O'REILLY®

Погружение в аналитику данных

От Excel к Python и R

bhv®

Джордж Маунт

Advancing into Analytics

From Excel to Python and R

George Mount

Beijing · Boston · Farnham · Sebastopol · Tokyo O'REILLY

Джордж Маунт

Погружение в аналитику данных

Санкт-Петербург
«БХВ-Петербург»
2023

УДК 004.43
ББК 32.973.26-018.1
М35

Маунт Дж.

М35 Погружение в аналитику данных: Пер. с англ. — СПб.: БХВ-Петербург, 2023. — 224 с.: ил.

ISBN 978-5-9775-6866-1

В книге приводятся практические приемы анализа данных. Рассказано, как исследовать и тестировать взаимосвязи между переменными в Excel и использовать его для статистики и анализа. Описан перенос данных из Excel в R, язык программирования с открытым исходным кодом, специально разработанный для выполнения статистического анализа. Отдельный раздел посвящен переносу данных из Excel в Python и выполнению полного анализа данных средствами этого языка. В результате читатель научится выполнять разведочный анализ данных (Exploratory Data Analysis, EDA) и проверку гипотез с использованием языков программирования Python и R.

Для аналитиков данных

УДК 004.43
ББК 32.973.26-018.1

Группа подготовки издания:

Руководитель проекта	*Павел Шалин*
Зав. редакцией	*Людмила Гауль*
Компьютерная верстка	*Ольги Сергиенко*
Оформление обложки	*Зои Канторович*

Подписано в печать 02.02 23
Формат 70×100 1/16 Печать офсетная. Усл. печ. л. 18,06
Тираж 1000 экз Заказ № 6025.
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20

Отпечатано с готового оригинал-макета
ООО "Принт-М", 142300, М О , г Чехов, ул. Полиграфистов, д. 1

ISBN 978-1-492-09434-0 (англ.)
ISBN 978-5-9775-6866-1 (рус.)

Оглавление

Предисловие **9**
Цель обучения 9
Предварительные условия 9
Технические требования 9
Требования к предварительной подготовке 10
Как я пришел к аналитике 10
«Excel — плохо, программирование — хорошо» 11
Преимущества Excel при обучении аналитике 12
Обзор книги 13
Упражнения в конце глав 13
Эта книга — не список готовых решений 14
Без паники! 14
Условные обозначения 14
Использование примеров кода 15
Контакты 16
Благодарности 16

ЧАСТЬ I. ОСНОВЫ АНАЛИТИКИ В EXCEL **17**

Глава 1. Основы разведочного анализа данных **19**
Что такое разведочный анализ данных? 19
Наблюдения 21
Переменные 21
Категориальные переменные 22
Количественные переменные 24
Закрепление материала: классификация переменных 25
Резюме: типы переменных 27
Исследование переменных в Excel 27
Исследование категориальных переменных 27
Исследование количественных переменных 30
Заключение 41
Упражнения 41

Глава 2. Понятие вероятности **42**
Вероятность и случайность 42
Вероятность и выборочное пространство 42
Вероятность и эксперименты 43
Безусловная и условная вероятность 43

Распределение вероятностей ... 43
Дискретное распределение вероятностей ... 44
Непрерывное распределение вероятностей ... 47
Заключение ... 55
Упражнения ... 55

Глава 3. Основы инференциальной статистики ... 56
Базовые понятия статистического вывода ... 56
Сбор данных для репрезентативной выборки ... 57
Формулирование гипотез ... 58
Разработка плана анализа ... 59
Анализ данных ... 62
Принятие решения ... 64
Это ваш мир... данные только живут в нем ... 71
Заключение ... 72
Упражнения ... 73

Глава 4. Корреляция и регрессия ... 74
«Корреляция не подразумевает причинно-следственную связь» ... 74
Понятие корреляции ... 75
От корреляции к регрессии ... 80
Линейная регрессия в Excel ... 81
Переосмысление результатов: ложные зависимости ... 87
Заключение ... 88
Переход к программированию ... 89
Упражнения ... 89

Глава 5. Стек анализа данных ... 90
Статистика, аналитика и наука о данных ... 90
Статистика ... 90
Аналитика данных ... 90
Бизнес-аналитика ... 91
Наука о данных ... 91
Машинное обучение ... 91
Различия без взаимоисключения ... 92
Значение стека анализа данных ... 92
Электронные таблицы ... 93
VBA ... 94
Современный Excel ... 95
Базы данных ... 96
Платформы бизнес-аналитики (BI) ... 97
Языки программирования для анализа данных ... 98
Заключение ... 99
Что будет дальше ... 100
Упражнения ... 100

ЧАСТЬ II. ОТ EXCEL К R ... 101

Глава 6. Первые шаги в R для пользователей Excel ... 103
Загрузка R ... 103
Начало работы с RStudio ... 103

Пакеты в R ... 112
Обновление R, RStudio и пакетов R ... 114
Заключение ... 114
Упражнения ... 116

Глава 7. Структуры данных в R ... 117
Векторы ... 117
Индексирование и подмножества векторов ... 119
От таблиц Excel к кадрам данных R ... 120
Импорт данных в R ... 122
Исследование кадра данных ... 126
Индексирование и подмножества кадров данных ... 128
Запись кадров данных ... 129
Заключение ... 130
Упражнения ... 130

Глава 8. Обработка и визуализация данных в R ... 131
Обработка данных с помощью пакета *dplyr* ... 131
Постолбцовые операции ... 132
Построчные операции ... 135
Агрегирование и объединение данных ... 137
dplyr и оператор *pipe* (%>%) ... 141
Преобразование данных с помощью *tidyr* ... 142
Визуализация данных с помощью *ggplot2* ... 145
Заключение ... 151
Упражнения ... 151

Глава 9. Кульминация: R для анализа данных ... 152
Разведочный анализ данных ... 153
Проверка гипотез ... 157
t-тест для независимых выборок ... 157
Линейная регрессия ... 159
Разделение и проверка данных для обучения и тестирования ... 161
Заключение ... 164
Упражнения ... 164

ЧАСТЬ III. ОТ EXCEL К PYTHON ... 165

Глава 10. Первые шаги в Python для пользователей Excel ... 167
Загрузка Python ... 167
Начало работы с Jupyter ... 168
Модули в Python ... 176
Обновление Python, Anaconda и пакетов Python ... 178
Заключение ... 178
Упражнения ... 178

Глава 11. Структуры данных в Python ... 180
Массивы *NumPy* ... 181
Индексирование и подмножества массивов *NumPy* ... 182
Кадры данных Pandas ... 184

Импорт данных в Python ... 185
Исследование кадра данных ... 187
Индексирование и подмножества кадров данных ... 188
Запись кадров данных ... 189
Заключение ... 189
Упражнения ... 189

Глава 12. Обработка и визуализация данных в Python ... 191
Постолбцовые операции ... 192
Построчные операции ... 194
Агрегирование и объединение данных ... 195
Преобразование данных ... 197
Визуализация данных ... 199
Заключение ... 203
Упражнения ... 203

Глава 13. Кульминация: Python для анализа данных ... 204
Разведочный анализ данных ... 205
Проверка гипотез ... 207
t-тест для независимых выборок ... 207
Линейная регрессия ... 208
Разделение и проверка данных для обучения и тестирования ... 210
Заключение ... 212
Упражнения ... 212

Глава 14. Заключение и дальнейшие шаги ... 213
Дополнительные элементы стека анализа данных ... 213
План исследований и бизнес-эксперименты ... 213
Дополнительные статистические методы ... 214
Наука о данных и машинное обучение ... 214
Контроль версий ... 214
Этика ... 215
Двигайтесь вперед и выбирайте то, что нравится ... 215
Напутствие ... 216

Предметный указатель ... 217

Об авторе ... 221

Об изображении на обложке ... 222

Предисловие

Вы приступаете к важному и содержательному учебному курсу, который включает в себя вопросы статистики, программирования и многое другое. Прежде чем начать наше путешествие, я хотел бы познакомить вас с задачами данного курса и рассказать, как я пришел к написанию этой книги и чего следует от нее ожидать.

Цель обучения

К концу книги вы научитесь выполнять *разведочный анализ данных* (exploratory data analysis, EDA) и *проверку гипотез, используя язык программирования*. Исследование и тестирование взаимосвязей — основа аналитики. В этой книге вы познакомитесь с инструментами и средствами, благодаря которым сможете перейти к освоению более совершенных методов анализа данных без особого труда.

Мы будем использовать такие мощные инструменты, как Excel, R и Python, которые позволят сделать процесс обучения непрерывным. Эта комбинация описана всего в нескольких книгах, хотя для аналитиков, включая меня, переход от электронных таблиц к программированию — обычное явление.

Предварительные условия

Для достижения поставленных целей необходимо выполнение указанных далее условий.

Технические требования

Я пишу эту книгу на компьютере с Windows и установленным Excel для настольного ПК из пакета Office 365. Если на вашем компьютере установлена платная версия Excel 2010 или более поздняя для Windows или Mac, вы сможете выполнять большинство инструкций из этой книги, за исключением, возможно, сводных таблиц и визуализации данных.

Несмотря на то что доступны как бесплатная, так и платная версии Excel, для некоторых функций, описанных в этой книге, понадобится платная десктопная версия.

Языки R и Python являются бесплатными инструментами с открытым исходным кодом, доступным для всех основных операционных систем. Далее в этой книге содержатся инструкции по их установке.

Требования к предварительной подготовке

Для изучения и понимания этой книги не требуется знания языков R или Python; тем не менее нужно знание Excel на среднем уровне.

Необходимо знать следующие темы по Excel, чтобы пройти учебный курс без помех.

- Абсолютные, относительные и смешанные ссылки.
- Условная логика и условное агрегирование (операторы `IF()`, `SUMIF()/SUMIFS()` и т. д.).
- Объединение источников данных (`VLOOKUP()`, `INDEX()/MATCH()` и т. д).
- Сортировка, фильтрация и агрегирование данных с помощью сводных таблиц.
- Основы построения графиков (гистограммы, линейные диаграммы и т. д).

Если вы хотите попрактиковаться в этих вопросах, прежде чем двигаться дальше, советую обратиться к книге Майкла Александера и соавт. «Excel 2019. Библия пользователя»[1] (Michael Alexander et al. Excel 2019 Bible).

Как я пришел к аналитике

Как и многие другие, к аналитике я пришел окольным путем. В школе я активно избегал математики, потому что слишком многое в ней казалось абсолютно теоретическим. Позже я делал курсовую работу по статистике и эконометрике, и это меня заинтересовало. *Практическое применение* математики к какой-то конкретной задаче показалось мне глотком свежего воздуха.

Очевидно, что такого знакомства со статистикой было недостаточно. Я учился в гуманитарном колледже, где получил прочные навыки в области письма и мышления, но очень незначительные знания в точных науках. Когда я получил первую постоянную работу, то был поражен объемом данных, которыми мне пришлось управлять. Бóльшая часть этих данных содержалась в электронных таблицах, и извлечь из них пользу без тщательной очистки и подготовки было весьма трудно.

Огромные затраты времени и сил на предварительную подготовку данных вполне ожидаемы: *New York Times* сообщала, что специалисты тратят от 50 до 80 % своего времени на подготовку данных для анализа. Но меня интересовало, существуют ли более эффективные способы очистки и хранения данных, управления ими. В частности, я хотел найти и применять такие способы, чтобы уделять больше времени *анализу данных*. В конце концов, я всегда находил приятным статистический анализ, но только не ручную, подверженную ошибкам подготовку данных в электронных таблицах.

Поскольку мне нравилось писать (спасибо гуманитарному образованию), я начал вести блог, посвященный Excel. Благодаря интересу и упорной работе блог приоб-

[1] Издана в России. — *Ред.*

рел популярность, и именно ему я приписываю большую часть моего профессионального успеха. Приглашаю вас посетить **stringfestanalytics.com**, где я по-прежнему пишу об Excel и общих вопросах анализа.

Как только я начал глубже изучать Excel, мой интерес распространился и на другие аналитические инструменты и технологии. К этому времени большую популярность в мире управления данными приобрели языки программирования с открытым исходным кодом — R и Python. Но в процессе освоения этих языков я испытывал некоторые, совершенно ненужные затруднения.

«Excel — плохо, программирование — хорошо»

Я заметил, что для пользователей Excel мотивация обучения языкам R или Python по большей части выглядит примерно так:

> «Вы долгое время только использовали Excel, хотя должны были заниматься программированием. Посмотрите на все эти проблемы, вызванные использованием Excel! Пора полностью избавиться от них!»

Такая позиция неправильная по нескольким причинам.

Во-первых, это не вполне точно.

> Выбор между программированием и использованием электронных таблиц часто представляется как своего рода борьба между добром и злом. В действительности их лучше рассматривать не как заменяющие, а как дополняющие друг друга инструменты. Электронные таблицы занимают свое место в аналитике, программирование — свое. Изучение и использование одного не исключает использование другого. Их взаимоотношения и взаимосвязи рассматриваются в *главе 5*.

Во-вторых, это в корне неверный методический принцип.

> Пользователи Excel интуитивно понимают, как работать с данными: они умеют их сортировать, фильтровать, группировать и объединять. Они знают, какие механизмы упрощают анализ, а какие требуют дополнительной очистки данных. Это огромный объем знаний, на который следует опираться. Правильное обучение должно служить для устранения разрыва между электронными таблицами и программированием. К сожалению, вместо этого большинство учебных программ с презрением сжигают мост между ними.

Исследования показывают: связывание новых знаний с уже имеющимися дает отличный результат. Так, в книге Питера С. Брауна и соавт. «Запомнить всё. Усвоение знаний без скуки и зубрежки»[2] (Peter C. Brown et al. Make It Stick: The Science of Successful Learning) говорится:

> «Чем лучше вы сможете объяснить, как ваши новые знания соотносятся с уже имеющимися, тем лучше вы их усвоите и тем больше выстроите связей, которые помогут вам в дальнейшем их использовать».

---

[2] Издана в России. — *Ред.*

Пользователю Excel сложно связать новые идеи со своими знаниями, если ему постоянно (и совершенно напрасно) твердят, что все его знания — полная ерунда. В этой книге используется иной подход, основывающийся на уже имеющихся у читателя знаниях об электронных таблицах, чтобы при переходе к языкам R и Python можно было опереться на твердую основу.

И электронные таблицы, и языки программирования являются мощными аналитическими инструментами. Нет необходимости отказываться от Excel только потому, что вы освоили языки R и Python.

Преимущества Excel при обучении аналитике

В действительности Excel — это уникальный инструмент обучения аналитике.

Excel снижает когнитивную нагрузку.

Когнитивная нагрузка — это количество логических связей или переходов, необходимых для понимания чего-либо. Часто учебный курс по аналитике выглядит следующим образом.

1. Изучение новейшей технологии.
2. Изучение способов внедрения новейшей технологии с помощью новейших методов *программирования*.
3. Переход к более прогрессивным технологиям, без уверенных представлений об основах предмета.

Концептуальные основы аналитики довольно сложны. Чтобы понять их при *одновременном* изучении программирования, требуются неимоверные когнитивные затраты. По причинам, которые мы обсудим далее, выполнение анализа данных при помощи программирования имеет большое преимущество. Но овладевать этими наборами навыков лучше по отдельности.

Excel является визуальным калькулятором.

Первое выпущенное на массовый рынок приложение электронных таблиц было названо VisiCalc — буквально: визуальный калькулятор. Это название указывает на один из наиболее важных коммерческих аргументов приложения. Языки программирования, особенно для новичков, напоминают «черный ящик»: напишите магическое слово, нажмите **run** и вуаля — результат готов. Возможно, программа все сделала правильно, но новичку сложно заглянуть внутрь и понять, почему (возможно, еще важнее — почему неправильно).

Excel же позволяет наблюдать, как осуществляется анализ шаг за шагом. Он может наглядно считать и пересчитывать. Вместо того чтобы просто поверить мне (или языкам программирования), вы сами примените ключевые принципы аналитики с помощью Excel.

Excel позволяет изучать основы анализа данных, не прибегая при этом к языку программирования. Это значительно снижает когнитивную нагрузку.

Обзор книги

Теперь, когда вам известен замысел книги и результаты, которых, я надеюсь, вы достигнете, кратко рассмотрим ее структуру.

Часть I. Основы аналитики в Excel.

Аналитика базируется на статистике. Из этой части вы узнаете, как исследовать и проверять взаимосвязи между переменными с помощью Excel. Мы будем использовать Excel, чтобы наглядно продемонстрировать некоторые наиболее важные принципы статистики и анализа. Знания статистической теории и основ анализа, полученные в этой части, обеспечат прочную основу для дальнейшего обучения программированию.

Часть II. От Excel к R.

Когда вы будете свободно владеть основами анализа данных, самое время выбрать один или два языка программирования. Мы начнем с R — языка программирования с открытым исходным кодом, специально разработанного для осуществления статистического анализа. Вы узнаете, как в полном объеме перенести все полученные знания о работе с данными из Excel в R. В заключение вам предстоит выполнить упражнение на языке R.

Часть III. От Excel к Python

Python — еще один язык с открытым исходным кодом, который следует изучить для осуществления эффективного анализа данных. Вы узнаете, как перенести фрагменты данных из Excel в этот язык и выполнить полный анализ данных, подобно тому, как это делалось в *части II*.

Упражнения в конце глав

Когда я читаю книги, то, как правило, пропускаю упражнения в конце главы, поскольку считаю более важным сохранить динамику чтения. *Не делайте так!*

Я предоставляю вам возможность применить на практике изученный материал в конце большинства глав. Решения к этим упражнениям вы найдете в репозитории на GitHub в папке **Exercise-solutions** (Решения для упражнений), которая содержит файлы с именами, совпадающими с названиями соответствующих глав: **https://github.com/stringfestdata/advancing-into-analytics-book**. Выполните все предлагаемые упражнения, а затем сравните свои ответы с решениями из репозитория. Так вы лучше усвоите изученный материал и заодно подадите хороший пример мне.

Наилучший способ обучения — активный: не применив полученные знания на практике, вы их, скорее всего, забудете.

Эта книга — не список готовых решений

В аналитике мне особенно нравится, что почти всегда одну и ту же проблему можно решить несколькими способами. Возможно, я покажу, как сделать что-либо одним способом, тогда как вам знаком другой.

Прежде всего в этой книге я хочу использовать Excel в качестве инструмента для обучения аналитике и помочь читателям совместить полученные знания с R и Python. Если бы я выложил сюда все способы выполнения задач по очистке и анализу данных, потерялась бы главная цель книги.

Вы можете предпочесть альтернативные способы решения той или иной задачи: в иных обстоятельствах действительно возможны лучшие подходы. Однако с учетом целей и задач этой книги я принял решение сфокусироваться на одних технологиях и исключить из поля зрения другие. Иначе книга превратилась бы в простой справочник, а она должна быть руководством по глубокому изучению аналитики.

Без паники!

Надеюсь, я буду для вас добрым и доступным для понимания автором. Однако конкретно для этой книги у меня есть девиз: не паниковать! Бесспорно, вам предстоит научиться сразу многому, поскольку придется погрузиться не только в теорию вероятностей и статистику, но и познакомиться с двумя языками программирования. Книга откроет для вас основы статистики, информатики и многое другое. Сначала это может пугать вас, но со временем вы начнете усваивать знания. Позвольте себе учиться методом проб и ошибок.

Я не сомневаюсь, что с теми знаниями в Excel, которые у вас имеются, поставленная цель вполне достижима в рамках одной книги. Временами вы можете испытывать разочарование и даже «синдром самозванца»[3] — такое случается довольно часто. Не позволяйте этим моментам затмевать реальный прогресс, которого вы достигнете.

Готовы? Тогда увидимся в *главе 1*.

Условные обозначения

- *Курсив:* обозначает новые или особенное важные термины и/или фразы, на которые нужно обратить внимание.
- **Жирный шрифт:** используется для элементов интерфейса, URL-адресов, адресов электронной почты, имен и расширений файлов, а также переменных наборов данных (dataset).

[3] Психологическое явление, когда человек приписывает свои достижения внешним условиям, стечению обстоятельств и т. п., а не собственным качествам, способностям и усилиям. — *Пер.*

- `Моноширинный шрифт`: применяется для листингов, а также для обозначения таких программных элементов, как имена переменных и функций в коде, базы данных, типы данных, переменные окружения, инструкции и ключевые слова.

Этот значок обозначает совет или предложение.

Этот значок обозначает примечание общего характера.

Этот значок обозначает предупреждение или предостережение.

Использование примеров кода

Дополнительные материалы (примеры кода, упражнения и т. д.) доступны для загрузки на GitHub по адресу: **https://github.com/stringfestdata/advancing-into-analytics-book**.

Вы можете загрузить и разархивировать копию нужной папки на вашем компьютере или, если вы знакомы с GitHub, клонировать ее. Этот репозиторий содержит полные копии скриптов и рабочих книг для каждой главы в главной папке. Все наборы данных, необходимые для изучения этой книги, расположены в отдельной подпапке папки наборов данных (datasets) вместе с примечаниями об их источнике и шагах, предпринятых для их сбора и очистки. Я советую вам создавать копии этих рабочих книг Excel, поскольку манипулирование исходными файлами может повлиять на последующие шаги. Все решения к упражнениям в конце глав можно найти в папке **Exercise-solutions**.

В случае возникновения проблем или технических вопросов при использовании примеров кода, пожалуйста, пишите по адресу: **bookquestions@oreilly.com**.

