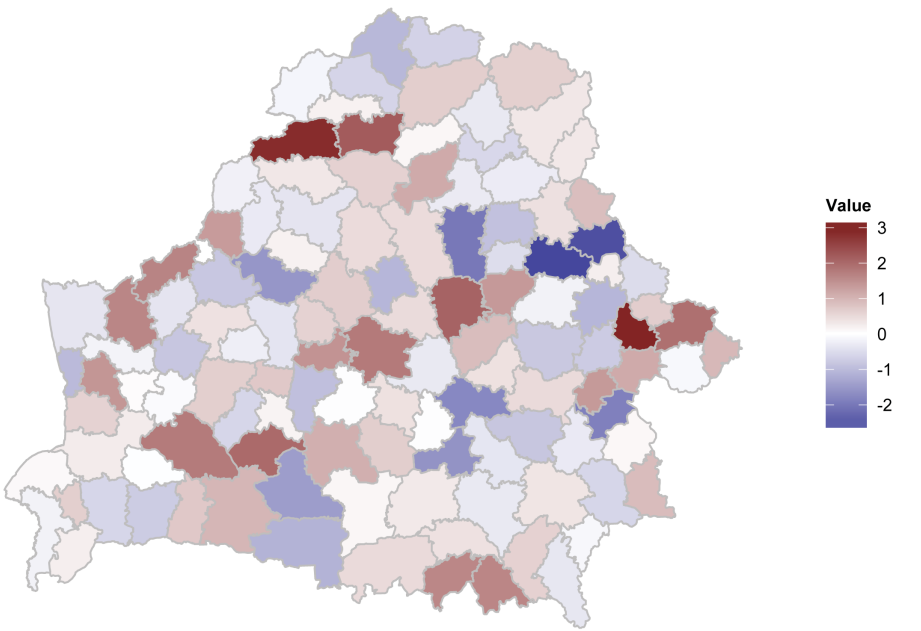


Рисунок 156

Библиография и интернет-ресурсы

Основные литературные ссылки по тексту книги

Литература по R

Зарядов И. С. Введение в статистический пакет R: типы переменных, структуры данных, чтение и запись информации, графика. М.: Изд-во РУДНБ, 2010а. 207 с. PDF: http://bit.ly/1GrlyK7.

Зарядов И. С. Статистический пакет R: теория вероятностей и математическая статистика. М.: Изд-во РУДНБ, 2010б. 141 с. PDF: http://bit.ly/1Js5UlS.

Кабаков Р. И. R в действии: Анализ и визуализация данных в программе R[1]. 2-е изд. / пер. с англ. П. А. Волковой. М.: ДМК Пресс, 2013. 580 с.

Шипунов А. Б., Балдин Е. М., Волкова П. А., Коробейников А. И., Назарова С. А., Петров С. В., Суфиянов В. Г. Наглядная статистика. Используем R! М.: ДМК Пресс, 2014. 298 с. PDF: http://bit.ly/1btoICD.

Шитиков В. К., Розенберг Г. С. Рандомизация и бутстреп: статистический анализ в биологии и экологии с использованием R. Тольятти: Кассандра, 2014. 314 с. PDF, данные и скрипты доступны на сайте авторов: http://bit.ly/1Js6Fv8.

Chernick M. R., LaBudde R. A. An Introduction to Bootstrap Methods with Applications to R. Wiley, 2011. 236 p.

Cleveland W. Visualizing Data. At&T Bell Laboratories, 1993. 360 p.

Dalgaard P. Introductory Statistics with R. 2nd ed. Springer, 2008. 364 p.

Everitt B. S., Hothorn T. A Handbook of Statistical Analyses Using R. 2nd ed. Chapman, Hall/CRC, 2010. 376 p.

Faraway J. J. Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models. Chapman, Hall/CRC, 2006. 345 p.

Faraway J. J. Linear Models with R. Chapman, Hall/CRC, 2004. 240 p.

Fox J. Applied Regression Analysis and Generalized Linear Models, 2nd ed. Sage Publications, 2008.

Fox J., Weisberg S. An R Companion to Applied Regression, 2nd ed. Sage Publications, 2010.

Gelman A., Hill J. Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press, 2006. 651 p.

Karp N. R Commander: An Introduction. 2014. 52 p. PDF: http://bit.ly/19jDYyc.

Lam L. An Introduction to R. Vrije Universiteit Amsterdam, 2010. 212 p. PDF: http://bit.ly/1E8tXSf.

Larson-Hall J. A Guide to Doing Statistics in Second Language Research Using R. New York, NY: Routledge, 2009. 320 p. PDF: http://bit.ly/1Gvc0OA.

Logan M. Biostatistical Design and Analysis Using R: A Practical Guide. Wiley-Blackwell, 2010. 576 p.

Maindonald J. H., Braun W. J. Data Analysis and Graphics Using R: An Example-Based Approach. 3rd ed. Cambridge University Press, 2010. 565 p.

Meys J., Vries A. R For Dummies. For Dummies, 2012. 406 p.

Pinheiro J. C., Bates D. M. Mixed-Effects Models in S and S-PLUS. Springer, 2000. 530 p.

Ricci V. Fitting Distributions with R. 2005. 24 p. PDF: http://bit.ly/1A1Bgag.

---

[1] В настоящее время готовится перевод обновленного издания.

Sarkar D. Lattice: Multivariate Data Visualization with R. Springer, 2008. 268 p.
Venables W. N., Ripley B. D. Modern Applied Statistics with S. 4th ed. Springer, 2002. 505 p.
Venables W. N., Smith D. M. An Introduction to R. R Development Core Team, 2014. 105 p. PDF: http://bit.ly/1PFNzSP. (Русский перевод: *Венэбльз У. Н., Смит Д. М.* Введение в R: Заметки по R: среда программирования для анализа данных и графики / пер. с англ. Вер. 3.1.0. М., 2014. 109 с. PDF: http://bit.ly/1E8uNyl.)
Wickham H. Advanced R. Chapman & Hall/CRC. The R Series, 2014. 476 p. (Официальная бесплатная версия книги доступна по адресу http://adv-r.had.co.nz.)
Wood S. N. Generalized Additive Models: An Introduction with R. Chapman, Hall/CRC, 2006. 410 p.
Zieffer A. S., Harring J. R., Long J. D. Comparing Groups: Randomization and Bootstrap Methods Using R. Wiley, 2011. 332 p.
Zuur A. F., Ieno E. N., Walker N. et al. Mixed Effects, Models and Extensions in Ecology with R. Berlin: Springer Sci., 2009. 574 p.

Общеметодическая литература по статистическому анализу

Бослаф С. Статистика для всех. М.: ДМК Пресс, 2014. 586 с.
Гланц С. Медико-биологическая статистика. М.: Практика, 1999. 459 с.
Дрейпер Н., Смит Г. Прикладной регрессионный анализ. М.: Финансы и статистика. Кн. 1. 1987. 366 с.; Кн. 2. 1987. 352 с.
Кобзарь А. И. Прикладная математическая статистика. М.: Физматлит, 2006. 816 с.
Справочник по прикладной статистике: в 2 т. / пер. с англ., под ред. Э. Ллойда, У. Ледермана, Ю. Н. Тюрина. М.: Финансы и статистика. Т. 1. 1989. 510 с.; Т. 2. 1990. 526 с.
Такахаси С. Занимательная статистика. Манга. М.: ДМК Пресс, 2010. 224 с.
Такахаси С. Занимательная статистика. Регрессионный анализ. Манга. М.: ДМК Пресс, 2014. 222 с.
Такахаси С., Ироха И. Занимательная статистика. Факторный анализ. Манга. М.: ДМК Пресс, 2014. 248 с.
Chambers J., Cleveland W., Kleiner B., Tukey P. Graphical Methods for Data Analysis (Statistics). Chapman and Hall/CRC, 1983. 336 p.
Hastie T. J., Tibshirani R. J. Generalized Additive Models. Chapman & Hall/CRC, 1990. 352 p.
Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. Springer, 2009. 745 p.
Harrell F. E. Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis. Springer, 2002. 572 p.
Legendre P., Legendre L. Numerical Ecology. 2nd ed. Amsterdam: Elsevier Science, 1998. 852 p.
McCullagh P., Nelder J. A. Generalized Linear Models. 2nd ed. Chapman and Hall/CRC, 1989. 532 p.
Singer J. D., Willett J. B. Applied Longitudinal Data Analysis: Modeling Change and Event Occurrence. New York: Oxford University Press, 2003. 672 p.
Sokal R. R., Rohlf F. J. Biometry: The Principles and Practice of Statistics in Biological Research. 3nd ed. N.Y.: Freeman, 1995. 887 p.
Zar J. H. Biostatistical Analysis. 5th ed. London: Pearson Prentice Hall, 2010. 945 p.

Библиографический указатель литературы по R

Буховец А. Г., Москалев П. В., Богатова В. П., Бирючинская Т. Я. Статистический анализ данных в системе R: учеб. пособие. Воронеж, 2012. ВГАУ, 124 с.
Волкова П. А., Шипунов А. Б. Статистическая обработка данных в учебно-исследовательских работах / под ред. Ю. А. Багринцева, Б. М. Каплан, Е. Н. Сиднева. М.: Экспресс, 2008. 60 с.

Гнатюк В. Выступ до R на прикладах (укр. яз.). Харьковский национальный экономический университет, 2010. 107 с.

Зорин А. В., Федоткин М. А. Введение в прикладной статистический анализ в пакете R: учебно-метод. пособие. Нижний Новгород: Нижегородский гос. ун-т им. Н. И. Лобачевского, 2010. 50 с.

Коробейников А. И., Малов С. В., Матвеева И. В. Анализ данных с R: Методические указания к спецкурсу «Вычислительные методы и пакеты в статистическом исследовании». СПб., 2010. 12 с.

Коробейников А. И., Малов С. В., Матвеева И. В. Основные алгоритмы численного анализа. Использование пакета R (S-PLUS) для анализа статистических данных: методические указания к практическим занятиям по дисциплине «Вычислительная математика». СПб., 2011. 40 с.

Определение языка R / пер. с англ. Вер. 3.1.0. Рабочая группа разработки R. М., 2014. 64 с.

Савельев А. А., Мухарамова С. С., Пилюгин А. Г. Основные понятия языка R: учебно-метод. пособие. Казань: Казанский университет, 2007. 29 с.

Савельев А. А., Мухарамова С. С., Пилюгин А. Г. Использование языка R для статистической обработки данных: учебно-метод. пособие. Казань: Казанский университет, 2007. 28 с.

Савельев А. А., Мухарамова С. С., Пилюгин А. Г., Чижикова Н. А. Геостатистический анализ данных в экологии и природопользовании (с применением пакета R): учеб. пособие. Казань: Казанский университет, 2012. 120 с.

Савельев А. А., Мухарамова С. С., Чижикова Н. А., Пилюгин А. Г. Теория пространственных точечных процессов в задачах экологии и природопользования (с применением пакета R). Казань: Казанский университет, 2014. 146 с.

Шишкин В. А. Программное обеспечение эконометрических расчетов. Применение пакетов прикладных программ. Пермь: ПГНИУ, 2014. 273 с.

Яу Н. Искусство визуализации в бизнесе: Как представить сложную информацию простыми образами / пер. с англ. А. Кирова. М.: Манн, Иванов и Фербер, 2013. 352 с.

Abedin J. Data Manipulation with R. Packt Publishing, 2014. 103 p.

Adler J. R in a Nutshell: A Desktop Quick Reference. 2nd ed. O'Reilly Media, 2012. 724 p.

Advances in Social Science Research Using R / ed. by H. D. Vinod. Springer, 2010. 205 p.

Aho K. A. Foundational and Applied Statistics for Biologists Using R. CRC Press, 2013.

Aizaki H., Nakatani T., Sato K. Stated Preference Methods Using R. Chapman, Hall/CRC, 2014. 254 p.

Albert J. Bayesian Computation With R. 2nd ed. Springer, 2009. 304 p.

Albert J., Rizzo M. R by Example. Springer, 2012. 374 p.

Allerhand M. A Tiny Handbook of R. Springer, 2011. 94 p.

Altham P. M. Introduction to Statistical Modelling in R. University of Cambridge, 2012. 115 p.