Цель данной книги — помочь вам выполнять свою работу. Как правило, если пример кода представлен в этой книге, вы можете использовать его в своих программах и документации. Если вы не воспроизводите существенную часть кода, то обращаться к O'Reilly за разрешением не нужно. Например, для написания программы с использованием нескольких фрагментов кода из этой книги разрешения не требуется. Однако продажа и распространение примеров из книг издательства O'Reilly требует специального разрешения. Ответ на вопрос с цитированием книги и примером кода разрешения не требует, тогда как включение в документацию к вашему продукту большого количества примеров кода требует разрешения.

Ссылка на источник приветствуется, но не обязательна. Как правило, ссылка на источник включает название, автора, издательство и ISBN. Например: Джордж Маунт. Погружение в аналитику: от Excel к Python и R (O'Reilly). Все права защищены, 2021, Джордж Маунт, 978-1-492-09434-0.

Если вы считаете, что использование примеров кода выходит за рамки разрешения, описанного выше, пожалуйста, обращайтесь по адресу: **permissions@oreilly.com.**

Контакты

У нас есть веб-страница для этой книги, где публикуются исправления, примеры и прочая информация: **https://oreil.ly/advancing-into-analytics**.

Вы можете найти нас на Facebook: **http://facebook.com/oreilly**.

Следите за нами в Twitter: **http://twitter.com/oreillymedia**.

Смотрите нас на YouTube: **http://www.youtube.com/oreillymedia.**

Благодарности

Прежде всего хочу поблагодарить Бога, давшего мне возможность развить мои таланты и делиться ими. Мне было исключительно приятно работать с Мишель Смит и Джоном Хасселом, я всегда буду признателен им за предложение написать эту книгу. Корбин Коллинз оказывал мне всестороннюю поддержку; Дэнни Эльфанбаум и техническая группа превратили рукопись в настоящую книгу; Эйден Джонсон, Феликс Цумштейн и Джордан Голдмейер выполнили неоценимое техническое рецензирование.

Привлечь людей к рецензированию книги непросто, поэтому я должен поблагодарить Джона Денниса, Тобиаса Цвингмана, Джо Балога, Барри Лили, Николь Лагерр и Алекса Бодле за их замечания. Я также хочу поблагодарить сообщества, которые, часто бесплатно, сделали эти знания и технологии доступными. Благодаря изучению аналитики я приобрел несколько фантастических друзей, поделившихся своей мудростью. Мои преподаватели во францисканской средней школе Падуи и колледже Хиллсдейл привили мне любовь к обучению и письму. Вряд ли я написал бы книгу без их влияния.

Я также благодарю моих мать и отца за их любовь и оказанную мне поддержку. И, наконец, моего покойного дедушку: спасибо за то, что научил меня ценить трудолюбие и порядочность.

ЧАСТЬ I

Основы аналитики в Excel

ГЛАВА 1

Основы разведочного анализа данных

«Никогда не знаешь, что войдет в эту дверь», — говорит Рик Харрисон в начале популярного шоу «Звезды ломбарда» (Pawn Stars). То же самое в аналитике: столкнувшись с новым набором данных, вы никогда не знаете, что можете в нем найти. Эта глава посвящена *изучению и описанию* набора данных (dataset), чтобы понять, какие сведения можно из него получить. Этот процесс называется *разведочным анализом данных* (exploratory data analysis, EDA).

Что такое разведочный анализ данных?

Американский математик Джон Тьюки (John Tukey) пропагандировал использование EDA в своей книге «Разведочный анализ данных». Тьюки подчеркивал, что прежде чем перейти к *подтверждению* ответов с помощью проверки гипотез и статистики выводов, сначала нужно изучить данные на предмет потенциальных исследовательских вопросов.

Разведочный анализ данных часто сравнивают с «интервьюированием» данных. Самое время познакомиться с ними и узнать, о каких интересных вещах они могут рассказать. Задачи этой части нашего интервью следующие.

- Классифицировать переменные как непрерывные, категориальные и т. д.
- Обобщить переменные, используя описательную статистику.
- Визуализировать переменные с помощью диаграмм.

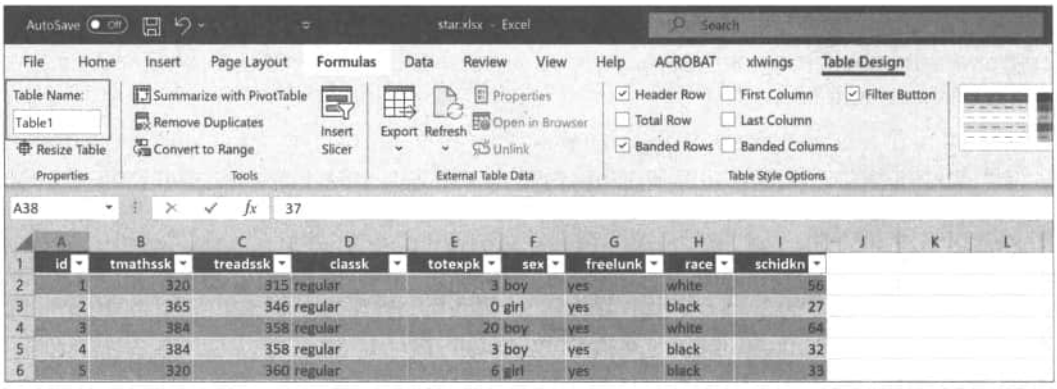
Разведочный анализ данных предоставляет много возможностей. Мы рассмотрим их, используя Excel и набор данных из реальной жизни. Этот набор данных находится в рабочей книге **star.xlsx**, расположенной в подпапке **star** папки **datasets** репозитория на GitHub к данной книге. Набор данных собран для изучения влияния размера класса на результаты тестирования. Для работы с этим примером и прочими демоверсиями на основе Excel вам следует выполнить следующие действия с исходными данными.

1. Создайте копию файла, чтобы не подвергать изменениям исходный набор данных. Позже мы импортируем некоторые из этих Excel-файлов в R или Python, так что любые изменения в исходных наборах данных отразятся на этих процессах.
2. Добавьте индексный столбец с именем **id**. В нем каждая строка набора данных будет нумероваться таким образом, что первая строка будет иметь ID, равный 1,

вторая — 2 и т. д. Это можно быстро сделать в Excel, если ввести цифры в несколько первых строк столбца, затем выделить данный диапазон и завершить выделение по этому шаблону с помощью функции **Flash Fill** (Мгновенное заполнение). Найдите маленький квадрат в нижнем правом углу активной ячейки, наведите на него курсор и удерживайте, пока не увидите маленький знак «плюс», затем заполните остальную часть диапазона. Добавление индексного столбца упростит анализ данных по группам.

3. Наконец, преобразуйте полученный набор данных в таблицу, выбрав любую ячейку в диапазоне и нажав последовательно **Insert** (Вставить) | **Table** (Таблицу). Для Windows сочетание клавиш <Ctrl>+<T>, для Mac — <Cmd>+<T>. Если у вашей таблицы есть заголовки, убедитесь в том, что установлен флажок **My table has headers** (Таблица с заголовками). У таблиц имеется множество преимуществ, одно из которых — их эстетическая привлекательность. Кроме того, в операциях с таблицами удобно ссылаться на столбцы по имени.

Вы можете присвоить таблице имя по своему желанию, щелкнув мышкой в любом месте внутри нее, затем выбрав последовательно на ленте **Table Design** (Конструктор таблиц) | **Table Name** (Имя таблицы) в группе **Properties** (Свойства), как показано на рис. 1.1.



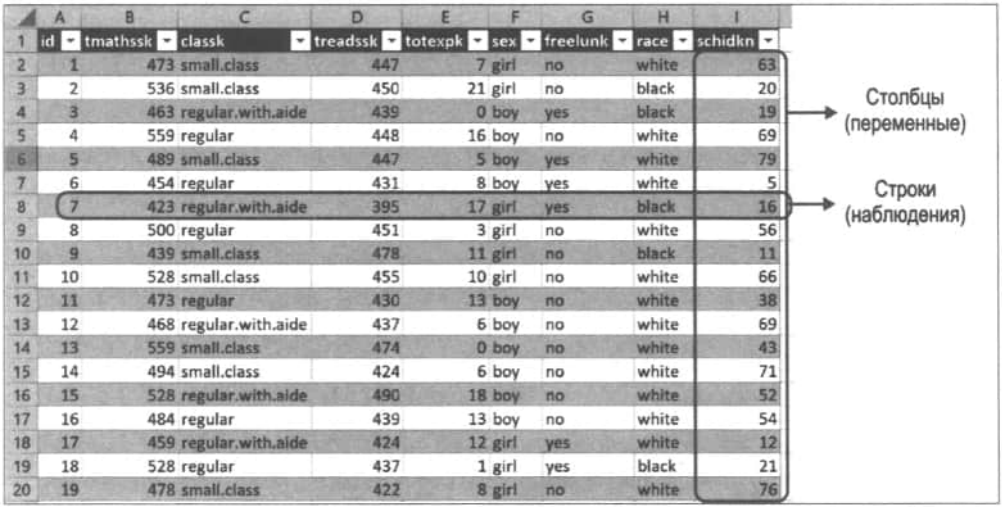
	A	B	C	D	E	F	G	H	I	J	K	L
1	id	tmathssk	treadssk	classk	totexpk	sex	freelunk	race	schidkn			
2	1	320	315	regular	3	boy	yes	white	56			
3	2	365	346	regular	0	girl	yes	black	27			
4	3	384	358	regular	20	boy	yes	white	64			
5	4	384	358	regular	3	boy	yes	black	32			
6	5	320	360	regular	6	girl	yes	black	33			

Рис. 1.1. Поле **Table Name**

Выполнение этих нескольких аналитических задач послужит хорошей практикой для работы с другими наборами данных в Excel. Для набора данных **star** заполненная таблица должна выглядеть так, как показано на рис. 1.2. Я назвал таблицу **star**. Этот набор данных представляет собой прямоугольник из столбцов и строк.

Вероятно, работая с достаточно большими объемами данных, вы поняли, что такая форма очень удобна для анализа. Иногда необходимо очистить данные, чтобы привести их в нужное состояние. Далее мы обсудим некоторые из операций по очистке данных, а сейчас остановимся на самих данных и разведочном анализе данных.

В аналитике мы часто оперируем *наблюдениями* и *переменными*, а не *строками* и *столбцами*. Рассмотрим значения этих терминов.



	A	B	C	D	E	F	G	H	I
1	id	tmathssk	classk	treadssk	totexpk	sex	freelunk	race	schidkn
2	1	473	small.class	447	7	girl	no	white	63
3	2	536	small.class	450	21	girl	no	black	20
4	3	463	regular.with.aide	439	0	boy	yes	black	19
5	4	559	regular	448	16	boy	no	white	69
6	5	489	small.class	447	5	boy	yes	white	79
7	6	454	regular	431	8	boy	yes	white	5
8	7	423	regular.with.aide	395	17	girl	yes	black	16
9	8	500	regular	451	3	girl	no	white	56
10	9	439	small.class	478	11	girl	no	black	11
11	10	528	small.class	455	10	girl	no	white	66
12	11	473	regular	430	13	boy	no	white	38
13	12	468	regular.with.aide	437	6	boy	no	white	69
14	13	559	small.class	474	0	boy	no	white	43
15	14	494	small.class	424	6	boy	no	white	71
16	15	528	regular.with.aide	490	18	boy	no	white	52
17	16	484	regular	439	13	boy	no	white	54
18	17	459	regular.with.aide	424	12	girl	yes	white	12
19	18	528	regular	437	1	girl	yes	black	21
20	19	478	small.class	422	8	girl	no	white	76

Рис. 1.2. Набор данных **star** в виде строк и столбцов

Наблюдения

В нашем наборе данных мы имеем 5748 строк: каждая представляет собой уникальное наблюдение. В данном случае проводятся измерения данных студентов; объектом наблюдения может быть что угодно — от физических лиц до целых народов.

Переменные

Каждый столбец содержит отдельную порцию данных о наблюдениях. Например, в наборе данных **star** можно найти баллы по чтению каждого студента (**treadssk**), а также тип класса (**classk**) студента. Обозначим эти столбцы как *переменные*. В табл. 1.1 указано, что́ измеряет каждый столбец.

Таблица 1.1. *Описание переменных набора данных* ***star***

Столбец	Описание
id	Уникальный идентификатор / индексный столбец
tmathssk	Общий приведенный балл по математике
treadssk	Общий приведенный балл по чтению
classk	Тип класса
totexpk	Общий стаж работы преподавателя
sex	Пол
freelunk	Право на бесплатный обед
race	Раса
schidkn	Индикатор школы

Готовы к тавтологии? Мы называем переменные *переменными*, поскольку их значения могут варьироваться в зависимости от наблюдений. Если бы каждое записанное наблюдение возвращало одни и те же измерения, анализировать было бы нечего. Каждая переменная может предоставить совершенно различную информацию о наблюдениях. Даже в этом относительно небольшом наборе данных в качестве переменных у нас есть текст, числа и утверждения да/нет. Некоторые наборы данных могут иметь десятки или даже сотни переменных.

Типы переменных полезно классифицировать, поскольку различия в них будут иметь большое значение, когда мы продолжим анализ. Помните, что эти различия носят довольно произвольный характер и могут изменяться в зависимости от целей и обстоятельств анализа. Вы убедитесь в том, что разведочный анализ данных, как и аналитика в целом, является высокоитеративным.

Классификация переменных является довольно приблизительной и, как почти всё в аналитике, основана не на жестких критериях, а на эмпирических закономерностях.

Мы рассмотрим различные *типы переменных*, показанные на рис. 1.3, а затем на этой основе классифицируем набор данных **star**.

Рис. 1.3. Типы переменных

Существуют и другие типы данных; например, мы не будем рассматривать разницу между интервальными и относительными данными. Вы можете ближе познакомиться с типами переменных в книге Сары Бослаф «Статистика для всех»[1] (Sarah Boslaugh. Statistics in a Nutshel). Давайте пройдемся по рис. 1.3, двигаясь слева направо.

Категориальные переменные

Иногда эти переменные, называемые также *качественными*, описывают качество или характеристику наблюдения. Типичный вопрос, на который они отвечают: «Какого типа?» Категориальные переменные часто, хотя и не всегда, представляются нечисловыми значениями.

[1] Издана в России. — *Ред.*

Пример категориальной переменной — страна происхождения. Как и любая другая переменная, она может принимать разные значения (США, Финляндия и т. д.), но выполнить качественное сравнение их между собой мы не можем (может ли кто-нибудь сказать, что такое «дважды Индонезия»?). Любое уникальное значение, которое принимает качественная переменная, называется уровнем этой переменной. Например, США, Финляндия или Индонезия могут быть тремя уровнями страны происхождения.

Поскольку качественные переменные описывают качество наблюдения, а не его количество, многие количественные операции над этими данными невозможны. Например, нельзя вычислить *среднее* значение страны происхождения, но можно рассчитать *наиболее распространенную,* или общую, частотность каждого уровня.

Категориальные величины также можно различать в зависимости от количества их уровней и целесообразности ранжирования этих уровней.

Двоичные переменные могут иметь только два уровня. Часто, но не всегда, эти переменные устанавливаются как ответы «да» или «нет». Вот несколько примеров двоичных переменных.

- Женат/замужем? (да или нет).
- Заказ сделан? (да или нет).
- Тип вина? (красное или белое).

В случае с типом вина подразумевается, что нас интересуют только данные по красному или белому вину... но если мы также захотим проанализировать розовое вино? В этом случае уже нельзя включить все три уровня и анализировать данные как двоичные.

Любая качественная переменная, имеющая более двух уровней, является *номинальной.* Вот некоторые примеры.

- Страна происхождения (США, Финляндия, Индонезия и т. д.).
- Любимый цвет (оранжевый, синий, жженая сиена и т. д.).
- Тип вина (красное, белое, розовое).

Обратите внимание: идентификационный номер в данном случае является категориальной переменной, заданной численно; хотя *можно* получить среднее значение идентификационного номера, это число не имеет смысла. Важно отметить, что *истинного упорядочения* номинальных переменных не существует. Например, красный как цвет не может быть явно расположен по уровню выше или ниже синего. Поскольку естественное упорядочение не вполне понятно, рассмотрим некоторые примеры его использования.

Порядковые переменные имеют более трех уровней, между которыми существует естественное упорядочение. Вот некоторые примеры порядковых переменных.

- Размер напитка (маленький, средний, большой).
- Курс (первый, второй, третий и т. д.).
- Дни недели (понедельник, вторник, среда, четверг, пятница).

Здесь *можно* однозначно упорядочить уровни: студент четвертого курса выше, чем первокурсник, тогда как сказать то же самое о красном и синем цвете нельзя. Хотя эти уровни можно *ранжировать*, не всегда удается количественно определить *разницу* между ними. Например, разница по размеру между маленьким и средним напитком может быть не такой, как между средним и большим.

Количественные переменные

Количественные переменные описывают измеримую величину наблюдения. Типичный вопрос, на который отвечают количественные переменные: «Сколько?» («Как много?»). Эти переменные почти всегда представлены числами. Также можно различать количественные переменные с учетом количества значений, которые они могут принимать.

Наблюдения *непрерывной переменной* теоретически могут принимать бесконечное число значений между любыми двумя другими значениями. Звучит несколько запутанно, но в мире природы непрерывные переменные довольно распространены. Вот некоторые примеры:

- рост (в диапазоне от 59 до 75 дюймов; наблюдение может быть 59,1; 74,99 или любое иное значение из этого диапазона);
- уровень pH;
- площадь поверхности.

Поскольку можно проводить количественные сравнения наблюдений за непрерывными переменными, к ним применим более полный анализ. Например, брать среднее значение *непрерывных переменных* целесообразно, тогда как для *категориальных* это не имеет смысла. Далее в этой главе вы узнаете, как анализировать непрерывные переменные путем поиска их описательных статистических данных в Excel.

Вместе с тем наблюдения *дискретной переменной* могут принимать только фиксированное число исчисляемых значений в промежутке между двумя величинами. Дискретные переменные весьма распространены в социальных науках и бизнесе. Вот некоторые примеры:

- количество человек в семье (в диапазоне от 1 до 10 наблюдение может принимать значение 2 или 5, но не 4,3);
- количество проданных единиц товара;
- количество деревьев в лесу.

Часто, имея дело с дискретными переменными, имеющими много уровней, или много наблюдений, мы рассматриваем их как непрерывные переменные для обеспечения более полного объема статистического анализа. Например, вы, возможно, слышали, что средняя семья в США имеет 1,93 ребенка. Понятно, что *в действительности* ни в одной семье нет такого количества детей. Это *дискретная переменная*, которая выражается целыми числами. Однако во многих наблюдениях это утверждение может быть полезным, чтобы определить, сколько детей ожидается в типичной семье.

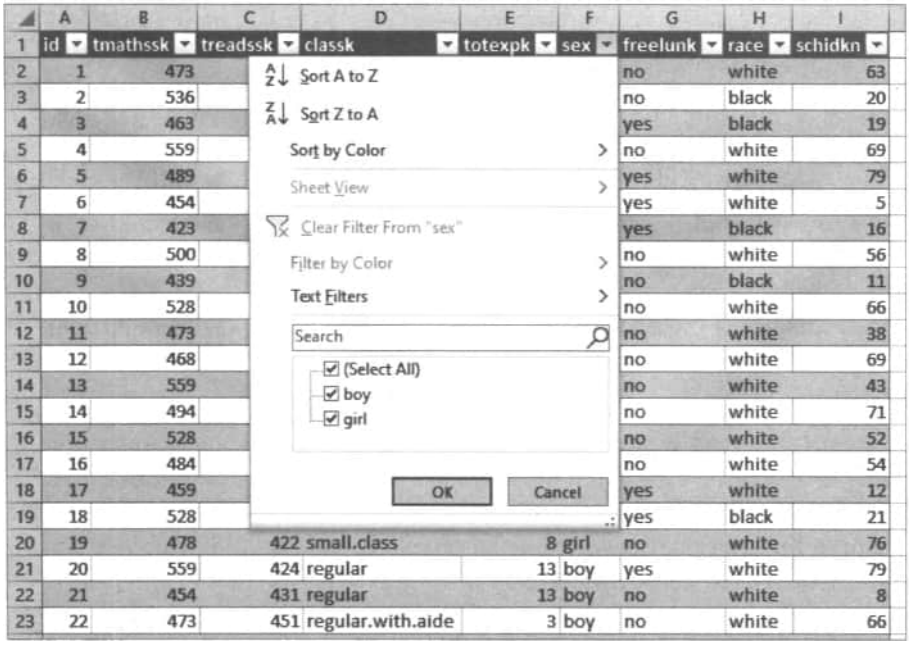
Но это еще не всё! В более углубленной аналитике часто приходится пересчитывать и смешивать переменные: например, можно выполнить *логарифмическое преобразование* одной переменной так, чтобы она удовлетворяла условиям анализа. Или выделить значение множества переменных в меньшее их количество с помощью метода, называемого *снижением размерности*. Однако данные методы выходят за рамки этой книги, и мы не будем на них останавливаться.

Закрепление материала: классификация переменных

Используя полученные знания, классифицируйте переменные набора **star** по типам, показанным на рис. 1.3. Обдумывая это задание, внимательно изучите данные. Я покажу, как сделать это простым способом, а с более тщательным процессом мы познакомимся чуть позже.

Один из простых способов понять, какие могут быть типы переменных, — найти количество уникальных значений, которые они могут принимать. В Excel это можно сделать с помощью фильтра. Я щелкнул на стрелке раскрывающегося списка рядом с переменной **sex** (рис. 1.4) и обнаружил, что она может принимать только два различных значения. Какой это может быть тип переменной, по вашему мнению? Просмотрите переменные, используя этот или иной способ.

В табл. 1.2 показано, как я классифицировал переменные.



	A	B	C	D	E	F	G	H	I
1	id	tmathssk	treadssk	classk	totexpk	sex	freelunk	race	schidkn
2	1	473					no	white	63
3	2	536					no	black	20
4	3	463					yes	black	19
5	4	559					no	white	69
6	5	489					yes	white	79
7	6	454					yes	white	5
8	7	423					yes	black	16
9	8	500					no	white	56
10	9	439					no	black	11
11	10	528					no	white	66
12	11	473					no	white	38
13	12	468					no	white	69
14	13	559					no	white	43
15	14	494					no	white	71
16	15	528					no	white	52
17	16	484					no	white	54
18	17	459					yes	white	12
19	18	528					yes	black	21
20	19	478	422	small.class	8	girl	no	white	76
21	20	559	424	regular	13	boy	yes	white	79
22	21	454	431	regular	13	boy	no	white	8
23	22	473	451	regular.with.aide	3	boy	no	white	66

Рис. 1.4. Использование фильтра для определения возможного количества различных значений переменной

Таблица 1.2. Классификация переменных

Переменная	Описание	Категориальная или количественная	Тип
id	Индексный столбец	Категориальная	Номинальная
tmathssk	Общий приведенный балл по математике	Количественная	Непрерывная
treadssk	Общий приведенный балл по чтению	Количественная	Непрерывная
classk	Тип класса	Категориальная	Номинальная
totexpk	Общий стаж работы преподавателя	Количественная	Дискретная
sex	Пол	Категориальная	Двоичная
freelunk	Право на бесплатный обед	Категориальная	Двоичная
race	Раса	Категориальная	Номинальная
schidkn	Индикатор школы	Категориальная	Номинальная

Некоторые из этих переменных, такие как **classk** (тип класса) и **freelunk** (право на бесплатный обед), было проще классифицировать. Другие, такие как **schidkn** (индикатор школы) и **id** (индексный столбец, или идентификатор), не были столь очевидными: они выражаются числовыми значениями, но их нельзя сравнить количественно.

Представление данных в числовом виде не означает, что они могут использоваться как количественные переменные.

Вы увидите, что только три из переменных являются количественными: **tmathssk** (общий приведенный балл по математике), **treadssk** (общий приведенный балл по чтению) и **totexpk** (общий стаж работы преподавателя) Я классифицировал первые две как непрерывные, и последнюю — как дискретную. Чтобы понять, почему именно так, рассмотрим сначала **totexpk**. Все наблюдения здесь выражаются целыми числами, начиная от 0 до 27. Поскольку эта переменная может принимать только фиксированное количество исчисляемых значений, я классифицировал ее как дискретную.

Но как насчет *tmathssk* и *treadssk* — экзаменационных баллов? Они также выражаются целыми числами, т. е. студент не может получить балл по чтению 528,5 балла, а только 528 или 529. Поэтому они являются *дискретными*. Однако, поскольку эти оценки могут принимать так много уникальных значений, на практике имеет смысл классифицировать их как непрерывные.

Возможно, вы удивитесь, что в такой строгой области, как аналитика, существует очень мало жестких правил.

Резюме: типы переменных

Изучай правила, чтобы знать, как правильно их нарушать.
Далай-лама XIV

Способ классификации переменной влияет на то, как мы можем обработать ее в анализе: например, вычислить среднее значение непрерывных переменных можно, тогда как номинальных переменных — нельзя. В то же время мы часто нарушаем правила исходя из целесообразности: например, берем среднее значение дискретной переменной, так что семья имеет в среднем 1,93 ребенка.

Продвигаясь дальше в изучении анализа, мы можем решить изменить и другие правила, переклассифицировать переменные или создать совершенно новые. Помните, что разведочный анализ данных (EDA) — это итеративный процесс.

Работа с данными и переменными — это итеративный процесс. Способ классификации переменных может меняться в зависимости от того, что мы обнаружим в ходе исследования, и от вопросов, ответы на которые нужно получить из имеющихся данных.

Исследование переменных в Excel

Продолжим исследование набора данных **star** с помощью *описательной статистики* и *визуализаций*. Мы будем выполнять анализ в Excel, хотя вы можете выполнять те же самые шаги в языках R или Python и получить те же результаты. К концу книги вы сможете выполнять разведочный анализ данных, используя все три метода.

Начнем исследование с категориальных переменных набора данных **star**.

Исследование категориальных переменных

Помните, что с помощью категориальных переменных мы измеряем *качество*, а не *количество*, поэтому, например, у них не будет среднего, минимума или максимума, имеющих какой-либо смысл. Тем не менее можно выполнить определенный анализ этих данных, в частности, посредством подсчета *частот*. В Excel это можно сделать с помощью *сводных таблиц* (PivotTables). Поместите курсор в любом месте набора данных **star** и выполните последовательно **Insert** (Вставить) | **PivotTable** (Сводная таблица), как показано на рис. 1.5, а затем нажмите кнопку **OK**.

Я хочу выяснить, сколько наблюдений можно получить для каждого типа класса. Для этого я перетащу переменную **classk** в область **Rows** (Строки) сводной таблицы, а переменную **id** — в область **Values** (Значения). По умолчанию Excel вычислит сумму по полю **id**, ошибочно предположив, что категориальная переменная является количественной. Мы не можем количественно сравнить идентификационные номера, но мы можем рассчитать их частоты. Для этого в Windows нажмите кнопку **Sum of id** (Сумма идентификаторов) в области **Values** и выберите **Value field settings** (Параметры полей значений). На вкладке **Summarize value field by** (Сум-

мировать поля значений с помощью[2]) выберите **Count** (Рассчитать) и нажмите кнопку **ОК**. На Mac для этого щелкните на значке **i** рядом с **Sum of id** (Сумма id). Теперь мы имеем то, что нужно: количество наблюдений для каждого типа класса. Такая таблица, показанная на рис. 1.6, известна как *одномерная таблица частот*.

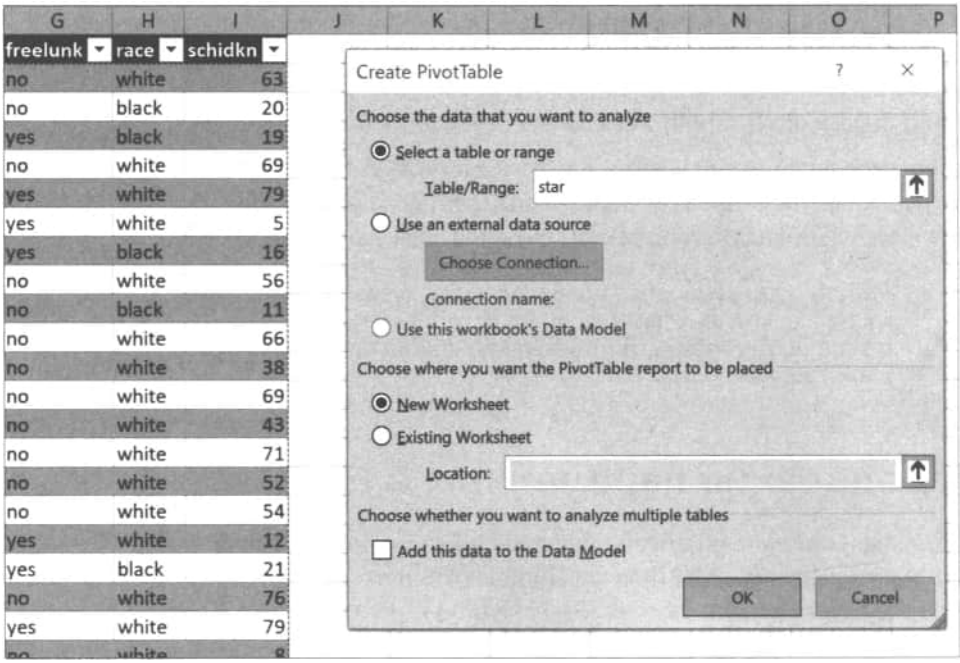


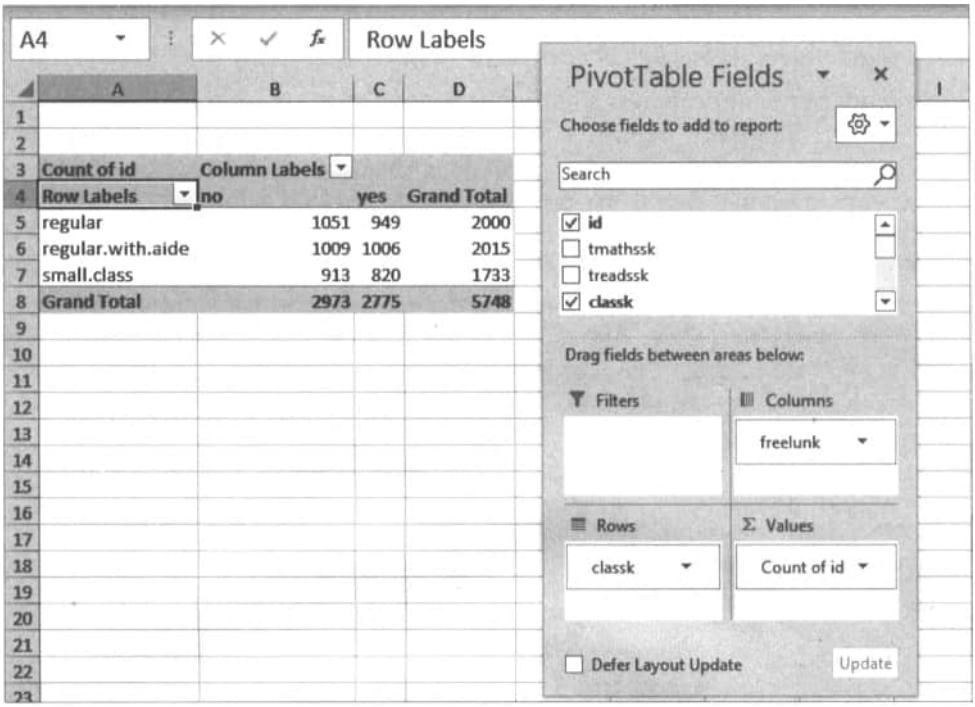
Рис. 1.5. Вставка сводной таблицы

	A	B
1		
2		
3	**Row Labels**	**Count of id**
4	regular	2000
5	regular.with.aide	2015
6	small.class	1733
7	**Grand Total**	**5748**
8		

Рис. 1.6. Одномерная таблица частот для типов класса

Давайте разобьем этот подсчет частотности на наблюдения за учащимися, которые имеют и не имеют право на бесплатный обед. Для этого поместите переменную **freelunk** в область **Columns** (Столбцы) сводной таблицы. Теперь мы имеем *двумерную таблицу частот*, как показано на рис. 1.7.

[2] В русском интерфейсе вкладка называется «Операции». — *Ред.*



Count of id	Column Labels		
Row Labels	no	yes	Grand Total
regular	1051	949	2000
regular.with.aide	1009	1006	2015
small.class	913	820	1733
Grand Total	2973	2775	5748

Рис. 1.7. Двумерная таблица частот

На протяжении всей книги мы будем создавать визуализации как часть анализа. С учетом всех прочих вопросов, которые нам предстоит охватить, мы не будем тратить много времени на принципы и методы визуализации данных. Однако вам стоит ознакомиться с этой областью знаний. Полезную информацию вы найдете в книге Клауса О. Уилке «Основы визуализации данных»[3] (Claus O. Wilke. Fundamentals of Data Visualization).

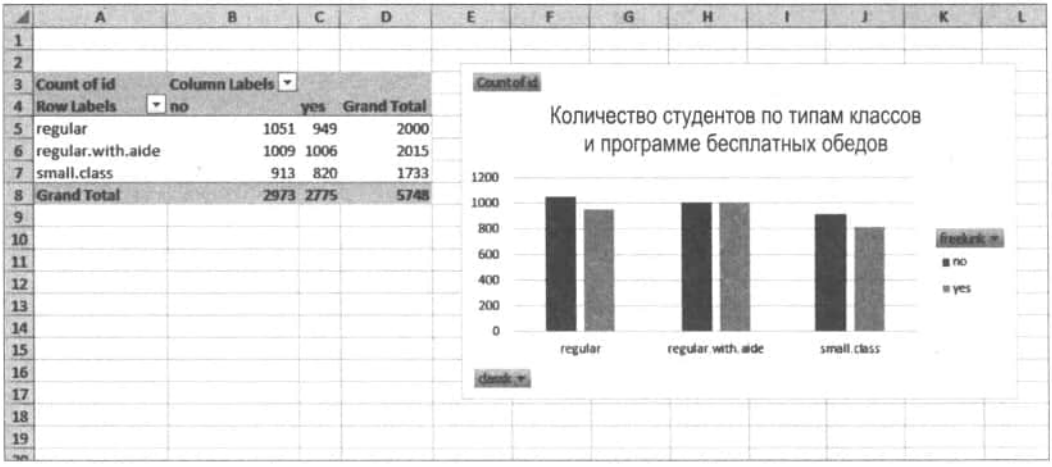
Визуализировать одномерную или двумерную таблицу можно с помощью гистограммы (также известной как «*барплот*» (barplot), или «*каунтплот*» (countplot)). Давайте построим двумерную таблицу частот. Для этого щелкните внутри сводной таблицы и затем последовательно нажмите **Insert** (Вставить) | **Clustered Column** | (Гистограмму с группировкой). Результат показан на рис. 1.8. Я добавлю заголовок к диаграмме, щелкнув на ее периметре, а затем на значке «плюс» (+), который появится в правом верхнем углу. В появившемся меню **Chart elements** (Элементы диаграммы) установите флажок **Chart title** (Заголовок диаграммы). Чтобы найти это меню на Mac, щелкните на диаграмме и выберите последовательно в верхнем меню **Design** (Дизайн) | **Add Chart Element** (Добавить элемент диаграммы). Таким образом мы еще несколько раз будем добавлять диаграммы на протяжении книги.

Обратите внимание, что и на графике, и в таблице количество наблюдений разделено по типам классов на студентов, участвующих в программе бесплатных обедов

[3] Издана в России. — *Ред.*

и не участвующих. Например, 1051 и 949 обозначают первую и вторую метки и столбцы в таблице и на графике соответственно.

Визуализация результата полезна даже для такого простого анализа, как двумерная таблица частот. Человеку значительно проще обработать линии и столбцы на графике, чем числа в таблице, поэтому по мере возрастания сложности анализа следует продолжать графически отображать его результаты.



Count of id	Column Labels		
Row Labels	no	yes	Grand Total
regular	1051	949	2000
regular.with.aide	1009	1006	2015
small.class	913	820	1733
Grand Total	2973	2775	5748

Рис. 1.8. Двумерная таблица частот, визуализированная в виде «каунтплота»

Мы не можем выполнить количественное сравнение категориальных данных, поэтому любой анализ, который мы будем производить, будет основываться на количестве их появлений (встречаемости). Это может показаться малоинтересным, но это важно: так мы сможем узнать, какие уровни величин наиболее распространены, и, возможно, нам понадобится сравнить эти уровни по другим переменным для дальнейшего анализа. А пока мы рассмотрим количественные переменные.

Исследование количественных переменных

Теперь проведем более полное исследование *сводной*, или *описательной*, статистики. *Описательная статистика* позволяет получить представление о наборе данных с использованием количественных методов. Частоты — это один из видов описательной статистики; мы рассмотрим и некоторые другие виды, а также узнаем, как рассчитать их в Excel.

Меры центральной тенденции — это один из наборов описательной статистики, выражающий значение или значения, которые принимает типичное наблюдение. Мы рассмотрим три наиболее распространенные меры.

1. Среднее значение: в частности, *среднее арифметическое* значение рассчитывается путем сложения всех наблюдений и деления этого числа на общее количество наблюдений. Из всех рассмотренных статистических показателей вы, возможно, знакомы с ним лучше всего, мы и далее будем с ним встречаться.

2. Медиана: наблюдение, находящееся *в середине* набора данных. Чтобы рассчитать медиану, отсортируйте или упорядочьте данные от низшего к высшему, затем подсчитайте данные с обеих сторон, чтобы найти середину. Если в середине найдено две величины, рассчитайте среднее, чтобы найти медиану.
3. Мода: наиболее часто встречающееся значение. Чтобы найти моду, нужно также отсортировать данные. Переменная может иметь одну, несколько или ни одной моды.

В Excel есть богатый набор статистических функций, в том числе для вычисления мер центральной тенденции, как показано в табл. 1.3.

Таблица 1.3. *Функции Excel для измерения центральной тенденции*

Статистическая величина	**Функция Excel**
Среднее значение	`AVERAGE(number1, [number2], ...)`
Медиана	`MEDIAN(number1, [number2], ...)`
Мода	`MODE.MULT(number1, [number2], ...)`

`MODE.MULT()` — это новая функция в Excel, которая использует средства динамических массивов, чтобы возвращать несколько возможных мод. Если у вас нет доступа к этой функции, попробуйте `MODE()`. С помощью этих функций можно найти меры центральной тенденции для **tmathssk**. Результат показан на рис. 1.9.

В результате этого анализа мы видим, что три меры (показателя) центральной тенденции имеют схожие значения, т. е. среднее арифметическое равно 485,6, медиана — 484 и мода — 489. Я также определил, как часто встречается мода: 277 раз.

	J	K	L	M
1		Центральная тенденция tmathssk		
2		Среднее значение	485.6480515	=AVERAGE(star[tmathssk])
3		Медиана	484	=MEDIAN(star[tmathssk])
4		Мода	489	=MODE.MULT(star[tmathssk])
5		Мода — сколько?	277	=COUNTIF(star[tmathssk],L4)
6				

Рис. 1.9. Расчет мер средней тенденции в Excel

На какой из этих мер центральной тенденции следует сосредоточиться? Отвечу на этот вопрос с помощью небольшого практического примера. Представьте, что вы консультируете некоммерческую организацию. Вас попросили просмотреть пожертвования и посоветовать, какой показатель центральной тенденции стоит отслеживать. Пожертвования показаны в табл. 1.4. Вам нужно посчитать и принять решение.

Таблица 1.4. *Данные для отслеживания показателя (меры) центральной тенденции*

Пожертвования, $				
10	10	25	40	120

Наиболее подходящим кажется среднее значение. Но *действительно* ли 41 доллар правильно характеризует имеющиеся данные? Все частные пожертвования, кроме одного, в 120 долларов, меньше этого значения. Это один из недостатков среднего значения: экстремальные величины значительно влияют на них.

Такой проблемы можно избежать с помощью медианы: значение в 25 долларов, возможно, является лучшим представлением «средней величины», чем 41 доллар. Недостаток этой меры в том, что она не учитывает точное значение каждого наблюдения: мы просто «отсчитываем» середину переменной, не учитывая относительную величину каждого наблюдения.

Таким образом, остается мода, которая предоставляет полезную информацию: наиболее *частое* пожертвование — 10 долларов. Однако 10 долларов не характеризуют пожертвования в целом. Как мы уже говорили, набор данных может иметь несколько мод или не иметь ни одной, поэтому ее нельзя считать стабильной мерой.

Что же мы ответим некоммерческой организации? Они должны отслеживать и все эти показатели. Каждая мера оценивает данные с разных точек зрения. Тем не менее в последующих главах вы увидите, что при проведении более сложного статистического анализа особое внимание обычно обращается на среднее значение.

Чтобы получить более полное представление об одном и том же наборе данных, мы часто будем анализировать несколько статистических характеристик. Необязательно одна мера лучше, чем другие.

Теперь, когда мы установили, где находится «центр» переменной, давайте посмотрим, как распределяются ее значения от центра. Существует несколько *характеристик изменчивости*; рассмотрим самые распространенные.

Прежде всего, это *диапазон*, или разница между максимальным и минимальным значениями. Несмотря на простоту получения, данный показатель очень чувствителен к наблюдениям: достаточно одного экстремального значения, и диапазон может ввести в заблуждение относительно того, где именно фактически находится большинство наблюдений.

Далее, *дисперсия* — показатель разброса наблюдений вокруг средней величины. Его расчет является более трудоемким, чем то, что мы рассматривали до сих пор. Чтобы рассчитать дисперсию, необходимо выполнить следующие шаги:

1. Найти среднее значение набора данных.
2. Вычесть среднее значение из каждого наблюдения (это *отклонение*).
3. Рассчитать сумму квадратов всех отклонений.
4. Разделить полученную сумму квадратов на количество наблюдений.

Это длинный путь. Для выполнения всех этих операций лучше использовать математическую запись. Поначалу это пугает, но к этому надо привыкнуть — сравните с приведенным списком в качестве альтернативы. Что более удобно для понимания? Математическая запись может предоставить более точный способ выражения

того, что нужно сделать. Например, с помощью уравнения 1.1 можно выполнить все шаги, необходимые для нахождения дисперсии.

Уравнение 1.1. *Формула для расчета дисперсии*

$$s^2 = \frac{\sum\left(X - \bar{X}\right)^2}{N}$$

s^2 — это дисперсия. Выражение $\left(X - \bar{X}\right)^2$ говорит о том, что необходимо вычесть среднее значение $\bar{X}$ из каждого наблюдения X и затем возвести результат в квадрат. $\sum$ говорит о необходимости просуммировать эти результаты. Наконец, полученный результат делится на количество наблюдений N.

Я и далее буду использовать математическую нотацию в этой книге, но только в тех случаях, когда этот способ более эффективен для выражения и понимания данной концепции, чем обычное изложение всех шагов. Попробуйте рассчитать дисперсию чисел, приведенных в табл. 1.5.

Таблица 1.5. *Данные для расчета дисперсии*

Число					
3	5	2	6	3	2

Поскольку такая статистика относительно более сложная, для проведения вычислений я буду использовать Excel. Скоро вы узнаете, как рассчитать дисперсию с помощью встроенных функций Excel. Результаты представлены на рис. 1.10.