Ando T. Bayesian Model Selection and Statistical Modeling. Chapman, Hall/CRC, 2010. 300 p.

Aragón T. J. Applied Epidemiology Using R. 2013. 240 p.

Armstrong A., Bakker A., Carroll R., Hare C., Poole K. T., Rosenthal H. Analyzing Spatial Models of Choice and Judgment with R Chapman, Hall/CRC, 2014. 351 p.

Arratia A. Computational Finance: An Introductory Course with R. Atlantis Press, 2014. 305 p.

Azzalini A., Capitanio A. The Skew-Normal and Related Families. Cambridge University Press, 2014. 272 p.

Baayen R. H. Analyzing Linguistic Data: A Practical Introduction to Statistics using R. Cambridge University Press, 2008. 369 p.

Barker T. Pro Data Visualization using R and JavaScript. Apress, 2013. 207 p.

Baron J., Li Y. Notes on the use of R for psychology experiments and questionnaires. University of Pennsylvania, 2003. 46 p.

Beaujean A. A. Latent Variable Modeling Using R: A Step-by-Step Guide. Routledge, 2014. 218 p.

Beckerman A. P., Petchey O. L. Getting Started with R: An introduction for Biologists. Oxford University Press, 2012. 128 p.
Beeley C. Web Application Development with R Using Shiny. Packt Publishing, 2013. 110 p.
Berridge D. M., Crouchley R. Multivariate Generalized Linear Mixed Models Using R. CRC Press, 2011. 284 p.
Beyersmann J., Allignol A., Schumacher M. Competing Risks and Multistate Models with R. Springer, 2012. 249 p.
Bioinformatics and Computational Biology Solutions Using R and Bioconductor / ed. by R. Gentleman [et al.]. Springer, 2005. 492 p.
Bivand R. S., Pebesma E., Gómez-Rubio V. Applied Spatial Data Analysis with R. 2nd ed. Springer, 2013. 414 p.
Bliese P. Multilevel Modeling in R. R Development Core Team, 2013. 88 p.
Bloomfield V. A. Computer Simulation and Data Analysis in Molecular Biology and Biophysics: An Introduction Using R. Springer, 2009. 321 p.
Bloomfild V. A. Using R for Numerical Analysis in Science and Engineering. Chapman, Hall/CRC, 2014. 352 p.
Bogetoft P., Otto L. Benchmarking with DEA, SFA, and R. Springer, 2011. 368 p.
Bolker B. M. Ecological Models and Data in R. Princeton University Press, 2008. 517 p.
Boogaart K. G. van den, Tolosana-Delgado R. Analyzing Compositional Data with R. Springer, 2013. 269 p.
Boos D. D., Stefanski L. A. Essential Statistical Inference: Theory and Methods. Springer, 2013. 567 p.
Bonnini S., Corain L., Marozzi M., Salmaso L. Nonparametric Hypothesis Testing: Rank and Permutation Methods with Applications in R. Wiley, 2014. 240 p.
Borcard D., Gillet F., Legendre P. Numerical Ecology with R. Springer, 2011. 319 p.
Braun W. J., Murdoch D. J. A First Course in Statistical Programming with R. Cambridge University Press, 2007.
Bretz F., Hothorn T., Westfall P. Multiple Comparisons Using R. Chapman, Hall/CRC, 2011. 194 p.
Broman K. W., Sen S. A Guide to QTL Mapping with R/qtl. Springer, 2009. 412 p.
Broström G. Event History Analysis with R. CRC Press, 2012. 234 p.
Burns P. The R Inferno. 2011. 126 p.
van Buuren S. Flexible Imputation of Missing Data. Chapman, Hall/CRC, 2012. 342 p.
Cano E. L., Moguerza J. M., Redchuk A. Six Sigma with R: Statistical Engineering for Process Improvement. Springer, 2012. 296 p.
Carmona R. Statistical Analysis of Financial Data in R. 2nd ed. Springer, 2014. 595 p.
Chambers J. M. Programming with Data: A Guide to the S Language. Springer, 1998. 486 p.
Chambers J. M. Software for Data Analysis: Programming with R. Springer, 2008.
Chambers J. M., Hastie T. J. Statistical models in S. Chapman, Hall/CRC, 1991. 608 p.
Chan N. H. Time Series: Applications to Finance with R and S-Plus. 2nd ed. Wiley, 2010. 316 p.
Chang M. Adaptive design theory and implementation using SAS and R. Chapman, Hall/CRC, 2008. 441 p.
Chang S. S. Exploring Everyday Things with R and Ruby: Learning About Everyday Things. O'Reilly Media, 2012. 252 p.
Chang W. R Graphics Cookbook. O'Reilly Media, 2012. 413 p.
Chatterjee S., Simonoff J. S. Handbook of Regression Analysis. Wiley, 2013. 237 p.
Chen D.-G., Peace K. E. Clinical Trial Data Analysis Using R. CRC Press, 2011. 381 p.
Chen D.-G., Peace K. E. Applied Meta-Analysis with R. Chapman, Hall/CRC, 2013. 342 p.
Cheung Y. K. Dose Finding by the Continual Reassessment Method. Chapman, Hall/CRC, 2011. 200 p.
Chihara L. M., Hesterberg T. C. Mathematical Statistics with Resampling and R. Wiley, 2011. 434 p.

Clark M. Introduction to R: A First Course in R. Center for Social Research University of Notre Dame, 2013. 29 p.
Claude J. Morphometrics with R. Springer, 2008. 316 p.
Coghlan A. A Little Book of R for Biomedical Statistics. University College Cork, 2013. 31 p.
Coghlan A. A Little Book of R for Multivariate Analysis. University College Cork, 2013. 47 p.
Coghlan A. A Little Book of R for Time Series. University College Cork, 2014. 71 p.
Cohen Y., Cohen J. Y. Statistics and Data with R: An Applied Approach Through Examples. Wiley, 2008. 603 p.
Computational Actuarial Science with R / ed. by A. Charpentier. Chapman, Hall/CRC, 2014. 604 p.
Cook D., Swayne D. F. Interactive and Dynamic Graphics for Data Analysis: With R and GGobi. Springer, 2007. 188 p.
Cotton R. Learning R. O'Reilly Media, 2013. 400 p.
Cowles M. K. Applied Bayesian Statistics: With R and OpenBUGS Examples. Springer, 2013. 240 p.
Cowpertwait P. S., Metcalfe A. Introductory Time Series with R. Springer, 2009. 262 p.
Crawley M. J. Statistics: An Introduction Using R. 2nd ed. Wiley, 2014. 336 p.
Crawley M. J. The R Book. 2nd ed. Wiley, 2013. 1076 p.
Crowder M. J. Multivariate Survival Analysis and Competing Risks. Chapman, Hall/CRC, 2012. 396 p.
Cryer J. D., Chan K.-S. Time Series Analysis: With Applications in R. 2nd ed. Springer, 2008. 491 p.
Curran J. M. Introduction to Data Analysis with R for Forensic Scientists. CRC Press, 2010. 317 p.
Danneman N., Heimann R. Social Media Mining with R: Deploy cutting-edge sentiment analysis techniques to real-world social media data using R. Packt Publishing, 2014.
Demidenko E. Mixed Models: Theory and Applications with R. 2nd ed. Wiley, 2013. 754 p.
Deshmukh S. Multiple Decrement Models in Insurance: An Introduction Using R. Springer, 2012. 220 p.
Dobrow R. P. Probability: With Applications and R. Wiley, 2014. 520 p.
Douc R., Moulines E., Stoffe D. Nonlinear Time Series: Theory, Methods and Applications with R Examples. Chapman, Hall/CRC, 2014. 551 p.
van der Dudoit S., Laan M. J. Multiple Testing Procedures with Applications to Genomics. Springer, 2008. 611 p.
Eddelbuettel D. Seamless R and C++ Integration with Rccp. Springer, 2013. 220 p.
Ekstrom C. T. The R Primer. Chapman, Hall/CRC, 2011. 299 p.
Ergül Ö. Guide to Programming and Algorithms Using R. Springer, 2013. 200 p.
Everitt B. S. An R and S-Plus Companion to Multivariate Analysis. 2nd ed. Springer, 2007. 221 p.
Everitt B. S., Hothorn T. An Introduction to Applied Multivariate Analysis with R. Springer, 2011. 283 p.
Everitt B. S., Rabe-Hesketh S. Analyzing Medical Data Using S-PLUS. Springer, 2001. 487 p.
Falissard F. Analysis of Questionnaire Data with R. Chapman, Hall/CRC, 2011.
Faraway J. J. Practical Regression and Anova using R. University of Bath, 2002. 2012 p.
Feigelson E. D., Babu G. J. Modern Statistical Methods for Astronomy: With R Applications. Cambridge University Press, 2012. 490 p.
Finch W. H., Bolin J. E., Kelley K. Multilevel Modeling Using R. CRC Press, 2014. 226 p.
Foulkes A. S. Applied Statistical Genetics with R: For Population-based Association Studies. Springer, 2009. 252 p.
Fox J. P. An R and S-Plus Companion to Applied Regression. SAGE Publications, Inc, 2002. 330 p.

Fox J. P. Bayesian Item Response Modeling: Theory and Applications. Springer, 2010. 326 p.
Frery A. C., Perciano T. Introduction to Image Processing Using R: Learning by Examples. Springer, 2013. 95 p.
Gaetan C., Guyon X. Spatial Statistics and Modeling. Springer, 2010. 308 p.
Galecki A., Burzykowski T. Linear Mixed-Effects Models Using R: A Step-by-Step Approach. Springer, 2013. 625 p.
Galwey N. W. Introduction to Mixed Modelling: Beyond Regression and Analysis of Variance. Wiley, 2006. 377 p.
Gardener M. Beginning R: The Statistical Programming Language. Wrox, 2012. 504 p.
Gardener M. The Essential R Reference. Wiley, 2013. 578 p.
Gentleman R. R Programming for Bioinformatics. Chapman, Hall/CRC, 2008. 328 p.
Gilli M., Maringer D., Schumann E. Numerical Methods and Optimization in Finance. Academic Press, 2011. 584 p.
Good P. I. Introduction to Statistics Through Resampling Methods and R. 2nd ed. Wiley, 2013. 217 p.
Greenacre M. Correspondence Analysis in Practice. 2nd ed. Chapman, Hall/CRC, 2007. 296 p.
Gries S. T. Quantitative Corpus Linguistics with R: A Practical Introduction. Routledge, 2009. 256 p.
Gries S. T. Statistics for Linguistics with R: A Practical Introduction. 2nd ed. De Gruyter Mouton, 2013. 375 p.
Hahne F., Huber W., Gentleman R., Falcon S. Bioconductor Case Studies. Springer, 2008. 296 p.
Hay-Jahans C. An R Companion to Linear Statistical Models. Chapman, Hall/CRC, 2011. 372 p.
Heiberger R. M., Holland B. Statistical Analysis and Data Display: An Intermediate Course with Examples in S-Plus, R, and SAS. Springer, 2004. 730 p.
Heiberger R. M., Neuwirth E. R Through Excel: A Spreadsheet Interface for Statistics, Data Analysis, and Graphics. Springer, 2009. 344 p.
Helsel D. R. Statistics for Censored Environmental Data Using Minitab and R. 2nd ed. Wiley, 2012. 344 p.
Henry M., Stevens H. A Primer of Ecology with R. Springer, 2009. 388 p.
Hettmansperger T. P., McKean J. W. Robust Nonparametric Statistical Methods and R. 2nd ed. CRC Press, 2011. 553 p.
Hilbe J. M., Robinson A. P. Methods of Statistical Model Estimation. Chapman, Hall/CRC, 2013. 243 p.
Hoff P. D. A First Course in Bayesian Statistical Methods. Springer, 2009. 270 p.
Højsgaard S., Edwards D., Lauritzen S. Graphical Models with R. Springer, 2012. 182 p.
Hollander M., Wolfe D. A., Chicken E. Nonparametric Statistical Methods. 3rd ed. Wiley, 2014. 848 p.
Horton N. J., Kleinman K. Using R for Data Management, Statistical Analysis, and Graphics. CRC Press, 2010. 296 p.
Huet S., Bouvier A., Poursat M.-A., Jolivet E. Statistical Tools for Nonlinear Regression: A Practical Guide With S-PLUS and R Examples. 2nd ed. Springer, 2003. 249 p.
Husson F., Le S., Pages J. Exploratory Multivariate Analysis by Example Using R. CRC Press, 2010. 240 p.
Hyndman R. J., Athanasopoulos G. Forecasting: principles and practice. Otexts, 2014. 292 p. (официальная бесплатная версия книги доступна по адресу https://www.otexts.org/fpp).
Iacus S. M. Simulation and Inference for Stochastic Differential Equations: With R Examples. Springer, 2008. 286 p.
Iacus S. M. Option Pricing and Estimation of Financial Models with R. Wiley, 2011. 472 p.