	A	B	C	D
1	наблюдения	среднее значение	отклонение	квадрат отклонения
2	3	3.5	-0.5	0.25
3	5	3.5	1.5	2.25
4	2	3.5	-1.5	2.25
5	6	3.5	2.5	6.25
6	3	3.5	-0.5	0.25
7	2	3.5	-1.5	2.25
8				
9	сумма квадратов отклонений	13.5	=SUM(D2:D7)	
10	число наблюдений	6	=COUNT(A2:A7)	
11	дисперсия	2.25	=B9/B10	

Рис. 1.10. Вычисление дисперсии в Excel

Результаты можно найти на листе **variability** рабочей книги **ch-1.xlsx**, прилагаемой к этой главе (см. репозиторий на GitHub).

Вы можете спросить, почему мы используем *квадрат отклонения*. Чтобы понять это, возьмем сумму неквадратичных отклонений. Получим ноль: эти отклонения взаимоисключают друг друга.

Проблема дисперсии в том, что теперь мы имеем дело с *квадратами отклонений* исходной величины. Такой способ анализа данных не является интуитивным. Что-

бы это исправить, мы возьмем квадратный корень из дисперсии, известный как *стандартное отклонение*. Теперь изменчивость выражается в терминах исходной единицы измерения среднего значения. Уравнение 1.2 показывает стандартное отклонение, выраженное математически.

Уравнение 1.2. *Формула для расчета стандартного отклонения*

$$s = \sqrt{\frac{\sum(X_i - \bar{X})^2}{N}}$$

Используя эту формулу, получим стандартное отклонение, как на рис. 1.10, равное 1,5 (квадратный корень от 2,25). Можно рассчитать эти характеристики изменчивости в Excel, используя функции, приведенные в табл. 1.6. Следует отметить, что для *выборочной* дисперсии используются функции, отличные от тех, которые применяются для *генеральной* дисперсии и стандартного отклонения. Для выборочных измерений в знаменателе используется $N - 1$, а не N, что дает большее значение дисперсии и стандартного отклонения.

Таблица 1.6. *Функции Excel для измерения изменчивости*

Статистическая величина	Функция Excel
Диапазон	MAX(number1, [number2], ...)_ - _MIN(number1, [num ber2], ...)
Дисперсия (выборочная)	VAR.S(number1, [number2], ...)
Стандартное отклонение (выборочное)	STDEV.S(number1, [number2], ...)
Дисперсия (генеральная)	VAR.P(number1, [number2], ...)
Стандартное отклонение (генеральное)	STDEV.P(number1, [number2], ...)

Различие между выборкой и совокупностью будет в центре внимания в последующих главах. На данный момент, если вы не уверены, что собрали все интересующие вас данные, воспользуйтесь *примерами* функций. Как вы начинаете понимать, имеется *несколько* описательных статистик, на которые следует обратить внимание. Можно ускорить их вычисление с помощью функций Excel, но также можно использовать его набор инструментов анализа данных для получения полного набора описательной статистики несколькими щелчками мыши.

Некоторые статистические показатели различаются при расчете для совокупных значений и выборки. Если вы не уверены, с чем имеете дело, рассматривайте это как выборку.

Хотя эта надстройка (add-in) включена в Excel, вам необходимо установить ее в первую очередь. В Windows в верхней панели выберите последовательно **File** (Файл) | **Options** (Параметры) | **Add-ins** (Надстройки). Затем нажмите кнопку **Go** (Начать) в нижней части меню. Выберите в меню **Analysis ToolPak** (Пакет инстру-

ментов анализа) и нажмите кнопку **OK**. Выбирать элемент **Analysis ToolPak — VBA** (Пакет инструментов анализа — VBA) необязательно. На Mac выберите в строке меню **Data** (Данные) | **Analysis Tools** (Инструменты анализа). Выберите в меню **Analysis ToolPak** (Пакет инструментов анализа), затем нажмите кнопку **OK**. Возможно, вам потребуется перезапустить Excel для завершения установки. После этого вы увидите новую кнопку **Data Analysis** (Анализ данных) во вкладке **Data**.

В табл. 1.1 мы определили, что **tmathssk** и **treadssk** являются непрерывными переменными. Давайте рассчитаем их описательную статистику с помощью этого инструментария (ToolPak). В верхнем меню выберите **Data | Data Analysis | Descriptive Statistics** (Описательная статистика). Отобразится меню, в котором следует выбрать входной диапазон `B1:C5749`. Не забудьте установить флажки для элементов **Labels in first row** (Метки в первой строке) и **Summary statistics** (Итоговая статистика). Меню должно выглядеть так, как показано на рис. 1.11. Остальные настройки можно не менять; нажмите кнопку **OK**.

В результате на новом листе будет вставлена описательная статистика для этих двух переменных, как показано на рис. 1.12.

Теперь для сравнения между группами давайте рассмотрим поиск описательной статистики для каждого уровня категориальной переменной. Для этого вставьте на новый лист новую сводную таблицу (PivotTable), используя данные набора **star**. Поместите переменную **freelunk** в область **Columns** (Столбцы), переменную **id** в область **Rows** (Строки) и **Sum of treadssk** — в область **Values** (Значения). Помните, что поле **id** является уникальным идентификатором, поэтому суммировать его не следует.

Рис. 1.11. Получение описательной статистики с помощью **Analysis ToolPak**

	A	B	C	D
1	*tmathssk*		*treadssk*	
2				
3	Среднее значение	485.6480515	Среднее значение	436.7423452
4	Стандартная ошибка	0.63010189	Стандартная ошибка	0.419080917
5	Медиана	484	Медиана	433
6	Мода	489	Мода	437
7	Стандартное отклонение	47.77153121	Стандартное отклонение	31.77285677
8	Выборочная дисперсия	2282.119194	Выборочная дисперсия	1009.514427
9	Коэффициент эксцесса (куртозис)	0.289321748	Коэффициент эксцесса (куртозис)	3.83779705
10	Асимметрия	0.473937363	Асимметрия	1.340898831
11	Диапазон	306	Диапазон	312
12	Минимум	320	Минимум	315
13	Максимум	626	Максимум	627
14	Сумма	2791505	Сумма	2510395
15	Счет	5748	Счет	5748
16				

Рис. 1.12. Описательная статистика, полученная с помощью **Analysis ToolPak**

В этой и всех прочих сводных таблицах, с которыми мы будем работать, лучше отключить все общие итоги, щелкнув внутри таблицы и выбрав последовательно **Design** (Конструктор) | **Grand totals** (Общие итоги) | **Off** (Отключить) — для строк и столбцов. Таким образом мы не будем ошибочно включать общие итоги в анализ. Теперь можно использовать данный пакет инструментов для добавления описательной статистики. Результат показан на рис. 1.13.

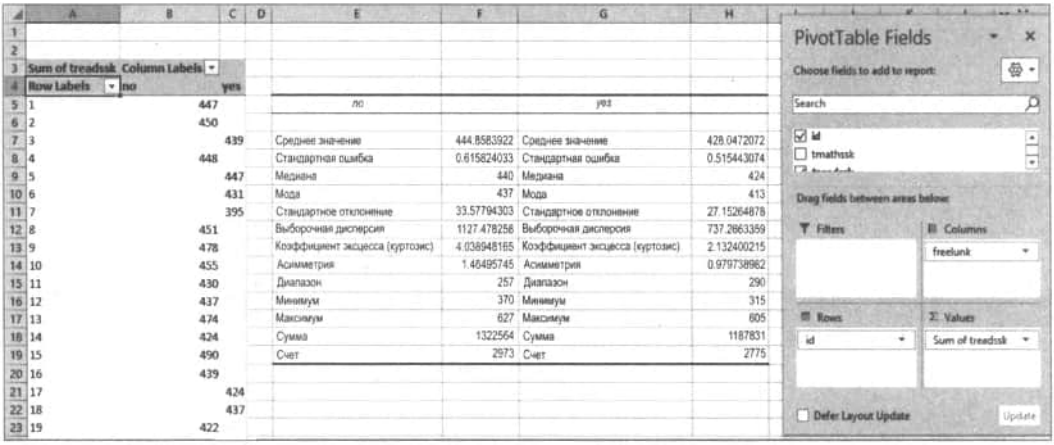


Рис. 1.13. Расчет описательной статистики по группам

Итак, вы уже знакомы с большинством измерений, остальные будут рассмотрены позже в этой книге. Может показаться, что вся информация, представленная в данном пакете инструментов, сводит на нет необходимость визуализации данных. В действительности визуализация по-прежнему является незаменимой в разведочном анализе данных. В частности, мы будем использовать ее для получения сведений о *распределении* наблюдений по всему диапазону значений переменной.

Прежде всего, посмотрим на гистограммы. С помощью этих графиков можно визуализировать относительную частоту наблюдений по интервалам. Чтобы построить гистограмму переменной **treadssk** в Excel, выберите этот диапазон данных, затем в верхнем меню выберите последовательно **Insert** (Вставить) | **Histogram** (Гистограмма). Результат показан на рис. 1.14.

На рис. 1.14 видно, что наиболее часто встречающийся интервал находится между значениями 426,6 и 432,8, и в этот диапазон попадает примерно 650 наблюдений. Ни одна из тестовых оценок в нашем примере не содержит десятичных дробей, но ось **x** может их включать, в зависимости от того, как Excel устанавливает интервалы (столбики гистограммы). Можно изменить количество столбиков гистограммы, щелкнув правой кнопкой мыши на оси **x** графика и выбрав элемент **Format Axis** (Формат оси). Справа отобразится меню (эти настройки недоступны на Mac).

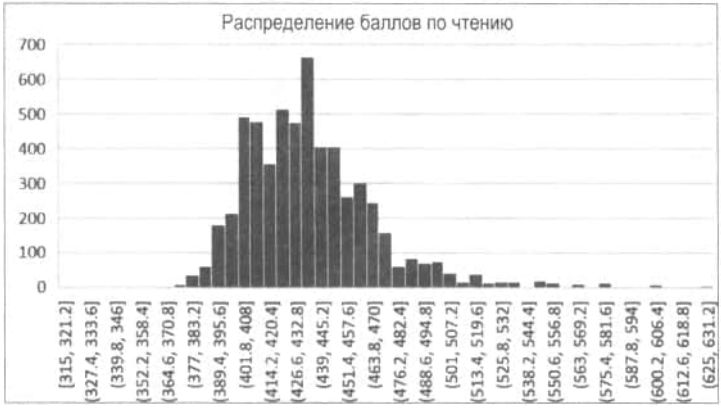


Рис. 1.14. Распределение баллов по чтению

По умолчанию Excel определил 51 столбик гистограммы. Но что, если мы (приблизительно) возьмем вдвое меньше или вдвое больше этого количества, до 25 и 100 соответственно? Настройте нужное количество в меню и посмотрите результат (рис. 1.15). Я называю это «приближением и удалением» деталей распределения.

При визуализации распределения посредством гистограммы видно, что значительное количество тестовых баллов сосредоточено в крайней правой части, но подавляющее большинство находится в диапазоне 400–500.

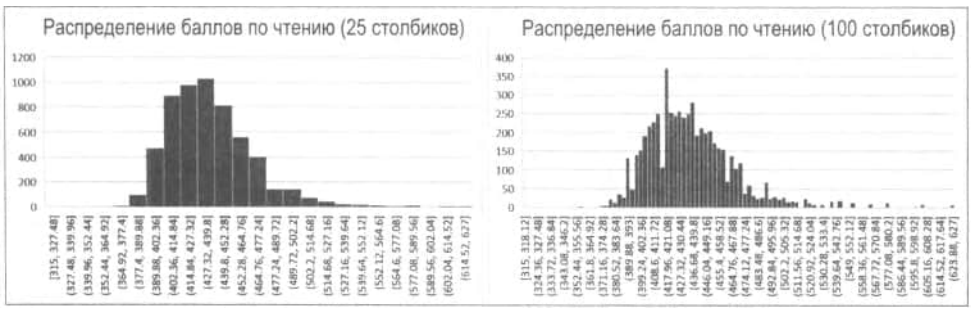
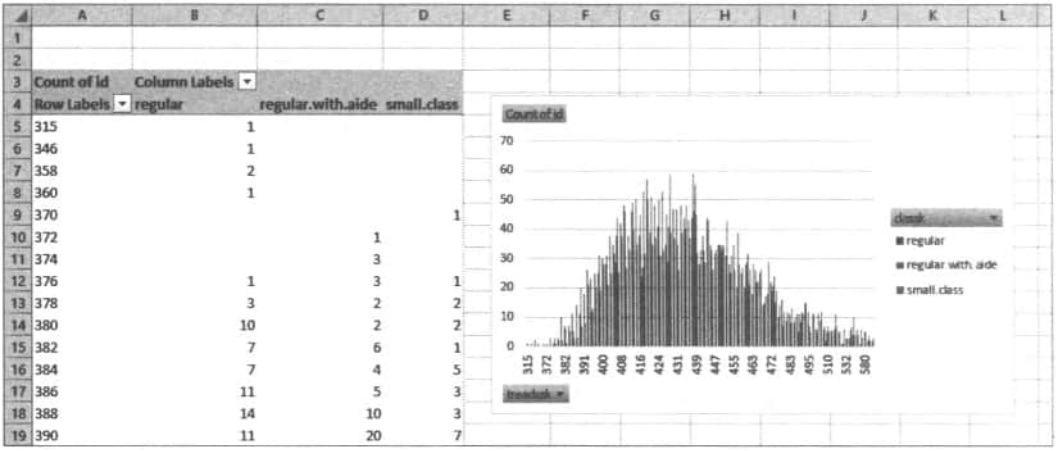


Рис. 1.15. Изменение количества столбиков гистограммы

Предположим, нам нужно посмотреть, как распределение баллов по чтению изменяется во всех трех классах. Здесь мы сравниваем непрерывную переменную по трем уровням категориальной переменной. Чтобы выполнить это с помощью гистограммы в Excel, потребуется небольшой «взлом», но для получения результата мы можем использовать сводные таблицы.

Вставьте новую сводную таблицу для набора данных **star**, затем перетащите **treadssk** в область **Rows** (Строки), **classk** — в область **Columns** (столбцы) и **Count of id** — в область **Values** (Значения). Напомню, последующий анализ будет проще, если уберать общие итоги из сводной таблицы.

Теперь построим по этим данным график. Щелкните в любом месте сводной таблицы и в верхнем меню выберите **Insert** (Вставить) | **Clustered Column** (Гистограмма с группировкой). Результат показан на рис. 1.16, и его очень сложно прочесть. Но если сравнить этот результат с исходной сводной таблицей, видно, что из студентов с баллом 380 10 посещали занятия в обычном классе (regular), 2 — занятия в обычном классе с ассистентом (regular with aide) и 2 — занятия в небольших классах (small class).



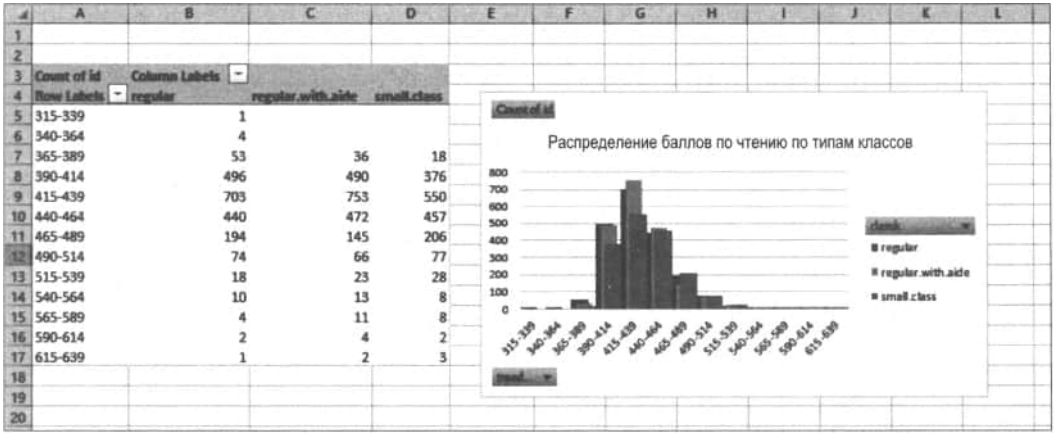
Count of id	Column Labels		
Row Labels	regular	regular.with.aide	small.class
315	1		
346	1		
358	2		
360	1		
370			1
372		1	
374		3	
376	1	3	1
378	3	2	2
380	10	2	2
382	7	6	1
384	7	4	5
386	11	5	3
388	14	10	3
390	11	20	7

Рис. 1.16. Гистограмма с несколькими группами

Теперь нужно собрать эти значения в более крупные интервалы. Для этого щелкните правой кнопкой мыши в любом месте значений первого столбца сводной таблицы и выберите **Group** (Группировать). Excel по умолчанию выполнит группировку с шагом 100; поменяйте его на 25.

Гистограмма становится распознаваемой. Переформатируем график так, чтобы он стал еще более целостным. Щелкните правой кнопкой мыши на любом столбике гистограммы и выберите **Format Data Series** (Формат ряда данных). Установите для **Series Overlap** (Перекрытие рядов) значение 75 % и для **Gap Width** (Ширина зазора) — значение 0 %. Результат показан на рис. 1.17.

Можно установить ширину зазора до полного пересечения, но тогда будет еще сложнее увидеть распределение для стандартного класса. Гистограммы — это один



Count of id	Column Labels		
Row Labels	regular	regular.with.aide	small.class
315-339	1		
340-364	4		
365-389	53	36	18
390-414	496	490	376
415-439	703	753	550
440-464	440	472	457
465-489	194	145	206
490-514	74	66	77
515-539	18	23	28
540-564	10	13	8
565-589	4	11	8
590-614	2	4	2
615-639	1	2	3

Рис. 1.17. Гистограмма с несколькими группами, созданная при помощи сводной таблицы

из лучших способов визуализации распределения непрерывной переменной, но они могут быть и чрезмерно громоздкими.

В качестве альтернативы давайте рассмотрим блочную диаграмму, или «боксплот» (boxplot). В ней распределение визуализируется с использованием *квартилей*. Центром блочной диаграммы является показатель, с которым вы уже знакомы, — *медиана*.

Можно представить медиану, являющуюся «серединой» набора данных, в качестве второго квартиля. Найти первый и третий квартили можно, равномерно разделив набор данных на квадранты и найдя их середины. На рис. 1.18 обозначены различные элементы блочной диаграммы («боксплота»).

Рис. 1.18. Элементы блочной диаграммы

Часть результирующего графика, находящаяся внутри «коробки», называется *межквартильным размахом*. Этот диапазон используется как основа для получения других частей графика. Остальной диапазон, который в 1,5 раза превышает межквартильный размах, представлен двумя линиями или «усами». Этот тип диаграммы в Excel известен как «ящик с усами» (Box & Whisker).

Наблюдения, не попавшие в этот диапазон, отображаются на графике отдельными точками и называются *выбросами*. Построение блочной диаграммы может оказаться более сложным, но справиться с ней нам поможет Excel. Вернемся к нашему примеру с **treadssk**. Выделите нужный диапазон и выберите в верхнем меню **Insert** (Вставить) | **Box & Whisker** (Ящик с усами).

На рис. 1.19 видно, что межквартильный диапазон (размах) находится в пределах от 415 до 450, а имеется несколько выбросов, особенно в верхней части. Аналогичные закономерности можно было увидеть и в данных гистограммы, хотя там полное распределение было представлено более наглядно, и мы могли исследовать его на разных уровнях детализации с разными длинами интервала. Так же как и описательная статистика, каждая визуализация предлагает уникальное представление данных, и ни одна из них не превосходит другие.

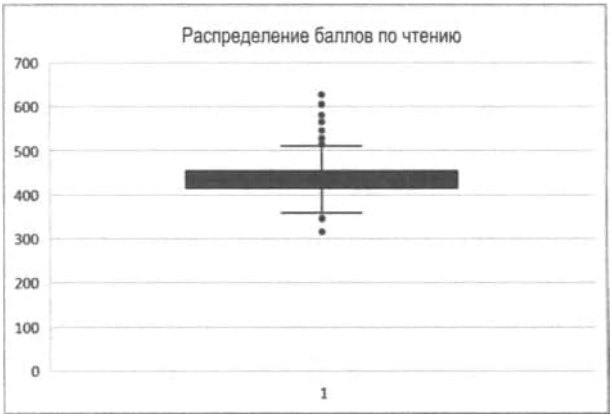


Рис. 1.19. Блочная диаграмма баллов по чтению

Одним из преимуществ блочной диаграммы является то, что она дает точную информацию о том, где находятся квартили данных и какие наблюдения считаются выбросами. Другое преимущество заключается в том, что сравнение нескольких групп упрощается. Чтобы построить блочные диаграммы нескольких групп в Excel, проще всего иметь искомую категориальную переменную слева от непрерывной переменной. Для этого переместите переменную **classk** влево от переменной **treadssk** в исходных данных. Выделите эти данные и выберите в верхнем меню **Insert** | **Box & Whisker**. На рис. 1.20 видно, что общее распределение баллов за чтение выглядит одинаково во всех трех группах.

Напомним, что с количественными данными можно делать значительно больше, чем подсчитывать частоты, а именно:

- определить, вокруг каких значений концентрируются данные, используя показатели центральной тенденции;
- определить, насколько разбросаны эти данные, используя характеристики изменчивости;
- визуализировать распределение данных с помощью гистограмм и блочных диаграмм.