James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning: with Applications in R[1]. Springer, 2013. 426 p.
Janert P. K. Data Analysis with Open Source Tools. O'Reilly Media, 2010. 540 p.
Jank W. Business Analytics for Managers. Springer, 2011. 189 p.
Jockers M. L. Text Analysis with R for Students of Literature. Springer, 2014. 199 p.
Jones O., Maillardet R., Robinson A. P. Introduction to Scientific Programming and Simulation using R. Chapman, Hall/CRC, 2009. 472 p.
de Jonge E., van der Loo M. An Introduction to Data Cleaning with R. Statistics Netherlands, 2013. 53 p.
Jureckova J., Picek J. Robust Statistical Methods with R. Chapman, Hall/CRC, 2005. 216 p.
Kaas R., Goovaerts M., Dhaene J., Denuit M. Modern Actuarial Risk Theory – Using R. 2nd ed. Springer, 2008. 394 p.
Kaltenbach H.-M. A Concise Guide to Statistics. Springer, 2011. 111 p.
Karian Z. A., Dudewicz E. J. Handbook of Fitting Statistical Distributions with R. Chapman, Hall/CRC, 2011.
Kateri M. Contingency Table Analysis: Methods and Implementation Using R. Springer, 2014. 315 p.
Keele L. Semiparametric Regression for the Social Sciences. Wiley, 2008. 230 p.
Kenett R., Salini S. Modern Analysis of Customer Surveys: with Applications Using R. Wiley, 2012. 524 p.
Kenett R., Zacks S. Modern Industrial Statistics: with applications in R, MINITAB and JMP. 2nd ed. Wiley, 2014. 586 p.
Kerns G. J. Introduction to Probability and Statistics Using R. G. Jay Kerns, 2010. 396 p.
Kleiber C., Zeileis A. Applied Econometrics with R. Springer, 2008. 222 p.
Kleinman K., Horton N. J. SAS and R: Data Management, Statistical Analysis and Graphics. 2nd ed. Chapman, Hall/CRC, 2014. 425 p.
Klemelä J. Smoothing of Multivariate Data: Density Estimation and Visualization. Wiley, 2009. 603 p.
Klemelä J. Multivariate Nonparametric Regression and Visualization: With R and Applications to Finance. Wiley, 2014. 396 p.
Knoblauch K., Maloney L. T. Modeling Psychophysical Data in R. Springer, 2012. 367 p.
Kolaczyk E. D., Csárdi G. Statistical Analysis of Network Data with R. Springer, 2014. 207 p.
Krause A., Olson M. The Basics of S-PLUS. 4th ed. Springer, 2005. 455 p.
Krijnen W. Applied Statistics for Bioinformatics Using R. Hanze University, 2009. 278 p.
Kruschke J. K. Doing Bayesian Data Analysis: A Tutorial Introduction with R and BUGS. Academic Press, 2010. 542 p.
Kuhn M.,Johnson K. Applied Predictive Modeling. Springer, 2013. 600 p.
Kuhnert P. M., Venables W. N. An Introduction to R: Software for Statistical Modelling & Computing. CSIRO Mathematical, Information Sciences, Australia, 2005. 364 p.
Lamigueiro O. P. Displaying Time Series, Spatial, and Space-Time Data with R. Chapman, Hall/CRC, 2014. 206 p.
Lantz B. Machine Learning with R. Packt Publishing, 2013. 396 p.
Lavine M. L. Introduction to Statistical Thought. 2013. 463 p.
Lawrence M. F., Verzani J. Programming Graphical User Interfaces in R. Chapman, Hall/CRC, 2012. 479 p.
Ledolter J. Data Mining and Business Analytics with R. Wiley, 2013.
Leipzig J., Li X. Data Mashups in R. O'Reilly Media, 2011. 40 p.
Li Y., Baron J. Behavioral Research Data Analysis with R. Springer, 2012. 245 p.

[1] В настоящее время готовится перевод данной книги.