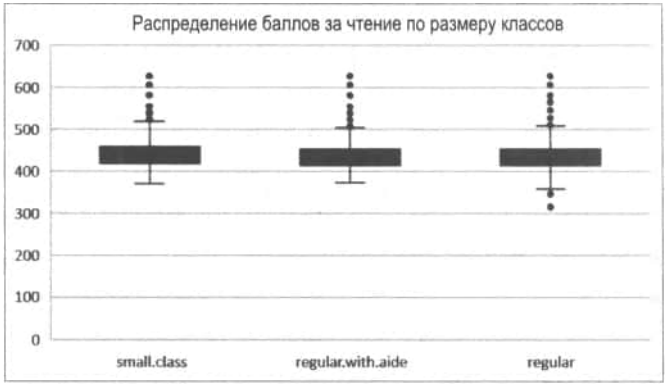


Рис. 1.20. Блочная диаграмма баллов за чтение по размеру (типу) классов

Существуют и другие описательные статистики и визуализации для изучения количественных переменных. О некоторых из них вы узнаете позже из этой книги. Но ответы на наиболее важные вопросы вы найдете в процессе изучения разведочного анализа данных (EDA).

Заключение

Хотя мы никогда не знаем, что́ именно получим в новом наборе данных, разведочный анализ дает возможность в этом разобраться. Из этой главы мы узнали, с каким видом переменных мы имеем дело в наборе данных **star** и как в целом выглядят и ведут себя наблюдения, являющиеся довольно глубокими. Эту работу мы продолжим в *главе 3:* изучим подтверждение выводов, полученных о данных при исследовании. Но сначала, в *главе 2*, познакомимся с понятием *вероятности*, которая дает значительную часть сведений для аналитического процесса.

Упражнения

Используйте для закрепления знаний о разведочном анализе данных набор housing, расположенный в репозитории на GitHub: **datasets** | **housing** (жилье) | **housing.xlsx**. Это набор данных из реальной жизни, включающий цены на жилье в городе Виндзор, Онтарио, Канада. Описание переменных находится на листе **readme** указанного файла. Выполните следующие задания и проведите собственный разведочный анализ данных.

1. Классифицируйте тип каждой переменной.
2. Постройте двумерную таблицу частот **airco** (кондиционеры) и **prefarea** (предпочитаемый район проживания).
3. Получите описательную статистику для переменной **price** (цена).
4. Визуализируйте распределение переменной **lotsize** (размер участка).

Решения этих и всех других упражнений из книги вы можете найти в папке **exercise-solutions** репозитория. Для каждой главы есть файл с соответствующим именем.

ГЛАВА 2

Понятие вероятности

Задумывались ли вы когда-нибудь о том, что́ в действительности подразумевают метеорологи, говоря о вероятности дождя, равной 30 %? Несмотря на свои прогнозы, они не могут сказать точно, будет ли дождь, т. е. они *не уверены в результате*. Они могут лишь выразить неопределенность количественно как величину между 0 % (дождя наверняка не будет) и 100 % (наверняка пойдет дождь).

У аналитиков данных, как и у метеорологов, нет магического шара. Зачастую, обладая всего лишь данными выборки, мы пытаемся сделать вывод о совокупности значений. Таким образом, нам тоже требуется количественно оценивать неопределенность как вероятность.

В этой главе мы будем изучать, как работает вероятность и как выводятся вероятности. Также мы будем использовать Excel для моделирования наиболее важных статистических положений, в значительной степени основанных на вероятности. Это даст вам прочную основу для перехода к следующим двум главам, в которых мы будем заниматься инференциальной статистикой в Excel.

Вероятность и случайность

В разговорной речи мы говорим «что-то происходит случайно» тогда, когда это кажется вырванным из контекста или беспорядочным. В вероятности событие считается *случайным*, если известно, что оно *будет иметь результат*, но неизвестно, какой именно.

Например, возьмем шестигранную игральную кость. Когда мы ее подбрасываем, то знаем, что она упадет на одну грань — не исчезнет и не упадет на несколько граней. В статистике под случайностью подразумевается знание, что результат *будет получен*, но неизвестно, *какой* именно.

Вероятность и выборочное пространство

Известно, что когда игральная кость упадет, мы увидим номер от 1 до 6. Этот набор, состоящий из всех результатов, называется *выборочным пространством*. Каждому из этих результатов присвоена вероятность, отличная от нуля, т. к. кость может упасть на любую из граней. В сумме эти вероятности равны 1, поскольку очевидно, что результатом будет одна из возможностей, относящихся к выборочному пространству.

Вероятность и эксперименты

Мы выяснили, что падение игральной кости происходит случайно, а также наметили его *выборочное пространство*. Теперь можно приступить к построению экспериментов для этого случайного события. С точки зрения вероятности эксперименты — это процедуры, которые могут воспроизводиться бесконечное число раз с последовательной выборкой возможных результатов.

Некоторые эксперименты требуют многих лет планирования, но наш, к счастью, прост — бросание кости. При каждом броске мы получаем значения от 1 до 6. Результат — это наши выходные данные. Каждый бросок кости — *пробный* эксперимент.

Безусловная и условная вероятность

С учетом того, что мы уже знаем о вероятности, типичный вероятностный вопрос о бросках кости может быть таким: «Какова вероятность того, что выпадет четверка?» Это называется *маргинальной* или *безусловной вероятностью*, т. к. рассматривается одно событие в отдельности.

А если задаться вопросом: «Какова вероятность выпадения двойки с учетом того, что в последнем испытании выпала единица?» Чтобы ответить на этот вопрос, необходимо обратиться к *совместной вероятности*. Иногда, рассматривая вероятность двух событий, мы знаем результат одного, но не знаем результат другого. Такой вариант известен как *условная вероятность*, одним из способов расчета которой является формула (правило) Байеса.

Мы не будем подробно рассматривать правило Байеса, так же как и множество областей теории вероятности и статистики, которые его применяют, но в будущем его изучение будет полезным. Прочтите книгу Уилла Курта «Байесовская статистика»[1] («Bayesian Statistics the Fun Way») — это будет захватывающим введением в данную тему. Вы увидите, что «байесианство» предлагает уникальный подход к работе с данными с помощью специальных аналитических приложений.

Научные направления, развившиеся на основе правила Байеса, расходятся с так называемыми частотными подходами, используемыми в этой книге, и многими классическими статистическими теориями.

Распределение вероятностей

Итак, мы узнали, почему бросание кубика является случайным экспериментом, и обозначили выборочное пространство возможных значений, которые наше испытание может принять. Известно, что сумма вероятностей каждого результата должна равняться 1, но какова относительная вероятность каждого результата? Чтобы

[1] Издана в России. — *Ред.*

узнать это, необходимо обратиться к распределению вероятностей. Распределение вероятностей — это список возможных результатов, которые может иметь эксперимент, а также распространенность каждого результата. Хотя распределение вероятностей может быть записано в виде формальной математической функции, мы сосредоточимся на ее количественном выходе.

В *главе 1* мы говорили о разнице между дискретными и непрерывными переменными. Также существуют соответствующие дискретные и непрерывные распределения вероятностей. Познакомимся поближе с первым из них.

Дискретное распределение вероятностей

Продолжим наш пример с подбрасыванием кости. Это считается *дискретным распределением вероятностей*, поскольку имеется счетное число результатов: например, хотя результатом броска кости может быть 2 или 3, но никогда — 2,25.

По сути, подбрасывание игральной кости представляет собой *дискретное равномерное* распределение вероятностей, потому что каждый результат равновероятен для любого испытания: т. е. вероятность того, что выпадет 4, такая же, как и относительно 2 и т. д. Точнее: вероятность каждого результата — один к шести.

Чтобы ознакомиться с этим и другими примерами в Excel, используемыми в этой главе, обратитесь к файлу **ch-2.xlsx** в репозитории на GitHub. Для большинства примеров я уже выполнил некоторые подготовительные действия на рабочем листе, остальное мы проделаем вместе с вами. Начнем с листа **uniform-distribution**. Каждый возможный результат X указан в диапазоне `A2:A7`. Как мы знаем, вероятность получения любого результата одинакова, следовательно, формула для `B2:B7` должна быть = 1/6. $P(X = x)$ — это означает вероятность того, что данное событие приведет к указанному результату.

Теперь выберите диапазон `A1:B7` и в верхнем меню нажмите последовательно **Insert** (Вставить) | **Clustered Column** (Гистограмма с группировкой). Распределение вероятности и визуализация должны выглядеть так, как показано на рис. 2.1.

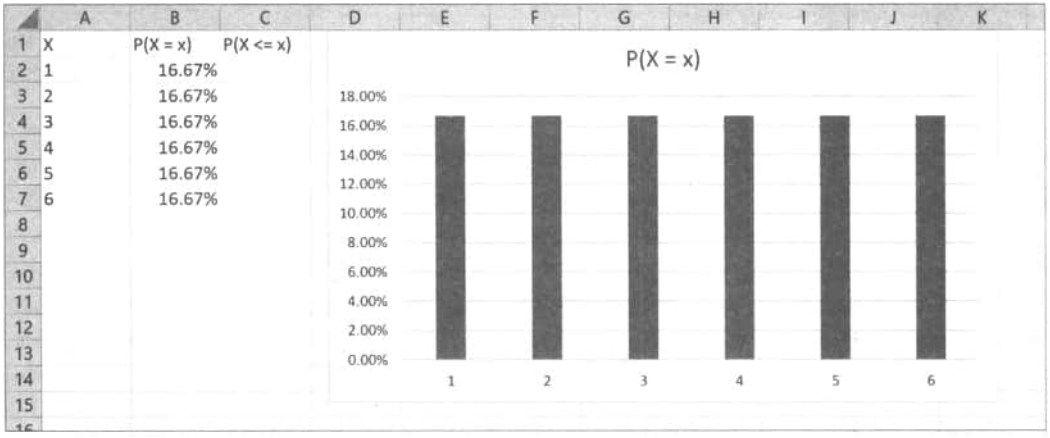
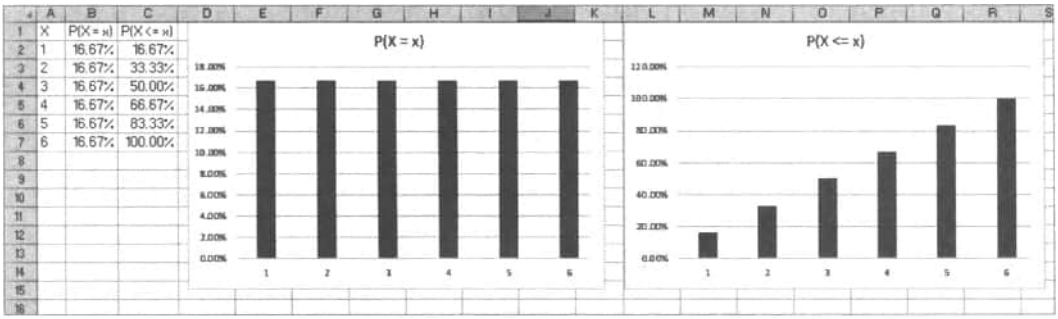


Рис. 2.1. Распределение вероятностей подбрасывания шестигранной игральной кости

Добро пожаловать в ваше первое распределение вероятности! Заметили пробелы между значениями в визуализации? Это явно указывает на дискретные, а не непрерывные результаты.

Иногда необходимо знать *кумулятивную вероятность* результата. В таком случае мы берем промежуточную сумму всех вероятностей, пока не достигнем 100 % (поскольку выборочное пространство в сумме должно равняться 1). Найдем вероятность того, что событие будет меньше или равно заданному результату в столбце C. С помощью формулы `=SUM($B$2:B2)` мы можем подсчитать промежуточную сумму в диапазоне `C2:C7`.

Теперь выделите диапазон `A1:A7` и, удерживая клавишу <Ctrl> в ОС Windows или клавишу <Cmd> на Mac, выделите `C1:C7`. Затем, выбрав этот несмежный диапазон, создайте вторую гистограмму с группировкой. Вы видите разницу между распределением вероятности и кумулятивным распределением вероятности на рис. 2.2?



	A	B	C
1	X	P(X = x)	P(X <= x)
2	1	16.67%	16.67%
3	2	16.67%	33.33%
4	3	16.67%	50.00%
5	4	16.67%	66.67%
6	5	16.67%	83.33%
7	6	16.67%	100.00%

Рис. 2.2. Сравнение вероятности и кумулятивного распределения вероятностей подбрасывания шестигранной игральной кости

Основываясь на логических и математических рассуждениях, мы предполагали, что вероятность выпадения любой стороны игральной кости составляет 1:6. Это можно назвать *теоретической вероятностью*. Также можно было найти распределение вероятности эмпирически, подбрасывая кубик много раз и записывая результаты. Это называется *экспериментальной вероятностью*. Наконец, с помощью экспериментов можно обнаружить, что вероятность выпадения каждой стороны игральной кости в действительности *не равна* 1:6, как предполагалось теоретически, и что преобладает одна сторона игральной кости.

Существует несколько способов получения экспериментальных вероятностей: во-первых, можно провести реальный эксперимент. Конечно, весьма утомительно бросать кости десятки раз и записывать результаты. Вместо этого мы можем заставить компьютер выполнить тяжелую работу и *смоделировать* эксперимент. Моделирование часто обеспечивает достаточно точное приближение к реальности и используется, когда проведение экспериментов в реальных условиях слишком сложно или требует много времени. Недостаток моделирования в том, что оно может не отразить все аномалии или особенности реального эксперимента.

Моделирование часто используется в аналитике с целью выяснения того, что может произойти в реальной жизни, когда прибегать к экспериментам слишком сложно или даже невозможно.

Чтобы смоделировать эксперимент с бросанием игральной кости, нужен способ с последовательным выбором числа от 1 до 6 в случайном порядке. Это можно сделать с помощью имеющегося в Excel генератора случайных чисел `RANDBETWEEN()`. Результаты, которые вы видите в книге, будут отличаться от тех, что вы получите в своих попытках... но *все* они будут случайными числами от 1 до 6.

Результаты, полученные вами с помощью генератора случайных чисел Excel, будут выглядеть иначе, чем те, что представлены в этой книге.

Откройте рабочий лист **experimental-probability**. В столбце `A` отмечены 100 испытаний подбрасывания игральной кости, для которых мы хотим записать результаты. Можно, конечно, начать бросать настоящую игральную кость и записывать результаты в столбец `B`. Но более эффективная, хотя и менее реалистичная альтернатива, — моделировать результаты с помощью функции `RANDBETWEEN()`.

Данная функция принимает два аргумента:

```
RANDBETWEEN(bottom, top)
```

Мы используем шестигранный кубик, поэтому наш диапазон находится в пределах от 1 до 6:

```
RANDBETWEEN(1, 6)
```

Функция `RANDBETWEEN()` возвращает только целые числа — именно то, что нам нужно в данном случае (напомню, это *дискретное* распределение). Используя маркер заполнения, можно сгенерировать результат для всех 100 испытаний. Не стоит слишком привязываться к текущим результатам: нажмите <F9> в Windows, <Fn>+<F9> на Mac или выберите в верхнем меню **Formulas** (Формулы) | **Calculate Now** (Рассчитать сейчас). Таким образом, данные вашей рабочей книги будут пересчитаны и случайные числа повторно сгенерированы.

Давайте сравним наши теоретические и экспериментальные вероятности бросания игральной кости в столбцах `D-F`. Столбец `D` будет использоваться для нумерации выборочного пространства: номера от 1 до 6. В столбце `E` поместите теоретическое распределение: 1/6, или 16,67 %. В столбце `F` рассчитайте экспериментальное распределение из столбцов `A` и `B`. Это процент случаев, когда каждый результат находится во всех испытаниях. Его можно рассчитать по формуле

```
=COUNTIF($B$2:$B$101, D2)/COUNT($A$2:$A$101)
```

Выделите диапазон `D1:F7` и в верхнем меню последовательно нажмите **Insert** (Вставить) | **Clustered Column** (Гистограмма с группировкой). Ваш рабочий лист должен выглядеть так, как показано на рис. 2.3. Попробуйте пересчитать результаты несколько раз.

	A	B	C	D	E	F
1	Испытание	Результат		X	Теоретическая	Экспериментальная
2	1	1		1	16.67%	19.00%
3	2	3		2	16.67%	15.00%
4	3	5		3	16.67%	15.00%
5	4	5		4	16.67%	20.00%
6	5	3		5	16.67%	18.00%
7	6	4		6	16.67%	13.00%
8	7	6				
9	8	6				
10	9	4				
11	10	1				
12	11	3				
13	12	1				
14	13	1				
15	14	6				
16	15	4				
17	16	4				

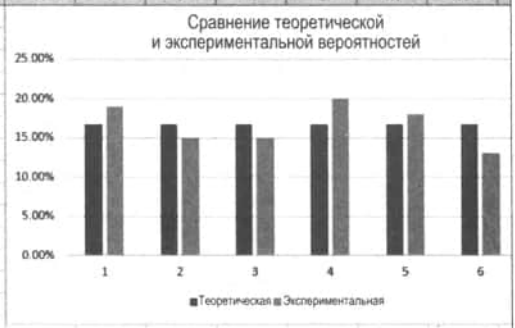


Рис. 2.3. Теоретическая и экспериментальная вероятности подбрасывания игральной кости

Похоже, мы не ошиблись, предсказав, основываясь на экспериментальном распределении, равную вероятность выпадения каждого числа. Разумеется, экспериментальное распределение не полностью соответствует теоретическому: случайность всегда содержит какую-то ошибку.

Тем не менее, возможно, если бы эксперимент проводился в реальных условиях, результаты отличались бы от тех, что мы получили при моделировании. Возможно, реальный процент выпадений не является правильным, и мы упустили это из виду, полагаясь на собственные рассуждения и алгоритм Excel. Звучит тривиально, но часто вероятности в реальной жизни ведут себя не так, как мы (или наши компьютеры) ожидаем.

Дискретное равномерное распределение — одно из множества дискретных распределений вероятности; также к часто используемым в аналитике относятся *биномиальное* и *пуассоновское* распределения.

Непрерывное распределение вероятностей

Распределение вероятностей считается непрерывным, если результат может принимать любое возможное значение между двумя другими. Мы сосредоточимся на *нормальном распределении*, или *колоколообразной кривой* (рис. 2.4). Возможно, вам знакома эта известная форма.

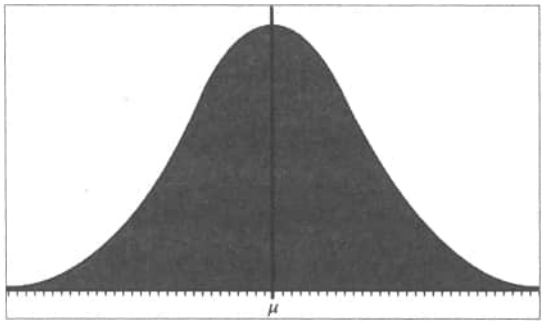


Рис. 2.4. Гистограмма нормального распределения

На гистограмме представлено абсолютно симметричное распределение, центрированное вокруг среднего значения переменной (μ). Давайте разберемся, что́ такое нормальное распределение и что́ оно показывает, используя Excel для иллюстрации фундаментальных статистических понятий, которые на нем основаны.

Помимо прочего, нормальное распределение стоит рассмотреть еще и потому, что оно очень распространено в природе. Например, на рис. 2.5 показаны гистограммы распределения роста студентов и уровня pH вина соответственно. Эти наборы данных, **heights** (рост) и **wine** (вино), можно найти в репозитории на GitHub в папке **datasets**.

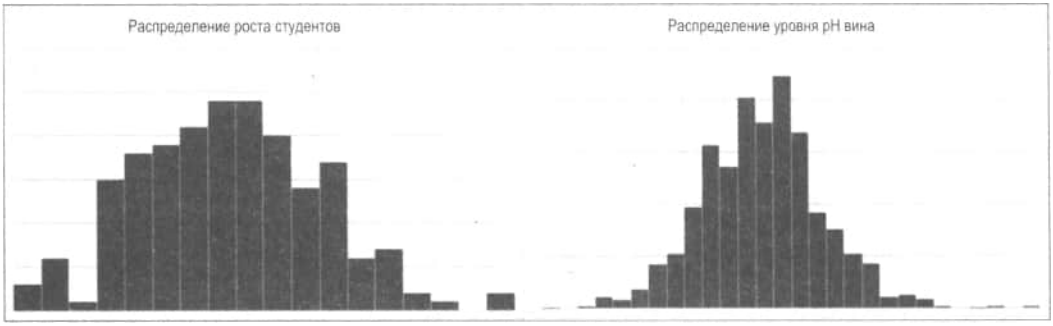


Рис. 2.5. Две нормально распределенные переменные из реальной жизни

Возможно, вам интересно, как можно узнать, что переменная имеет нормальное распределение. Это хороший вопрос. Вернемся к нашему примеру с бросанием игральной кости: мы перечислили все возможные результаты, получили теоретическое распределение, затем (с помощью моделирования) получили экспериментальное распределение и сравнили их. Гистограммы на рис. 2.5 следует рассматривать как *экспериментальные распределения*: данные в этом случае были собраны вручную, а не с помощью моделирования.

Существует несколько способов определить, достаточно ли реальный набор данных с его экспериментальным распределением близок к теоретическому нормальному распределению. Рассмотрим колоколообразную кривую: симметричная форма, большинство значений расположены вблизи центра. К другим способам относится *оценка асимметрии и эксцесса*, являющихся двумя дополнительными сводными статистиками, измеряющими симметрию и «островершинность» распределения соответственно. Также можно проверить нормальность с помощью методов статистического вывода. С основами статистического анализа вы познакомитесь в *главе 3* — будем следовать правилу: «Узнаешь, когда увидишь».

Когда мы рассматриваем реальные данные, то имеем дело с *экспериментальным* распределением. Оно никогда не будет соответствовать теоретическому распределению.

Нормальное распределение предлагает несколько легко запоминающихся указаний относительно того, какой процент наблюдений предполагается выявить в пределах

заданного числа стандартных отклонений от среднего значения. В частности, для нормально распределенной переменной мы ожидаем следующие значения:

- 68 % наблюдений будут находиться в пределах одного стандартного отклонения от среднего значения;
- 95 % наблюдений будут находиться в пределах двух стандартных отклонений от среднего значения;
- 99,7 % наблюдений будут находиться в пределах трех стандартных отклонений от среднего значения.

Это правило известно как *эмпирическое*, или *правило 68–95–99,7*. Давайте рассмотрим его в действии, используя Excel; перейдем к рабочему листу **empirical-rule**, как показано на рис. 2.6.

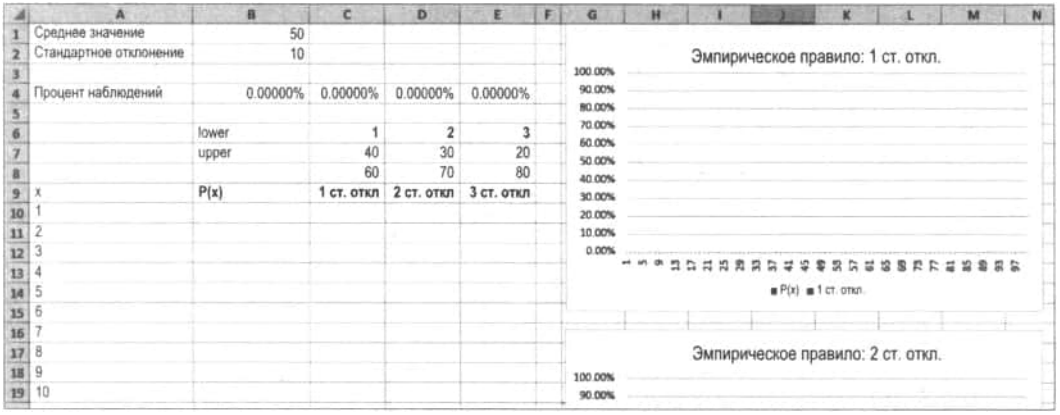


Рис. 2.6. Рабочий лист с эмпирическим правилом

В ячейках `A10:A109` расположены значения 1–100. Наша задача: определить в ячейках `B10:B109`, какой процент наблюдений примет эти значения для нормально распределенной переменной со средним значением 50 и стандартным отклонением 10 (ячейки `B1` и `B2` соответственно). Затем нужно найти, какой процент наблюдений находится в пределах одного, двух и трех стандартных отклонений среднего значения в ячейках `C10:E109`. Как только мы это сделаем, диаграммы справа заполнятся. В ячейках `C4:E4` также будет показан общий процент для каждого столбца.

Нормальное распределение является непрерывным, т. е. теоретически наблюдения могут принимать любое значение между двумя другими. Это приводит к тому, что вероятности может соответствовать *множество результатов*. Для упрощения принято группировать эти наблюдения в дискретные диапазоны. *Функция массы вероятности* (probability mass function, PMF) возвращает вероятности, найденные для каждой дискретной области в диапазоне наблюдения. Чтобы рассчитать функцию массы вероятности для нашей переменной в диапазоне 1–100, мы используем функцию Excel `NORM.DIST()`. Поскольку эта функция является более сложной, чем те, что мы использовали до сих пор, я описал каждый аргумент в табл. 2.1.

Таблица 2.1. *Аргументы функции* `NORM.DIST()`

Аргумент	Описание
`X`	Результат, для которого необходимо найти вероятность
`Mean`	Среднее значение распределения
`Standard_dev`	Стандартное отклонение распределения
`Cumulative`	Если значение `TRUE` (истинно), возвращается кумулятивная функция; если `FALSE` (ложно), возвращается функция массы

На нашем рабочем листе в столбце `A` находятся результаты, в столбцах `B1` и `B2` — среднее значение и стандартное отклонение соответственно, и мы хотим получить распределение массы вместо кумулятивного распределения. Кумулятивное распределение вернет промежуточную сумму вероятности, которая в данном случае не нужна. Для `B10` мы применим следующую формулу:

```
=NORM.DIST(A10, $B$1, $B$2, 0)
```

С помощью маркера заполнения мы получим процентную вероятность принятия наблюдением каждого значения от 0 до 100. Например, в ячейке `B43` видно, что существует приблизительно 1,1 % шанса, что наблюдение будет равно 34.

В ячейке `B4` видно, что результат, вероятно, будет между 1 и 100 более чем в 99,99 % случаев. Важно отметить, что это число не равно 100 %, т. к. наблюдение в непрерывном распределении может принимать любое возможное значение — не только от 1 до 100. В ячейках `C7:E8` я записал формулы для нахождения диапазона значений в пределах одного, двух и трех стандартных отклонений среднего значения.

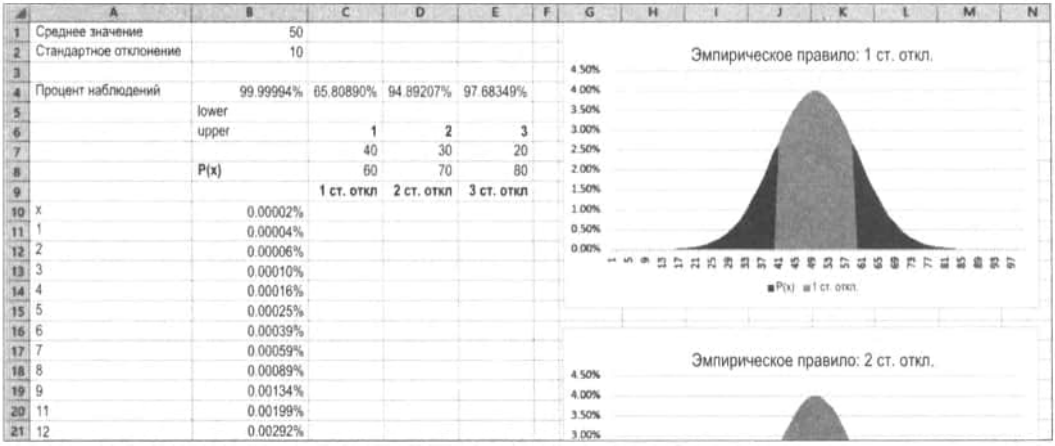
Мы можем использовать эти пороговые значения с условной логикой, чтобы определить, какие части функции массы вероятности в столбце `B` можно найти внутри этих соответствующих областей. Вставьте следующую формулу в ячейку `C10`:

```
= IF(AND($A10 > C$7, $A10 < C$8), $B10, "")
```

Эта функция перенесет вероятность из столбца `B`, если значение в столбце `A` попадает в диапазон стандартного отклонения. Если оно выходит за пределы диапазона, ячейка остается пустой. С помощью маркера заполнения можно применить эту формулу ко всему диапазону `C10:E109`. Теперь ваш рабочий лист должен выглядеть так, как показано на рис. 2.7.

Ячейки `C4:E4` указывают на то, что мы нашли 65,8, 94,9 и 99,7 % значений в пределах одного, двух и трех стандартных отклонений среднего соответственно. Эти числа очень близки к правилу 68–95–99,7.

Теперь посмотрим на итоговые визуализации: значительное большинство наблюдений находится в пределах одного стандартного отклонения, и еще больше — в пределах двух. По трем стандартным отклонениям на рис. 2.8 трудно увидеть часть диаграммы, которая *не охвачена*, но тем не менее она есть (помните: это всего лишь 0,3 % всех наблюдений).



	A	B	C	D	E
1	Среднее значение	50			
2	Стандартное отклонение	10			
3					
4	Процент наблюдений	99.99994%	65.80890%	94.89207%	97.68349%
5		lower			
6		upper	1	2	3
7			40	30	20
8		P(x)	60	70	80
9			1 ст. откл	2 ст. откл	3 ст. откл
10	x	0.00002%			
11	1	0.00004%			
12	2	0.00006%			
13	3	0.00010%			
14	4	0.00016%			
15	5	0.00025%			
16	6	0.00039%			
17	7	0.00059%			
18	8	0.00089%			
19	9	0.00134%			
20	11	0.00199%			
21	12	0.00292%			

Рис. 2.7. Демонстрация эмпирического правила в Excel

Рис. 2.8. Эмпирическое правило, визуализированное в Excel

Что произойдет, если стандартное отклонение в нашем примере изменить на 8? Или на 12? Форма колоколообразной кривой остается симметричной относительно среднего значения, равного 50, но она сжимается и расширяется: при более низком стандартном отклонении кривая «суживается», и наоборот. В любом случае эмпирическое правило приблизительно применяется к данным. Если среднее значение передвинуть к 49 или 51, «центр» кривой переместится по оси x. Переменная может иметь любое среднее значение и стандартное отклонение и все-таки будет нормально распределенной; ее функция массы вероятности изменится.

На рис. 2.9 показано нормальное распределение с разными средними значениями и стандартными отклонениями. Обе кривые, несмотря на очень разные формы, следуют эмпирическому правилу.

Нормальное распределение может иметь любую возможную комбинацию среднего значения и стандартного отклонения. Итоговая функция плотности вероятности изменится, но приблизительно будет следовать эмпирическому правилу.

Нормальное распределение также является важным из-за его места в *центральной предельной теореме, ЦПТ* (central limit theorem, CLT). Я называю эту теорему «недостающим звеном в статистике»; о причинах вы узнаете чуть позже в этой и последующих главах.

В качестве примера центральной предельной теоремы используем другую распространенную азартную игру: рулетку. Колесо европейской рулетки с равной вероят-

Рис. 2.9. Различные нормальные распределения вероятностей

ностью выдает любое число от 0 до 36 (в отличие от американской, где слоты колеса обозначены 0 и 00). С учетом того, что вы узнали о бросании игральной кости, какой это вид распределения вероятности? Дискретное равномерное. Не кажется ли вам странным, что мы анализируем это распределение в примере, который, как я уже сказал, характеризует нормальное распределение? За это нужно благодарить центральную предельную теорему. Чтобы увидеть эту теорему в действии, откройте рабочий лист **roulette-dist** и смоделируйте 100 вращений колеса рулетки в ячейках `B2:B101` с помощью функции `RANDBETWEEN()`:

```
RANDBETWEEN(0, 36)
```

Визуализируйте результат, используя гистограмму. Ваш рабочий лист должен выглядеть так, как показано на рис. 2.10. Попробуйте пересчитать результаты

	A	B
1	Испытание	Результат
2	1	36
3	2	2
4	3	0
5	4	12
6	5	8
7	6	11
8	7	33
9	8	32
10	9	8
11	10	21
12	11	33
13	12	23
14	13	29
15	14	13
16	15	14
17	16	30

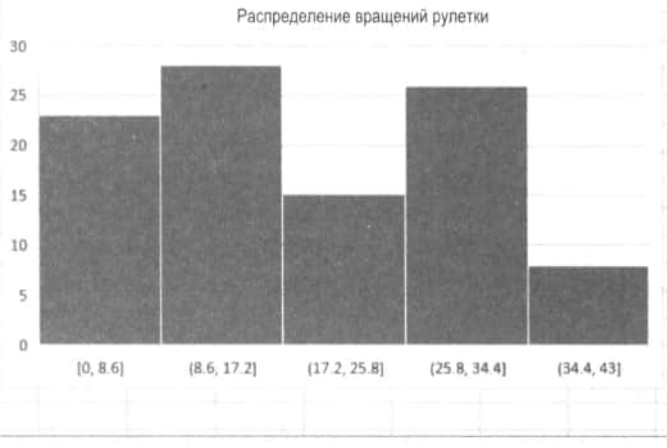


Рис. 2.10. Распределение смоделированных вращений рулетки

несколько раз. Вы увидите, что каждый раз получается довольно плоская гистограмма. Это и есть дискретное равномерное распределение, при котором любое число от 0 до 36 выпадет с одинаковой вероятностью.

Теперь откройте рабочий лист **roulette-sample-mean-dist**. Здесь мы сделаем нечто иное: имитируем 100 вращений, а затем возьмем их среднее значение. Мы сделаем это 100 раз и построим гистограмму распределения средних значений. Это «среднее от средних значений» известно как *выборочное среднее*. Выполнив это с помощью функций `RANDBETWEEN()` и `AVERAGE()`, вы должны получить график, подобный тому, что изображен на рис. 2.11.

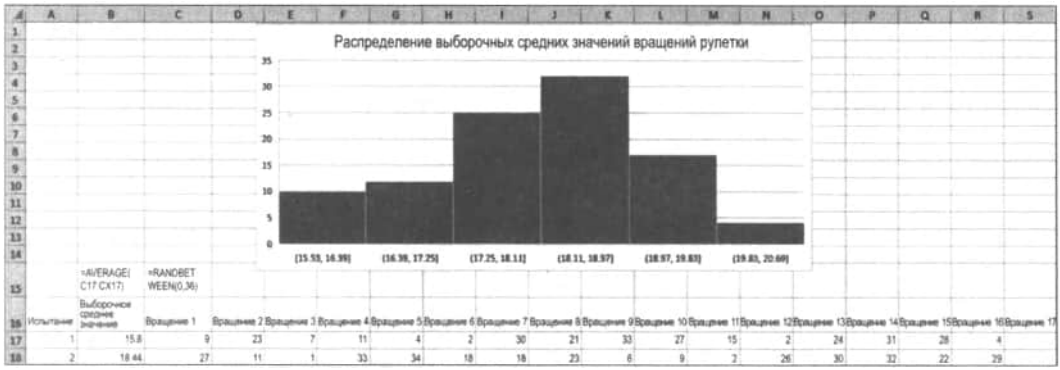


Рис. 2.11. Распределение выборочных средних значений смоделированных вращений рулетки

Это распределение уже не похоже на прямоугольник: оно выглядит как колоколообразная кривая. И оно симметрично, причем большинство наблюдений сгруппированы вокруг центра, т. е. теперь мы имеем нормальное распределение. Но как может быть нормальным распределение выборочных средних, если сами вращения рулетки таковыми не являются? Добро пожаловать в совершенно особый вид магии, известный как *центральная предельная теорема*, ЦПТ (central limit theorem, CLT).

Формально ЦПТ говорит следующее:

> Распределение выборочных средних будет нормальным или почти нормальным, если размер выборки достаточно велик.

Это явление меняет правила игры, поскольку позволяет использовать уникальные особенности нормального распределения (такие как эмпирическое правило), чтобы заявлять о выборочных средних значениях переменной, даже если сама эта переменная не имеет нормального распределения.

Вы обратили внимание на то, что написано мелким шрифтом? ЦПТ применяется только в случае, если *размер выборки достаточно велик*. Замечание важное, но неоднозначное: достаточно велик — это насколько? Попробуем разобраться с помощью еще одного примера в Excel. Чтобы воспроизвести его, откройте рабочий лист **law-of-large-numbers**. В столбце B смоделируем результаты 300 попыток вращения рулетки, используя функцию `RANDBETWEEN(0, 36)`.

В столбце C мы хотим получить скользящее среднее значение результата. Этого можно достичь с помощью смешанных ссылок; в столбце C введите следующее выражение и протащите его через все 300 попыток:

```
=AVERAGE($B$2:B2)
```

В результате будет найдено скользящее среднее столбца B. Выберите полученные результирующие данные в столбце C, перейдите в верхнее меню и щелкните **Insert** (Вставить) | **Line** (График). Взгляните на свой график и несколько раз пересчитайте данные в рабочей книге. Каждая результирующая имитация будет отличаться от показанной на рис. 2.12, но, как и там, среднее значение имеет тенденцию стремиться к 18 с увеличением числа вращений, что вполне оправданно, поскольку это среднее значение между 0 и 36. Ожидаемое значение известно как *математическое ожидание*.

	A	B	C
1	Испытание	Результат	Средний результат
2	1	21	21
3	2	32	26.5
4	3	0	17.66666667
5	4	8	15.25
6	5	8	13.8
7	6	29	16.33333333
8	7	33	18.71428571
9	8	23	19.25
10	9	4	17.55555556
11	10	2	16
12	11	28	17.09090909
13	12	29	18.08333333
14	13	27	18.76923077
15	14	33	19.78571429
16	15	9	19.06666667
17	16	7	18.3125

Рис. 2.12. Закон больших чисел, визуализированный в Excel

Данное явление известно как *закон больших чисел*, ЗБЧ (law of large numbers, LLN).

Формально ЗБЧ говорит следующее:

> Среднее значение результатов, полученных в ходе испытаний, приближается к ожидаемой величине тем больше, чем больше испытаний проведено.

Это определение, однако, поднимает вопрос, которым мы уже задавались ранее: как велик должен быть размер выборки, чтобы применять ЦПТ? Часто в качестве порогового значения предлагается число 30. Более консервативные рекомендации предлагают размер выборки 60 или 100. Взгляните на рис. 2.12 с учетом этих рекомендаций по размеру выборки. Среднее значение действительно приближается к ожидаемой величине около этих пороговых значений.

Закон больших чисел дает нестрогое эмпирическое правило для достаточного размера выборки, соответствующего ЦПТ.

Размеры выборки 30, 60 и 100 — это чисто эмпирические правила; существуют значительно более строгие способы определения размеров выборки, необходимых для применения ЦПТ. А пока что запомните следующее: с учетом того, что размер нашей выборки соответствует этим пороговым значениям, выборочное среднее должно быть близким к ожидаемой величине (благодаря ЗБЧ) и быть нормально распределенным (благодаря ЦПТ).

Существует несколько видов непрерывных распределений вероятностей, таких как экспоненциальное и треугольное. Мы сосредоточились на нормальном распределении как из-за повсеместности его применения в реальном мире, так и из-за его особых статистических свойств.

Заключение

Как говорилось в начале главы, аналитики данных живут в мире неопределенностей. Мы часто заявляем о генеральной совокупности (популяции), обладая всего лишь данными выборки. Используя принципы и основные положения, описанные в этой главе, мы сможем сделать это, одновременно количественно измеряя присущую вероятности неопределенность. В *главе 3* мы подробно остановимся на элементах *проверки гипотез*, являющейся основным методом анализа данных.

Упражнения

Используя Excel и ваши знания о вероятностях, подумайте над следующими вопросами.

1. Каково ожидаемое значение броска шестигранной игральной кости?
2. Рассмотрите переменную, которая нормально распределена со средним значением 100 и стандартным отклонением 10.
   - Какова вероятность того, что наблюдение этой переменной примет значение 87?
   - Какой процент наблюдений вы ожидаете обнаружить между 80 и 120?
3. Если математическое ожидание вращения европейской рулетки равно 18, означает ли это, что лучше ставить на этот номер, чем на какой-либо другой?

ГЛАВА 3

Основы инференциальной статистики

В *главе 1* изложены основы для исследования набора данных посредством классификации, суммирования и визуализации переменных. Все это необходимо для начала изучения аналитики, однако не следует на этом останавливаться: хотелось бы также знать, могут ли полученные выборочные данные распространяться на *генеральную совокупность* (популяцию).

Фактически мы не знаем, что́ именно найдем в генеральной совокупности, поскольку не имеем достаточных сведений о ней в целом. Однако, используя принципы вероятности, рассмотренные в *главе 2*, мы можем выявить количественную неопределенность выборки, которая также будет присутствовать и в генеральной совокупности (популяции).

Оценка значений генеральной совокупности по выборке называется *инференциальной статистикой* и осуществляется посредством *проверки гипотез*. Это основные понятия, рассматриваемые в данной главе. Возможно, вы изучали инференциальную статистику в школе, что, без достаточной ясности и практического применения, могло вызвать у вас разочарование. Именно поэтому я сделал настоящую главу как можно более практичной путем исследования реального набора данных с помощью Excel.

К концу главы вы будете владеть основными понятиями, на которых базируется значительная часть аналитики. Практическому применению полученных знаний посвящена *глава 4*.

В конце *главы 1* были приведены упражнения с набором данных **housing** (жилье), который мы будем использовать в этой главе. Найти его можно в подпапке **housing** папки **datasets** репозитория на GitHub. Сделайте копию этого набора, добавьте индексный столбец и конвертируйте набор данных в таблицу с именем **housing**.

Базовые понятия статистического вывода

Определение характеристики генеральной совокупности на основе выборки кажется чем-то невероятным, не так ли? Как и любой фокус, инференциальная статистика со стороны может показаться простой. Но для посвященных — это кульминация серий тщательно отлаженных шагов.

1. *Сбор данных для репрезентативной выборки.* Технически этот шаг *предшествует* проверке гипотез, но необходим для ее успешности. Важно, чтобы выборка достоверно отображала генеральную совокупность.

2. *Формулирование гипотез.* Прежде всего необходимо сформулировать *исследовательскую гипотезу*, т. е. некое утверждение, которое обусловливает проводимый анализ и каким-то образом объясняет генеральную совокупность. Затем необходимо сформулировать *статистическую гипотезу*, чтобы проверить, подтверждают ли данные предполагаемое объяснение.
3. *Разработка плана анализа.* Нужно наметить методы, которые будут использоваться для проведения исследования, и критерии его оценки.
4. *Анализ данных.* Здесь подразумевается обработка числовых данных и представление доказательств для оценки исследования.
5. *Принятие решения.* Наступает момент истины: предстоит сравнить критерии оценки из шага 2 с реальными результатами из шага 3 и сделать заключение о том, подтверждают ли полученные данные статистическую гипотезу.

Далее представлен краткий обзор по каждому из этих шагов. Затем мы применим изложенные концепции к набору данных **housing** (жилье).

Сбор данных для репрезентативной выборки

Как говорилось в *главе 2*, по закону больших чисел среднее значение выборки должно приближаться к ожидаемым показателям по мере увеличения размера выборки. Этот же закон дает эмпирическое правило относительно того, какая выборка является достаточной для инференциальной статистики. Однако мы предполагаем, что работаем с *репрезентативной выборкой*, т. е. набором наблюдений, который достоверно отображает закономерность. Если бы выборка не была репрезентативной, то не было бы никаких оснований предполагать, что ее выборочное среднее с увеличением количества наблюдений приближается к среднему значению генеральной совокупности.