Lindsey J. K. Statistical Analysis of Stochastic Processes in Time. Cambridge University Press, 2004. 354 p.
Link *W. A., Barker R. J.* Bayesian Inference: with ecological applications. Elsevier/Academic, Boston, MA. 2010. 339 p.
van der Loo M. P., de Jonge E. Learning RStudio for R Statistical Computing. Packt Publishing, 2012. 126 p.
Lovelace R., Cheshire J. Introduction to Visualising Spatial Data in R. 2014. 24 p.
Lumley T. Complex Surveys: A Guide to Analysis Using R. Wiley, 2010. 297 p.
Lumley T. R Fundamentals and Programming Techniques. R Core Development Team. 2006. 225 p.
MacFarland T. W. Two-Way Analysis of Variance: Statistical Tests and Graphics Using R. Springer, 2012. 150 p.
MacFarland T. W. Introduction to Data Analysis and Graphical Presentation in Biostatistics with R: Statistics in the Large. Springer, 2014. 167 p.
Mahajan S. Street-Fighting Mathematics: The Art of Educated Guessing and Opportunistic Problem Solving. The MIT Press, 2010. 152 p.
Maindonald J. H. Using R for Data Analysis and Graphics. Introduction, Code and Commentary. Australian National University, 2008. 88 p.
Marin J.-M., Robert C. Bayesian Essentials with R. 2nd ed. Springer, 2014. 312 p.
Marques de Sá J. Applied Statistics Using SPSS, STATISTICA, MATLAB and R. Springer, 2007. 520 p.
Martinussen T., Scheike T. H. Dynamic Regression Models for Survival Data. Springer, 2006. 470 p.
Mathur S. K. Statistical Bioinformatics with R. Academic Press, 2010. 336 p.
Matloff N. The Art of R Programming: A Tour of Statistical Software Design. No Starch Press, 2011. 400 p.
Matthiopoulos J. How to be a Quantitative Ecologist: The 'A to R' of Green Mathematics and Statistics. Wiley, 2011. 490 p.
McCallum Q. E., Weston S. Parallel R. O'Reilly Media, 2011. 126 p.
McCarthy M. A. Bayesian Methods for Ecology. Cambridge University Press, 2007. 312 p.
Micheaux P. L. de, Drouilhet R., Liquet B. The R Software: Fundamentals of Programming and Statistical Analysis. Springer, 2013. 628 p.
Millard S. P. EnvStats: An R Package for Environmental Statistics. 2nd ed. Springer, 2013. 305 p.
Millard S. P., Neerchal N. K., Dixon P. Environmental Statistics with S-PLUS. CRC Press, 2000. 848 p.
Mirman D. Growth Curve Analysis and Visualization Using R. Chapman, Hall/CRC, 2014. 188 p.
Mittal H. V. R Graphs Cookbook. Packt Publishing, 2011. 272 p.
Morton A., Mengersen K. L., Playford G., Whitby M. Statistical Methods for Hospital Monitoring with R. Wiley, 2013. 426 p.
Muenchen R. A. R for SAS and SPSS Users. 2nd ed. Springer, 2011. 715 p.
Muenchen R. A., Hilbe J. M. R for Stata Users. Springer, 2010. 542 p.
Murrell P. R Graphics. Chapman, Hall/CRC, 2005. 322 p.
Murtagh F. Correspondence Analysis and Data Coding with Java and R. Chapman, Hall/CRC, 2005. 256 p.
Myers W. L., Patil G. P. Multivariate Methods of Representing Relations in R for Prioritization Purposes. Springer, 2012. 304 p.
Nagarajan R., Scutari M., Lebre S. Bayesian Networks in R: with Applications in Systems Biology. Springer, 2013. 168 p.
Nason G. P. Wavelet Methods in Statistics with R. Springer, 2008. 268 p.
Navarro D. J. Learning Statistics with R. lulu.com, 2014. 562 p.

Nolan D., Temple-Lang D. XML and Web Technologies for Data Sciences with R. Springer, 2014. 677 p.
O'Gorman T. W. Adaptive Tests of Significance Using Permutations of Residuals with R and SAS. Wiley, 2012. 364 p.
Ohri A. R for Business Analytics. Springer, 2013. 319 p.
Oja H. Multivariate Nonparametric Methods with R: An approach based on spatial signs and ranks. Springer, 2010. 244 p.
Owen J. The R Guide. University of Richmond, 2010. 61 p.
Pace L. Beginning R: An Introduction to Statistical Programming. Apress, 2012. 336 p.
Paradis E. R for Beginners. Universite Montpellier, 2005. 76 p.
Paradis E. Analysis of Phylogenetics and Evolution with R. 2nd ed. Springer, 2012. 404 p.
Peng R. D., Dominici F. Statistical Methods for Environmental Epidemiology with R: A Case Study in Air Pollution and Health. Springer, 2008. 154 p.
Petris G., Petrone S., Campagnoli P. Dynamic Linear Models with R. Springer, 2009. 252 p.
Pfaff B. Analysis of Integrated and Cointegrated Time Series with R. 2nd ed. Springer, 2008. 190 p.
Pfaff B. Financial Risk Modelling and Portfolio Optimization with R. Wiley, 2013. 374 p.
Plant R. E. Spatial Data Analysis in Ecology and Agriculture Using R. CRC Press, 2012. 637 p.
Prajapati V. Big Data Analytics with R and Hadoop. Packt Publishing, 2013. 238 p.
Pruscha H. Statistical Analysis of Climate Series: Analyzing, Plotting, Modeling, and Predicting with R. Springer, 2013. 200 p.
Quick J. M. Statistical Analysis with R Beginner's Guide. Packt Publishing, 2010. 300 p.
Ramsay J., Hooker G., Graves S. Functional Data Analysis with R and MATLAB. Springer, 2009. 213 p.
Rasch D., Pilz J., Verdooren L. R., Gebhardt A. Optimal Experimental Design with R. Chapman, Hall/CRC, 2011. 345 p.
Rasch D., Kubinger K., Yanagida T. Statistics in Psychology Using R and SPSS. Wiley, 2011. 568 p.
Rayner J. C., Thas O., Best D. J. Smooth Tests of Goodness of Fit: Using R. 2nd ed. Wiley, 2009. 290 p.
Reimann C., Filzmoser P., Garrett R., Dutter R. Statistical Data Analysis Explained: Applied Environmental Statistics with R. Wiley, 2008. 359 p.
Ritz C., Streibig J. C. Nonlinear Regression with R. Springer, 2008. 148 p.
Rizopoulos D. Joint Models for Longitudinal and Time-to-Event Data: With Applications in R. Chapman, Hall/CRC, 2012. 275 p.
Robert C., Casella G. Introducing Monte Carlo Methods with R. Springer, 2010. 284 p.
Robinson A. P. Icebreake R. University of Melbourne, 2010. 159 p.
Robinson A. P., Hamann J. D. Forest Analytics with R: An Introduction. Springer, 2011. 354 p.
Robinson R., White H. Elementary Statistics. 2014. 404 p.
Rosenbaum P. R. Design of Observational Studies. Springer, 2010. 385 p.
Rousseeuw P. J., Ruts I., Tukey J. W. The bagplot: a bivariate boxplot // The American Statistician. 1999. V. 53, № 4. P. 382–387.
Ruppert D. Statistics and Data Analysis for Financial Engineering. Springer, 2011. 638 p.
Sanchez G. Handling and Processing Strings in R. 2013. 113 p.
Santos-Fernández E. Multivariate Statistical Quality Control Using R. Springer, 2012. 127 p.
Schafer J. L. Analysis of Incomplete Multivariate Data. Chapman, Hall/CRC, 1997. 444 p.
Schumacker R., Tomek S. Understanding Statistics Using R. Springer, 2013. 296 p.
Schumacker R. A., Akers A. Understanding Statistical Concepts Using S-Plus. Lawrence Erlbaum Associates, 2001. 384 p.