В процессе исследования репрезентативность выборки наилучшим образом обеспечивается на этапах концептуализации и сбора данных; трудно устранить проблемы, связанные с выборкой, когда данные уже собраны. Существует множество подходов к сбору данных, являющемуся важной частью аналитического процесса, но он выходит за рамки этой книги.

Наилучший момент для обеспечения репрезентативности выборки — этап сбора данных. Если вы работаете с предварительно собранными данными, обратите внимание на шаги, предпринятые с этой целью.

Статистическая предвзятость

Сбор данных нерепрезентативной выборки — один из множества способов введения в эксперимент *статистической предвзятости*. В культурном смысле предвзятость может восприниматься как склонность или предубеждение в пользу или против какого-либо предмета или человека. В анализе данных это еще один потенциальный источник необъективности. Иными словами, что-то является статистически необъективным, если рассчитывается неким способом, который систематически отличается от истинного базового оценочного параметра. Выявление и исправление предвзятости — одна из основных задач анализа.

Получение репрезентативной выборки генеральной совокупности порождает вопрос: что есть изучаемая (целевая) совокупность? Эта совокупность может быть настолько общей или конкретной, насколько мы хотим. Например, представим, что нам нужно исследовать рост и вес собак. В этом случае совокупность может состоять из всех собак или из одной конкретной породы. Также это могут быть собаки определенной возрастной группы или пола. Некоторые целевые совокупности могут иметь более важное теоретическое значение или быть более простыми для выборки технически. Целевая совокупность может быть какой угодно, но выборка этой совокупности должна быть репрезентативной.

Набор данных **housing**, включающий 546 наблюдений, является, вероятно, достаточно большой выборкой для достоверной инференциальной статистики. Но репрезентативен ли он? Трудно сказать с уверенностью, не имея некоторого понимания о методах сбора или целевой совокупности. Эти данные взяты из рецензируемого Журнала прикладной эконометрики («Journal of Applied Econometrics»), и им можно доверять. Вы можете получить для работы недостаточно качественные данные, поэтому стоит подумать или узнать о процедурах сбора данных и составления выборки.

Что касается целевой совокупности, в файле данных **readme**, который находится в репозитории к книге: **datasets** | **housing**, указано, что источник данных — информация о продаже домов в Виндзоре, провинция Онтарио, Канада. Цены на жилье в Виндзоре могут быть лучшей целевой совокупностью; результаты, например, могут распространяться или не распространяться на цены в Канаде или даже в Онтарио. Это к тому же устаревший набор данных, взятый из статьи, написанной в 1990-х годах, поэтому нет гарантии, что выводы, полученные в результате, могут применяться к сегодняшнему рынку жилья, хотя бы даже в Виндзоре

Формулирование гипотез

Имея некоторую уверенность в том, что выборка достаточно репрезентативна для генеральной совокупности, можно начать думать, что́ именно мы хотели бы сделать, выдвигая гипотезы. Возможно, вы получили информацию о какой-то тенденции или необычном явлении в имеющихся данных. Возможно, что-то показалось вам странным при их проверке в процессе разведочного анализа данных. Самое время *подумать* о том, что́ предполагается получить в результате анализа. В случае с набором данных **housing**, пожалуй, вряд ли кто-то не согласится с тем, что в доме необходим кондиционер. Очевидно, что дома с кондиционером продаются по более высокой цене, чем без кондиционера. Такое неформальное утверждение о взаимосвязи между данными называется *исследовательской гипотезой*. Другой вариант формулирования этой взаимосвязи — кондиционирование воздуха в жилище *влияет* на его продажную цену. Дома в Виндзоре — это *генеральная совокупность* (популяция), а дома, имеющие и не имеющие кондиционер, — две ее группы, или *подмножества популяции*.

Итак, у вас есть гипотеза о влиянии наличия кондиционера на продажную цену. Это прекрасно! Для аналитика крайне важно иметь развитую интуицию и твердое мнение о своей работе. Однако, по утверждению американского инженера Уильяма

Эдвардса Деминга, «Богу верим, от остальных требуются данные». В действительности, *самое главное* — знать, присутствует ли на самом деле предполагаемая взаимосвязь в генеральной совокупности. Для этого и понадобится инференциальная статистика.

Как вы уже заметили, язык статистики, как правило, отличается от повседневного. На первый взгляд он может показаться педантичным, но его тонкости могут многое сказать о том, как работает анализ данных. Один из таких примеров — *статистическая гипотеза*. Чтобы проверить, подтверждают ли имеющиеся данные предполагаемую взаимосвязь, выдвинем две статистические гипотезы. Пока что просто взгляните на них (мы рассмотрим их немного позже).

1. *Нулевая (основная) гипотеза (H0)*. Не существует разницы в средней продажной цене между домами с кондиционированием воздуха и без него.
2. *Альтернативная гипотеза (Ha)*. Существует разница в средней продажной цене между домами с кондиционированием воздуха и без него.

Эти гипотезы являются взаимоисключающими по определению, т. е. если одна из них верна, то другая должна быть ложной. Кроме того, они проверяемы и фальсифицируемы, поскольку для их проверки и опровержения можно использовать доказательства из реальной жизни. Здесь трудно переоценить грандиозные идеи философии науки; главное, следует убедиться, что ваши гипотезы могут быть действительно проверены с помощью имеющихся данных.

Сейчас нужно оставить все предвзятые мнения, как, например, то, что составляет исследовательскую гипотезу. В данный момент мы предполагаем *отсутствие* эффекта. В конце концов, почему нет? У нас есть только *выборка* данных генеральной совокупности, поэтому мы никогда не узнаем истинное значение или параметр совокупности. Вот почему первая, т. е. *нулевая*, *гипотеза*, или H0, изложена так своеобразно.

В противовес ей существует *альтернативная гипотеза*, или Ha. Если в данных нет доказательств нулевой гипотезы, то должны быть доказательства в пользу альтернативной, учитывая их формулировки. Тем не менее никогда нельзя сказать, что мы *доказали* истинность любой из них, поскольку на самом деле нам неизвестен параметр генеральной совокупности. Вполне возможно, что эффект, обнаруженный в выборке, — это просто случайность, и в генеральной совокупности мы бы его не нашли. Фактически измерение вероятности возникновения эффекта и будет основным элементом проверки гипотез.

Результаты проверки гипотезы не «доказывают» корректность ни одной из них, поскольку «истинный» параметр совокупности изначально неизвестен.

Разработка плана анализа

Теперь, когда у нас есть статистические гипотезы, пришло время определить методы для их проверки. Соответствующий статистический критерий (тест) для той или иной гипотезы зависит от множества факторов, в том числе от типа переменных,

используемых в анализе: непрерывных, категориальных и т. д. Это еще одна причина, по которой следует классифицировать переменные в ходе разведочного анализа данных. В частности, метод (тест), который мы планируем применять, зависит от типов независимых и зависимых переменных.

Изучение причинно-следственных связей определяет большую часть процессов в аналитике; мы используем *независимые* и *зависимые* переменные для моделирования и анализа этих связей (однако помните: поскольку мы имеем дело с выборками, невозможно с уверенностью заявлять о какой-либо причинно-следственной связи). В *главе 2* мы говорили об экспериментах как о повторяющихся событиях, генерирующих определенный набор случайных результатов. В качестве примера мы использовали эксперимент с бросанием игральной кости, но в реальной жизни большинство экспериментов намного сложнее. Рассмотрим очередной пример.

Предположим, мы — исследователи, которых интересует, что́ способствует росту растений. Коллега высказал предположение, что полив растений может оказать положительное влияние. Мы решили провести эксперименты для данной гипотезы. В процессе наблюдений мы используем разное количество воды, не забывая записывать данные. Затем мы ждем несколько дней и измеряем, насколько выросли растения. В этом эксперименте у нас есть две переменные: количество воды и интенсивность роста растений. Какая переменная, по вашему мнению, является независимой, а какая — зависимой?

Полив (количество воды) — это *независимая переменная*, потому что именно его мы контролируем в рамках эксперимента. Интенсивность роста растений — *зависимая переменная*, т. к., по нашему предположению, она должна меняться при изменении независимой переменной. Независимая переменная часто записывается в первую очередь: например, растения *сначала* поливают, а *затем* они растут.

Как правило, независимые переменные записываются перед зависимыми, поскольку причина предшествует следствию.

Вернемся к примеру с продажей жилья. Как оптимально смоделировать взаимосвязи между наличием в жилище кондиционера и продажной ценой? Вполне резонно, что сначала устанавливается кондиционер, а *затем* дом продается. Это делает **airco** (кондиционер) и **price** (цена) независимой и зависимой переменными соответственно.

Чтобы проверить влияние двоичной независимой переменной на непрерывную зависимую переменную, мы будем использовать так называемый *t-тест* для *независимых выборок*. Нет смысла запоминать лучший тест для каждого отдельного случая. Наша задача состоит в том, чтобы найти общий подход к выработке правильных выводов о генеральной совокупности на основании выборки.

Большинство статистических критериев (тестов) содержат некоторые предположения об исследуемых данных. Если эти предположения не подтверждаются, результаты тестов могут быть неточными. Например, t-тест для независимых выборок предполагает, что ни одно наблюдение не влияет на другое и что каждое наблюде-

ние находится в одной и только одной группе (т. е. они *независимы*). Для адекватной оценки среднего значения генеральной совокупности в тесте обычно предполагаются нормально распределенные выборки; однако можно обойти это ограничение для больших наборов данных с помощью магии центральной предельной теоремы. Excel поможет обойти также одно допущение: дисперсия каждой совокупности одинакова.

Итак, мы знаем, какой тест будем использовать, но нам необходимо установить правила для его выполнения. Во-первых, следует определить *статистическую значимость* теста. Давайте вернемся к одному из сценариев, в котором эффект, ожидаемый в выборке, оказался случайным и не обнаружился бы в генеральной совокупности. Этот сценарий в конце концов осуществится, потому что в действительности мы никогда не узнаем среднее значение генеральной совокупности. Иными словами, мы *не уверены* в результате... и, как говорилось в *главе 2*, можно *количественно* оценить неуверенность числом от 0 до 1. Это число называется *альфой* (alpha) и выражает статистическую значимость теста.

Альфа показывает, насколько нас устраивает вероятность того, что в действительности нет никакого эффекта в генеральных совокупностях, хотя в выборках он был случайно обнаружен. Общий порог для альфы, который мы будем использовать на протяжении всей книги, составляет 5 %. Иными словами, мы признаем, что связь данных существует, если уровень значимости меньше или равен 5 %.

В этой книге мы следуем стандартному соглашению о двухсторонней проверке гипотез на уровне статистической значимости в 5 %.

Другие принятые уровни значимости — 10 или 1 %. Единого «правильного» уровня альфы не существует; величина зависит от множества факторов, таких как цель исследования, простота интерпретации и т. д.

Возможно, вы спросите: почему нас *вообще* может устраивать возможность утверждать об эффекте, когда его нет. Другими словами, почему альфа не равна 0? В этом случае мы не можем *ничего* утверждать о генеральной совокупности на основе выборки. По сути, имея альфу, равную 0, мы сказали бы, что это может быть чем угодно, поскольку мы никогда не хотим ошибаться в истинном значении совокупности. Ошибки — это риск, на который необходимо пойти, чтобы сделать хоть какой-то вывод.

Также необходимо определить интересующее нас *направление* эффекта. Например, мы предполагаем, что кондиционирование воздуха в жилище оказывает *положительный* эффект на продажную цену: средняя продажная цена домов, имеющих кондиционер, выше, чем не имеющих. Однако эффект может оказаться и отрицательным: возможно, вы имеете дело с генеральной совокупностью, предпочитающей отсутствие кондиционирования. Или, например, бывает климат, в котором использование кондиционера редко оправдано, а его наличие — просто ненужные расходы. Теоретически такие сценарии возможны; если есть какие-либо сомнения, статистический тест должен проверять *и положительные, и отрицательные* эффекты. Такой тест называется *двусторонним* (two-tailed test, или «двухвостый»

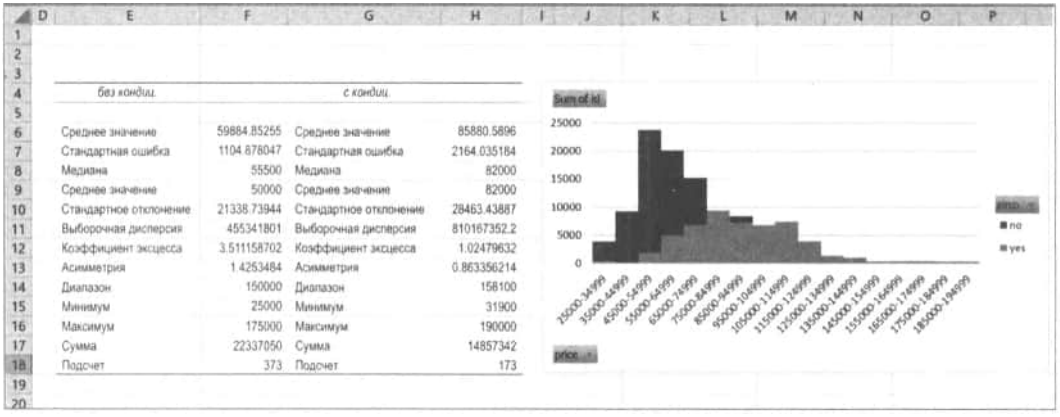
от англ. *two tail*, букв.: «два хвоста»), — его мы и будем использовать в этой книге. Также существуют односторонние тесты, или критерии (one-tailed), но они используются относительно редко и выходят за рамки данной книги.

Все это может показаться чрезмерным, — ведь мы еще даже не коснулись самих данных, но каждый шаг необходим, чтобы гарантировать: мы, аналитики, справедливо подходим к данным, когда, наконец, переходим к расчетам. Результат проверки гипотез зависит от уровня статистической значимости и количества проверенных направлений («хвостов»). Как вы увидите позже, немного отличающиеся входные данные теста, такие как уровень статистической значимости, могут привести к разным результатам. Возникает реальный соблазн сначала произвести вычисления, а *затем* подобрать конкретный тест или критерий для получения подходящего результата. Но мы должны стараться избегать соблазна подгонять результаты под цели.

Анализ данных

Наступил момент, которого вы, вероятно, ждали: обработка данных. Эта часть работы, как правило, привлекает наибольшее внимание, и сейчас мы сосредоточимся именно на ней, но не забывайте, что обработка и анализ данных — один из многих этапов проверки гипотез. Помните, что анализ данных является итеративным (повторяющимся) процессом. Маловероятно (и неправильно), что до проверки гипотез вы *не проанализировали* данные. Разведочный анализ данных в действительности разработан как предшественник проверки гипотез, или *подтверждение* анализа данных. Прежде чем начать делать выводы, следует всегда получить описательную статистику набора данных. Руководствуясь этим правилом, поступим так и с нашим набором данных **housing** (жилье) и затем перейдем к анализу.

На рис. 3.1 показаны описательная статистика и визуализация распределения цены для обоих уровней **airco**. Чтобы вспомнить, как это сделать, обратитесь к *главе 1*. Я переименовал инструментарий выходных данных, чтобы продемонстрировать измеряемые параметры в каждой группе.



	без кондиц.		с кондиц.	
Среднее значение		59884.85255	Среднее значение	85880.5896
Стандартная ошибка		1104.878047	Стандартная ошибка	2164.035184
Медиана		55500	Медиана	82000
Среднее значение		50000	Среднее значение	82000
Стандартное отклонение		21338.73944	Стандартное отклонение	28463.43887
Выборочная дисперсия		455341801	Выборочная дисперсия	810167352.2
Коэффициент эксцесса		3.511158702	Коэффициент эксцесса	1.02479632
Асимметрия		1.4253484	Асимметрия	0.863356214
Диапазон		150000	Диапазон	158100
Минимум		25000	Минимум	31900
Максимум		175000	Максимум	190000
Сумма		22337050	Сумма	14857342
Подсчет		373	Подсчет	173

Рис. 3.1. Разведочный анализ для набора данных **housing**

Гистограмма показывает, что обе группы данных (с кондиционером и без) примерно нормально распределены, и описательная статистика свидетельствует об относительно больших размерах выборки. Хотя значительно больше домов не имеют кондиционеров (373 против 173), это не обязательно является проблемой для t-теста.

Т-тест независимой выборки не чувствителен к разнице размеров выборки между двумя группами, если каждая группа *достаточно* велика. Но эта разница может влиять на другие статистические показатели.

На рис. 3.1 также показано среднее значение выборки групп: приблизительно $ 86 000 для домов с кондиционером и $ 60 000 для домов без него. Это приятно видеть, но *на самом деле* мы хотим выяснить, можно ли ожидать такого эффекта в генеральной совокупности. Вот тут наступает время t-теста, и мы снова обратимся к сводным таблицам и инструментарию анализа данных (**Analysis ToolPak**), чтобы его выполнить.

Вставьте сводную таблицу в новый рабочий лист, поместив переменную **id** в область **Rows** (Строки), переменную **airco** в область **Columns** (Столбцы) и переменную **Sum of price** в область **Values** (Значения). Удалите из отчета все общие итоги. Эту настройку данных легко выполнить в меню **t-test**, для доступа к которому выберите последовательно в верхнем меню **Data** (Данные) | **Data Analysis** (Анализ данных) | **t-Test: Two-Sample Assuming Unequal Variances** (Двухвыборочный t-тест с различными дисперсиями). Упомянутые здесь «дисперсии» — это дисперсии подмножеств популяции. В действительности мы не знаем, являются ли они одинаковыми, поэтому лучше выбрать этот вариант, предполагая равные дисперсии для более консервативных результатов.

Появится диалоговое окно; заполните его, как показано на рис. 3.2. Убедитесь, что установлен флажок **Labels** (Метки). Непосредственно над этим параметром нахо-

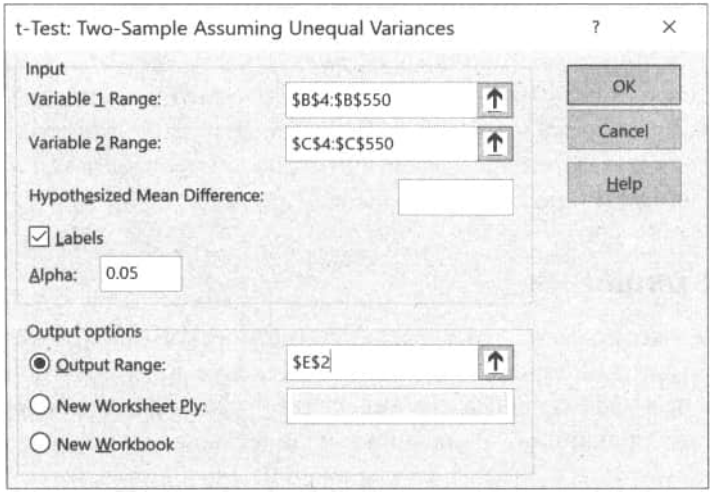


Рис. 3.2. Меню настройки t-теста в **ToolPak**

дится параметр **Hypothesized Mean Difference** (Гипотетическая средняя разность). По умолчанию данное поле пустое — это означает, что мы проверяем нулевое различие. Это и есть нулевая гипотеза, и менять ничего не надо. Ниже расположен параметр **Alpha**, означающий уровень статистической значимости; по умолчанию в Excel он устанавливается равным 5 %, что нам и нужно.

Результаты показаны на рис. 3.3, где группы обозначены как «с кондиционером» и «без кондиционера», чтобы было понятно, какие группы представлены. Далее мы рассмотрим выбранные части полученных результатов.

	D	E	F	G
1				
2				
3		Двухвыборочный t-тест с различными дисперсиями		
4			*без кондиц.*	*с кондиц.*
5		Среднее значение	59884.85255	85880.5896
6		Дисперсия	455341801	810167352.2
7		Наблюдения	373	173
8		Гипотетическая средняя разность	0	
9		df	265	
10		t Start	-10.69882731	
11		P(T<=t) one-tail	9.6667E-23	
12		t Critical one-tail	1.650623976	
13		P(T<=t) two-tail	1.93334E-22	
14		t Critical two-tail	1.968956281	
15				

Рис. 3.3. Результаты t-теста

Таким образом, мы получили некоторую информацию о выборках в ячейках `F5:G7`: их средние значения, дисперсии и размеры выборки. Также мы видим гипотетическую среднюю разность, равную нулю.

Пропустим немного статистики и посмотрим на ячейку `F13`, `P(T <= t)` с показателем *двустороннего теста* (two-tail). Возможно, вам это мало о чем говорит, но название «двусторонний» (или «двухвостый», two-tail) должно быть знакомо, поскольку данный тип теста мы решили использовать ранее вместо одностороннего (one-tail). Данный показатель называется *p-значением* (p-value), и мы будем использовать его при принятии решения о проверке гипотезы.

Принятие решения

Вы уже знаете, что альфа — это уровень статистической значимости, или уровень, по которому мы можем спокойно предположить, что в генеральной совокупности имеется реальный эффект, когда его нет, потому что эффект, обнаруженный в выборке, является случайным. P-значение количественно определяет вероятность того, что мы обнаружим в данных этот сценарий, и для принятия решения мы сравниваем его с альфой:

- если p-значение меньше или равно альфе, нулевая гипотеза отвергается;
- если p-значение больше альфы, нулевая гипотеза не отвергается.