Seefeld K., Linder E. Statistics Using R with Biological Examples. University of New Hampshire, Durham, 2007. 324 p.
Selvin S. Modern Applied Biostatistical Methods: Using S-Plus. Oxford University Press, 1998. 476 p.
Shahbaba B. Biostatistics with R: An Introduction to Statistics Through Biological Data. Springer, 2012. 368 p.
Shalizi C. Advanced Data Analysis from an Elementary Point of View. Spring, 2013. 586 p.
Sheather S. A Modern Approach to Regression with R. Springer, 2009. 393 p.
Shumway R. H., Stoffer D. S. Time Series Analysis and Its Applications: With R Examples. 3rd ed. Springer, 2011. 604 p.
Siegmund D., Yakir B. The Statistics of Gene Mapping. Springer, 2007. 337 p.
Sinha P. P. Bioinformatics with R Cookbook. Packt Publishing, 2014. 340 p.
Soetaert K., Cash J., Mazzia F. Solving Differential Equations in R. Springer, 2012. 248 p.
Soetaert K., Herman P. M. J. A Practical Guide to Ecological Modelling: Using R as a Simulation Platform. Springer, 2009. 372 p.
Spector P. Data Manipulation with R. Springer, 2008. 152 p.
Stanton J. M. Introduction to Data Science. 2013. 182 p.
Steyerberg E. W. Clinical Prediction Models: A Practical Approach to Development, Validation, and Updating. Springer, 2009. 500 p.
Stodden V., Leisch F., Peng R. D. Implementing Reproducible Research. Chapman, Hall/CRC, 2014. 448 p.
Stowell S. Using R for Statistics. Apress, 2014. 232 p.
Suess E. A., Trumbo B. E. Introduction to Probability Simulation and Gibbs Sampling with R. Springer, 2010. 307 p.
Shumway R., Stoffer D. Time Series Analysis and Its Applications. With R Examples. 3rd ed. Springer, 2010. 610 p.
Swenson N. G. Functional and Phylogenetic Ecology in R. Springer, 2014. 212 p.
Takezawa K. Introduction to Nonparametric Regression. Wiley, 2005. 568 p.
Takezawa K. Guidebook to R Graphics Using Microsoft Windows. Wiley, 2013. 272 p.
Takezawa K. Learning Regression Analysis by Simulation. Springer, 2014. 310 p.
Tattar P. N. R Statistical Application Development by Example Beginner's Guide. Packt Publishing, 2013. 344 p.
Teetor P. 25 Recipes for Getting Started with R: Excerpts from the R Cookbook. O'Reilly Media, 2011. 62 p.
Teetor P. R Cookbook. O'Reilly Media, 2011. 438 p.
Thiem A., Duşa A. Qualitative Comparative Analysis with R: A User S Guide. Springer, 2013. 99 p.
Tollefson M. R Quick Syntax Reference. Apress, 2014. 209 p.
Torgo L. Data Mining with R: Learning with Case Studies. Chapman, Hall/CRC, 2010. 305 p.
Trosset M. W. An Introduction to Statistical Inference and Its Applications with R. Chapman, Hall/CRC, 2009. 496 p.
Tsay R. S. An Introduction to Analysis of Financial Data with R. Wiley, 2012. 416 p.
Tsay R. S. Multivariate Time Series Analysis: With R and Financial Applications. Wiley, 2014. 522 p.
Tufféry S. Data Mining and Statistics for Decision Making. Wiley, 2011. 704 p.
Valiente G. Combinatorial Pattern Matching Algorithms in Computational Biology Using Perl and R. Chapman, Hall/CRC, 2009. 368 p.
Varmuza K., Filzmoser P. Introduction to Multivariate Statistical Analysis in Chemometrics. CRC Press, 2009. 336 p.
Vasishth S., Broe M. The Foundations of Statistics: A Simulation-based Approach. Springer, 2010. 195 p.

Venables W. N., Ripley B. D. S programming. Springer, 2000. 265 p.
Verzani J. Using R for Introductory Statistics. Chapman, Hall/CRC, 2004. 432 p.
Verzani J. Getting Started with RStudio. O'Reilly Media, 2011. 98 p.
Voss J. An Introduction to Statistical Computing: A Simulation-based Approach. Wiley, 2013. 396 p.
Wehrens R. Chemometrics with R: Multivariate Data Analysis in the Natural Sciences and Life Sciences. Springer, 2011. 285 p.
Wickham H. Practical tools for exploring data and models. 2008. 122 p.
Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer, 2009. 213 p.
Wienke A. Frailty Models in Survival Analysis. Chapman, Hall/CRC, 2010. 312 p.
Wilcox R. R. Introduction to Robust Estimation and Hypothesis Testing. 3rd ed. Academic Press, 2012. 690 p.
Wildi O. Data Analysis in Vegetation Ecology. 2nd ed. Wiley, 2013. 315 p.
Willekens F. Multistate Analysis of Life Histories with R. Springer, 2014. 308 p.
Williams G. Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery. Springer, 2011. 395 p.
Wright D. B., London K. Modern Regression Techniques Using R: A Practical Guide. SAGE Publications Ltd, 2009. 216 p.
Wünschiers R. Computational Biology: A Practical Introduction to BioData Processing and Analysis with Linux, MySQL, and R. 2nd ed. Springer, 2013. 449 p.
Würtz D., Setz T., Chalabi Y., Lam L., Ellis A. Basic R for Finance. Zurich: Rmetrics Association & Finance Online Publishing, 2010. 312 p.
Yakir B. Introduction to Statistical Thinking. The Hebrew University, 2011. 314 p.
Yan X., Su X. G. Linear Regression Analysis: Theory and Computing. World Scientific Publishing Company, 2009. 348 p.
Zhao Y. R and Data Mining: Examples and Case Studies. Academic Press, 2012. 256 p.
Zhao Y., Cen Y. Data Mining Applications with R. Academic Press, 2013. 514 p.
Zivot E., Wang J. Modeling Financial Time Series with S-PLUS. 2nd ed. Springer, 2006. 998 p.
Zucchini W., MacDonald I. L. Hidden Markov Models for Time Series: An Introduction Using R. Chapman, Hall/CRC, 2009. 278 p.
Zumel N., Mount J. Practical Data Science with R. Manning Publications, 2014. 416 p.
Zuur A. F., Ieno E. N., Meesters E. A Beginner's Guide to R. Springer, 2009. 220 p.
Zuur A. F., Ieno E. N., Smith G. M. Analysing Ecological Data. Springer, 2007. 672 p.

Рекомендуемые интернет-ресурсы

Русскоязычные ресурсы

Сообщество «Язык и среда R» на платформе LiveJournal: новости из мира R, обмен опытом: http://r-statistics.livejournal.com.
Группа по R на платформе ВКонтакте: новости из мира R, обмен опытом: http://vk.com/rstatistics.
Русскоязычная консультация по R на форуме «Классическая и молекулярная биология»: http://bit.ly/1E8LHwH.
Ветка по R на форуме сообщества GIS-Lab: http://bit.ly/1DyTAJq.
«Введение в R» на русском языке: http://bit.ly/1Gvqq1k.
Заметки по R в блоге «Записки разработчика» А. Акиньшина: http://aakinshin.blogspot.ru/search/label/r.
Материалы по R Е. Балдина: http://bit.ly/1QwI9eg.
Заметки по R в блоге А. Виноградова: http://alexwin1961.livejournal.com/tag/r.

Заметки по R в блоге Р. Габидуллина: http://voliadis.ru/taxonomy/term/1.
Блог А. Дубасовой «Статистический анализ данных в экспериментальной лингвистике. Программа R» (с видеоуроками): http://statinr.blogspot.com.
Блог «Анализ малых данных» А. Дьяконова: новости, обзоры, рекомендации по анализу данных: https://alexanderdyakonov.wordpress.com.
Заметки по R в блоге «О программировании, алгоритмах и не только» С. Едунова: http://www.algorithmist.ru/search/label/R.
Блог «Язык R в финансах и экономике» И. Езепова: http://rforfinance.ru.
Заметки по R в «Блоге странного ученого» М. Касянчука: http://bit.ly/1HHx5qx.
Блог «R: Анализ и визуализация данных» С. Мастицкого: http://r-analytics.blogspot.com.
Заметки по R на сайте «Матрунич консалтинг» А. Матрунича: http://matrunich.com/tags/r.
Блог «Биостатистика и язык R» А. Огурцова: http://biostat-r.blogspot.com.
Заметки по R в блоге «Когнитивная психология и эмоции» А. Четверикова: http://chetvericov.ru/tag/r.
Заметки по R в блоге «Электронные семечки» О. Шмелева: http://bit.ly/1Glubns.
Блог «R по-русски»: https://rrus.wordpress.com.
Вики-учебник по R: http://bit.ly/1HJ6c7i.
Серия статей по R на портале Хабрахабр: http://habrahabr.ru/hub/r.
Библиографические материалы по R на сайте «Энциклопедия психодиагностики Psylab.info»: http://psylab.info/R:Литература.
Datareview.info: информационно-образовательный портал, посвященный вопросам анализа и обработки данных: http://datareview.info.

Англоязычные ресурсы

Официальный сайт R-проекта: http://www.r-project.org.
Репозиторий *CRAN* – более 6200 дополнительных пакетов для R: http://cran.r-project.org.
Репозиторий *Bioconductor* – дополнительные пакеты для R, преимущественно нацеленные на обработку данных из области биоинформатики: http://www.bioconductor.org.
Репозиторий *The Omega Project for Statistical Computing* – разнообразные дополнительные пакеты для R: http://www.omegahat.org.
Архив выпусков постоянного журнала *The R Journal:* http://journal.r-project.org/archive.
Архив выпусков журнала *The Journal of Statistical Software*: http://www.jstatsoft.org.
R-bloggers – агрегатор сообщений из блогов, посвященных R: http://www.r-bloggers.com.
Вопросы/ответы по R на портале *Stackoverflow*: http://bit.ly/1FodwDv.
Inside-R – справочные материалы и другие ресурсы по R: http://www.inside-r.org.
Quick-R – справочник по использованию R: http://www.statmethods.net.
Обучающие материалы от Калифорнийского университета в Лос-Анджелесе: http://statistics.ats.ucla.edu/stat/r.
Cookbook for R – сборник рецептов: http://www.cookbook-r.com.
R Tutorial Series – большой сборник обучающих материалов: http://rtutorialseries.blogspot.com.
Курс лекций по R от профессора V. Zoonekynd: http://bit.ly/1dgEfH3.
Блог K. Cichini с примерами использования R в биологических исследованиях (и не только): http://thebiobucket.blogspot.com.
Курс лекций по количественным методам в географии от E. Root: http://bit.ly/1J0WYR6.
Страница профессора K. Strimmer из Имперского колледжа Лондона, на которой представлен ряд полезных авторских пакетов для R: http://strimmerlab.org/software.html.
Лабораторные работы по статистике в R для студентов-экологов: http://ecology.msu.montana.edu/labdsv/R.
Страница профессора P. Legendre, содержащая много авторских скриптов и функций R: http://bit.ly/1E8UMpv.

Книги издательства «ДМК Пресс» можно заказать
в торгово-издательском холдинге «Планета Альянс» наложенным платежом,
выслав открытку или письмо по почтовому адресу:
115487, г. Москва, 2-й Нагатинский пр-д, д. 6А.
При оформлении заказа следует указать адрес (полностью),
по которому должны быть высланы книги;
фамилию, имя и отчество получателя.
Желательно также указать свой телефон и электронный адрес.
Эти книги вы можете заказать и в интернет-магазине: **www.alians-kniga.ru**.
Оптовые закупки: тел. **(499) 782-38-89**.
Электронный адрес: **books@alians-kniga.ru**.

Мастицкий Сергей Эдуардович
Шитиков Владимир Кириллович

Статистический анализ и визуализация данных с помощью R

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Корректор *Синяева Г. И.*
Верстка *Чаннова А. А.*
Дизайн обложки *Мовчан А. Г.*

Формат 70×100 1/16.
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 16,5. Тираж 300 экз.

Веб-сайт издательства: www.дмк.рф