Давайте пройдемся по этому статистическому жаргону применительно к нашим данным. Как вероятность p-значение всегда находится между 0 и 1. В нашем случае p-значение в ячейке `F13` очень маленькое, настолько маленькое, что Excel записал его в экспоненциальном представлении. Его можно прочитать как 1,93, умноженное на 10 в степени –22, — очень маленькое число. Таким образом, можно говорить, что если бы в генеральной совокупности эффект не обнаружился, мы ожидали бы найти эффект, который наблюдали в выборках, значительно меньше, чем в 1 % случаев. Это значительно ниже 5 %, поэтому мы можем отвергнуть нулевую гипотезу. Когда p-значение настолько мало, что его нужно представлять в экспоненциальной нотации, результаты часто будут обобщаться как «$p < 0{,}05$».

Предположим, что p-значение равно 0,08 или 0,24. В этих случаях мы *не могли бы отвергнуть* нулевую гипотезу. К чему такой странный язык? Почему бы просто не сказать, что мы «доказали» либо нулевую, либо альтернативную гипотезу? Все это возвращает нас к неопределенности инференциальной статистики. Мы никогда не знаем истинных значений подмножеств популяции (генеральной совокупности), поэтому безопаснее начать с предположения, что они равны. Результаты теста могут подтвердить или опровергнуть доказательства того и другого, но они никогда не смогут однозначно *доказать* их.

Поскольку p-значения используются для принятия решения о проверке гипотез, важно также знать, что́ они *не могут* показать. Например, частым заблуждением является убеждение, что p-значение — это вероятность совершения ошибки. В действительности p-значение *допускает*, что нулевая гипотеза верна, независимо от того, что́ показала выборка; предположение о наличии «ошибки» в выборке вовсе не отменяет этого допущения. P-значение говорит *только* о том, в каком проценте случаев мы обнаружили бы эффект, присутствовавший в выборке, при отсутствии эффекта в генеральной совокупности.

P-значение — это не вероятность совершения ошибки; напротив, это вероятность нахождения эффекта в выборке при отсутствии эффекта в генеральной совокупности.

Еще одно распространенное заблуждение — полагать, что чем меньше p-значение, тем больше эффект. Однако p-значение представляет собой всего лишь меру *статистической значимости*: оно говорит о том, насколько *вероятен* эффект в совокупности. P-значение не указывает на *существенную* значимость, или возможный размер эффекта. Программное обеспечение для статистических исследований, как правило, отображает только статистическую, но не существенную значимость. Выходные данные Excel относятся именно к таким случаям: хотя они возвращают p-значение, но не возвращают *доверительный интервал*, или диапазон, в пределах которого мы ожидали бы найти генеральную совокупность.

Чтобы найти доверительный интервал, можно использовать так называемое *критическое значение критерия*, показанное в ячейке `F14` на рис. 3.3. Число (1,97) может показаться произвольным, но можно правильно интерпретировать его, приняв во

внимание информацию из *главы 2*. С помощью t-теста мы взяли выборочную разность в средних ценах на жилье. Если бы мы продолжили брать случайные выборки и строить графики распределения средних разностей, это распределение было бы... *нормальным* в соответствии с центральной предельной теоремой.

Нормальное распределение и T-распределение

Для меньших размеров выборки *t-распределение* (распределение Стьюдента) используется для определения критических значений t-теста. Но с увеличением размеров выборки критические значения сходятся к значениям, найденным в нормальном распределении. Когда в этой книге я говорю о конкретных критических значениях, я использую значения, полученные из нормального распределения; они могут несколько отличаться от того, что вы видите в Excel, из-за размера выборки. Для размеров выборок, которые мы используем, исчисляемых сотнями, различия должны быть незначительными.

По эмпирическому правилу при нормальном распределении мы можем ожидать, что около 95 % наблюдений попадут в пределы двух стандартных отклонений среднего значения. В особом случае нормально распределенной переменной со средним значением, равным 0, и стандартным отклонением, равным 1 (называемым *стандартным нормальным распределением*), мы можем сказать, что около 95 % всех наблюдений попадут в диапазон от –2 до 2. Точнее, они должны попасть в диапазон между –1,96 и 1,96, и именно так определяется двустороннее критическое значение. На рис. 3.4 показана область, внутри которой мы ожидали бы найти параметр генеральной совокупности с доверительным интервалом 95 %.

Рис. 3.4. Доверительный интервал 95 % и критическое значение, визуализированные на гистограмме

Статистический критерий и критическое значение

Ячейка `F10` на рис. 3.3 возвращает *статистический критерий*. Хотя для принятия решения о проверке гипотезы мы использовали p-значения, также можно было использовать статистический критерий: если он выходит за пределы внутреннего диапазона критических значений, мы отвергаем нулевую гипотезу. Статистический критерий и p-значение в основном будут согласованными; если одно указывает на значимость, то же самое сделает и другое. Поскольку p-значение, как правило, легче интерпретировать, его используют чаще, чем статистический критерий.

Уравнение 3.1 отображает формулу для нахождения доверительного интервала двустороннего t-теста для независимых выборок. Отмеченные элементы мы затем рассчитаем в Excel.

Уравнение 3.1. *Формула для нахождения доверительного интервала*

$$c.i. = \left(\bar{X}_1 - \bar{X}_2\right) \pm ta_{/2} \times \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}.$$

Разберем эту формулу по частям. $\left(\bar{X}_1 - \bar{X}_2\right)$ представляет собой точечную оценку; $ta_{/2}$ — это критическое значение, а $\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$ — стандартная ошибка. Произведение критического значения и стандартной ошибки — это предел погрешности.

Это уравнение выглядит довольно пугающе, поэтому, чтобы сделать его более определенным, я уже рассчитал доверительный интервал и его элементы для нашего примера (рис. 3.5). Мы не будем задерживаться на формальном уравнении, а сосредоточимся на вычислении и интерпретации результатов.

	D	E	F	G
1				
2		Двухвыборочный t-тест с различными дисперсиями		
3				
4			*без кондиц.*	*с кондиц.*
5		Среднее значение	59884.85255	85880.5896
6		Дисперсия	455341801	810167352.2
7		Наблюдения	373	173
8		Гипотетическая средняя разность	0	
9		df	265	
10		t Start	-10.69882731	
11		P(T<=t) one-tail	9.6667E-23	
12		t Critical one-tail	1.650623976	
13		P(T<=t) two-tail	1.93334E-22	
14		t Critical two-tail	1.968956281	
15				
16		Точечная оценка	25995.73705	=G5-F5
17		Критическое значение	1.968956281	=F14
18		Стандартная ошибка	2429.774429	=SQRT((F6/F7)+(G6/G7))
19		Предел погрешности	4784.119625	=F17*F18
20		Нижний предел доверительного интервала	21211.61742	=F16-F19
21		Верхний предел доверительного интервала	30779.85667	=F16+F19

Рис. 3.5. Вычисление доверительного интервала в Excel

Во-первых, обратите внимание на *точечную оценку* в ячейке `F16`, или эффект в генеральной совокупности, который мы, скорее всего, обнаружим. Это разница в выборочных средних значениях. В сущности, если выборка является репрезентативной по отношению к генеральной совокупности, то разница в выборке и математическом ожидании должна быть незначительной. Но, вероятно, они не будут одинаковыми; мы получим диапазон значений, внутри которого на 95 % уверены, что обнаружим истинное различие.

Далее, *критическое значение* в ячейке F17. Excel предоставил это значение, но я указал его здесь для простоты анализа. Как уже упоминалось ранее, мы можем использовать эту величину, чтобы найти 95 % значений в пределах примерно двух стандартных отклонений от среднего.

В ячейке F18 находится *стандартная ошибка*. Вы уже встречали этот термин, в итоговых данных описательной статистики инструментария анализа данных (**Analysis ToolPak**) (см., например, рис. 3.1). Чтобы понять, как работает стандартная ошибка, представьте, что вам пришлось снова и снова выполнять повторную выборку цен на жилье из генеральной совокупности. Каждый раз вы получали бы несколько иное выборочное среднее значение. Эта изменчивость известна как *стандартная ошибка*. Чем больше отклонение стандартной ошибки от среднего значения, тем менее точно выборка представляет генеральную совокупность.

Стандартную ошибку одной выборки можно рассчитать, разделив ее стандартное отклонение на размер выборки. Поскольку нужно найти стандартную ошибку для *разности* между средними значениями, формула будет немного сложнее, но шаблон сохраняется тот же самый: в числителе будет стоять изменчивость выборок, а в знаменателе — количество наблюдений. Это вполне резонно, поскольку мы ожидаем большей изменчивости в выборочной разности, если каждое среднее значение выборки само по себе содержит большую изменчивость. Так как мы собираем выборки большего размера, то ожидаем, что они будут менее изменчивы по отношению к генеральной совокупности.

Теперь возьмем произведение критического значения и стандартной ошибки, чтобы получить *предел погрешности* в ячейке F19. Возможно, этот термин вам знаком, — он часто включается в результаты опросов. Предел погрешности позволяет оценить, насколько велика изменчивость вокруг точечной оценки. Это видно в случае, показанном на рис. 3.5: хотя мы считаем, что разница в совокупности составляет $ 25,996, однако можем ошибиться на целых $ 4,784.

Поскольку это двусторонний тест (two-tailed test), разница может быть найдена в *любом направлении*. Поэтому нам необходимо и вычесть, и прибавить предел погрешности, чтобы определить верхнюю и нижнюю границы доверительного интервала. Эти значения представлены в ячейках F20 и F21 соответственно. В чем тут суть? Мы на 95 % уверены, что средняя продажная цена домов с кондиционером на сумму от $ 21,211 до 30,780 выше, чем домов без кондиционера.

Зачем прилагать столько усилий, чтобы получить доверительный интервал? Как показатель существенной, а не статистической значимости, он часто больше подходит для широкой аудитории, поскольку переводит результаты проверки статистической гипотезы обратно на язык исследовательской гипотезы. Например, представьте, что вы аналитик в банке, отчитывающийся о результатах исследования цен на жилье перед руководством. Ваши руководители не знали бы, с чего начать выполнение t-теста, если бы от этого зависели их карьеры, — и их карьеры *действительно* зависят от правильности принятия решений на основе этого анализа, поэтому вы постараетесь сделать его как можно более понятным. Какое из утверждений, представленных ниже, более полезно и понятно, по вашему мнению?

- «Мы отвергли нулевую гипотезу о том, что не существует разницы в средней продажной цене домов с кондиционером и без него при $p < 0,05$».
- «С уверенностью на 95 % можно сказать, что средняя продажная цена домов с кондиционером примерно на сумму от $ 21,200 до 30,800 выше, чем домов без кондиционера».

Почти каждый поймет суть второго утверждения, тогда как первое требует значительных статистических познаний и навыков. Однако доверительные интервалы предназначены не только для непрофессионалов: в кругах ученых и аналитиков данных они также используются наряду с p-значениями. Наконец, p-значение измеряет *только* статистический эффект, но *не* измеряет существенный.

Хотя p-значения и доверительный интервал показывают результаты с разных сторон, принципиально они *всегда согласуются*. Для иллюстрации этого утверждения проведем еще одну проверку гипотезы с использованием набора данных **housing**. На этот раз нам нужно узнать, есть ли существенная разница в среднем размере участка (**lotsize**) между домами с полностью оборудованным цокольным (**fullbase-yes**) этажом и без него (**fullbase-no**). Эта связь также может быть проверена с помощью t-теста; я выполню те же самые шаги, что и раньше, на новом рабочем листе, а результаты теста отражены на рис. 3.6 (не забудьте сначала проанализировать описательную статистику новых переменных).

	D	E	F	G	H
1					
2		Двухвыборочный t-тест с различными дисперсиями			
3					
4			*fullbase-no*	*fullbase-yes*	
5		Среднее значение	5074.814085	5290.502618	
6		Дисперсия	4683966.27	4726820.23	
7		Наблюдения	355	191	
8		Гипотетическая средняя разность	0		
9		df	387		
10		t Start	-1.107303893		
11		P(T<=t) one-tail	0.134425163		
12		t Critical one-tail	1.648800515		
13		P(T<=t) two-tail	0.268850325		
14		t Critical two-tail	1.966112774		
15					
16		Точечная оценка	215.6885333	=G5-F5	
17		Критическое значение	1.966112774	=F14	
18		Стандартная ошибка	194.78711873	=SQRT((F6/F7)+(G6/G7))	
19		Предел погрешности	382.9734396	=F17*F18	
20		Нижний предел доверительного интервала	-167.2849064	=F16-F19	
21		Верхний предел доверительного интервала	598.6619729	=F16+F19	
22					

Рис. 3.6. Влияние наличия полностью оборудованного цокольного этажа на размер участка

Результаты этого теста не являются статистически значимыми: с учетом p-значения, равного 0,27, мы ожидали бы найти эффект более чем в четверти выборок, предполагая отсутствие его в генеральной совокупности. Что касается существен-

ной значимости, мы на 95 % уверены, что разница в среднем размере участка находится примерно в диапазоне от –167 до 599 квадратных футов. Иными словами, истинное различие может быть как положительным, так и отрицательным, и мы не можем говорить об этом с уверенностью. Основываясь на любом из этих результатов, нельзя отвергнуть нулевую гипотезу, поскольку не существует значимого различия в среднем размере участка. Эти результаты всегда будут согласовываться, т. к. и тот и другой частично основываются на уровне статистической значимости: альфа определяет, как мы оцениваем p-значение, и устанавливает критическое значение, используемое для получения доверительного интервала.

Проверка гипотез и интеллектуальный анализ данных

Сомнительно, что мы действительно *ожидали* найти существенную связь между размером участка и полностью оборудованным цокольным этажом. В сущности, как связаны эти переменные, менее понятно по сравнению с тем, как наличие кондиционера может влиять на продажную цену. На самом деле я решил протестировать этот параметр, чтобы получить предполагаемую *незначительную* связь исключительно в демонстрационных целях. В большинстве других случаев более перспективным будет проанализировать данные и поискать *значимые* связи. Простые вычисления позволяют более свободно подходить к анализу данных, но, если вы не можете объяснить результаты своего анализа на основе логики, теории или предыдущих эмпирических доказательств, вам следует относиться к ним осторожно, независимо от того, насколько сильными или убедительными они являются.

Если вам приходилось строить финансовую модель, вы, возможно, знакомы с анализом «что если» (анализ альтернатив), который позволяет увидеть, как может меняться результат с учетом различных входных данных и допущений. В этом ключе давайте проверим, как могли отличаться результаты нашего t-теста размера подвала/участка. Поскольку мы будем работать с результатами инструментария (ToolPak) анализа данных, имеет смысл скопировать и вставить данные из ячеек `E2:G21` в новый диапазон, чтобы сохранить оригинал. Я помещу свои данные в ячейки `J2:L22` текущего рабочего листа. Также я переименую выходные данные и выделю ячейки `K6:L6` и `K14` — так будет понятно, что они были изменены.

Давайте теперь «поиграем» с размерами выборки и критическим значением. Не глядя на полученный доверительный интервал, попробуйте угадать, что́ произойдет, основываясь на своих знаниях о взаимосвязи этих величин. Во-первых, я установлю размер выборки равным 550 наблюдениям для каждой группы. Это рискованный шаг; *в действительности* мы не собрали 550 наблюдений, но чтобы разобраться в статистике, временами приходится «испачкать руки». Затем мы изменим статистическую значимость с 95 на 90 %. Результирующее критическое значение равно 1,64. Это тоже рискованно, поскольку статистическая значимость должна быть закреплена перед анализом по причинам, которые вы увидите прямо сейчас.

Рис. 3.7 отображает результат анализа «что если» (what-if analysis). Наш доверительный интервал между $ 1 и 430 указывает на статистическую значимость, хотя и с большим трудом, ведь она чрезвычайно близка к нулю

Существуют способы рассчитать соответствующее p-значение, но поскольку вы знаете, что оно всегда принципиально согласуется с доверительным интервалом,

	D	I	J	K	L
1					
2			Двухвыборочный t-тест с различными дисперсиями. АНАЛИЗ «ЧТО ЕСЛИ»		
3					
4				*fullbase-no*	*fullbase-yes*
5			Среднее значение	5074.814085	5290.502618
6			Дисперсия	4683966.27	4726820.23
7			Наблюдения	550	550
8			Гипотетическая средняя разность	0	
9			df	387	
10			t Start	-1.107303893	
11			P(T<=t) one-tail	0.134425163	
12			t Critical one-tail	1.648800515	
13			P(T<=t) two-tail	0.268850325	
14			t Critical two-tail	1.64	
15			Точечная оценка	215.6885333	=L5-K5
16			Критическое значение	1.64	=K14
17			Стандартная ошибка	130.8071898	=SQRT((K6/K7)+(L6/L7))
18			Предел погрешности	$215	=K17*K18
19			Нижний предел доверительного интервала	$1	=K16-K19
20			Верхний предел доверительного интервала	$430	=K16+K19
21					
22					

Рис. 3.7. Результаты анализа «что если» доверительного интервала

мы пропустим это упражнение. Теперь наш тест стал значимым, и это может иметь решающее значение для финансирования, известности и славы.

Мораль этой истории в том, что результаты проверки гипотез легко могут быть подтасованы. Иногда нужен всего лишь другой уровень статистической значимости, чтобы решить исход дела в пользу отказа от нулевой гипотезы. Привести к этому может и повторная выборка или, как в нашем примере, ложное увеличение числа наблюдений. Даже при отсутствии умышленных нарушений всегда будет существовать область неопределенности в попытках найти параметр генеральной совокупности, который фактически неизвестен.

Это ваш мир... данные только живут в нем

Весьма заманчиво при инференциальной статистике механически подставлять известные p-значения без учета более обширных аспектов сбора данных или существенной значимости. Вы уже поняли, насколько могут быть чувствительны результаты к изменениям в статистической значимости или размере выборки. Чтобы увидеть другую возможность, рассмотрим еще один пример из набора данных **housing**.

Самостоятельно протестируйте на значимое различие продажную цену домов, в которых используется (**gas-yes**) или не используется (**gas-no**) газ для нагрева воды. Соответствующие переменные — **price** (цена) и **gashw** (газ в доме). Результаты показаны на рис. 3.8.

Нельзя отвергнуть нулевую гипотезу, основываясь только на p-значении: оно все-таки больше 0,05. Но 0,067 *не очень сильно* отличается, поэтому здесь надо быть внимательнее. Прежде всего, рассмотрите размер выборки: имея всего 25 наблюдений домов с газом, возможно, стоит собрать больше данных, прежде чем оконча-

	F	G	H	I	J
1					
2		Двухвыборочный t-тест с различными дисперсиями			
3					
4					
5			*gas-yes*	*gas-no*	
6		Среднее значение	79428	67579.06334	
7		Дисперсия	923472100	698250450.3	
8		Наблюдения	25	521	
9		Гипотетическая средняя разность	0		
10		df	26		
11		t Start	1.915131244		
12		P(T<=t) one-tail	0.033268787		
13		t Critical one-tail	1.70561792		
14		P(T<=t) two-tail	0.066537575		
15		t Critical two-tail	2.055529439		
16					
17		Точечная оценка	-11848.93666	=I6-H6	
18		Критическое значение	2.055529439	=H15	
19		Стандартная ошибка	6187.010263	=SQRT((H7/H8)+(I7/I8))	
20		Предел погрешности	12717.58173	=H18*H19	
21		Нижний предел доверительного интервала	-24566.51839	=H17-H20	
22		Верхний предел доверительного интервала	868.6450729	=H17+H20	
23					

Рис. 3.8. Результаты t-теста влияния использования газа на продажную цену

тельно отвергнуть нулевую гипотезу. Вы, конечно, обратили бы внимание на размер выборки до начала теста в процессе описательной статистики.

Аналогично доверительный интервал показывает, что истинное значение находится в диапазоне приблизительно между –900 и $ 24,500. С учетом таких сумм имеет смысл глубже разобраться в этом вопросе. Если вы просто вслепую отвергните нулевую гипотезу, опираясь на p-значение, вы упустите потенциально важную связь. Будьте внимательны к этим потенциальным «пограничным случаям»: если один из них обнаружился в конкретном наборе данных, не сомневайтесь, что проявятся и другие.

Статистика и аналитика являются мощными инструментами для понимания мира, но они только то, что они есть, — инструменты. Без уверенного и умелого специалиста они в лучшем случае бесполезны, а в худшем — вредны. Не стоит полностью доверять p-значению; рассмотрите более широко, как работает статистика, а также проанализируйте цель, к которой вы стремитесь (без подтасовки результатов, что, как вы видели, вполне возможно). Помните: это ваш мир, данные только живут в нем.

Заключение

В начале у вас мог возникнуть вопрос, почему в книге по аналитике целая глава посвящена такой малопонятной теме, как вероятность. Теперь, я думаю, связь ясна: поскольку параметры совокупности неизвестны, необходимо количественно выра-

зить эту неопределенность как вероятность. В данной главе мы опирались на инференциальную статистику и проверку гипотез для исследования разницы в средних значениях между двумя группами. В следующей главе мы будем изучать влияние одной переменной на другую с помощью метода *линейной регресии*, о котором вы, вероятно, слышали. Статистическая основа, несмотря на другой критерий, остается той же.

Упражнения

Теперь ваша очередь делать вероятностные выводы о наборе данных. В репозитории на GitHub найдите набор данных **tips.xlsx** в подпапке **tips** (чаевые) папки **datasets** и выполните следующие упражнения:

1. Протестируйте связь между временем суток (ланч или ужин — lunch, dinner) и общим счетом (total bill) и ответьте на следующие вопросы.
   - Каковы ваши статистические гипотезы?
   - Являются ли ваши результаты статистически значимыми? Каким образом это доказывает ваши гипотезы?
   - Какова предполагаемая величина эффекта?
2. Ответьте на те же вопросы, но относительно связи между временем суток и чаевыми (tips).

ГЛАВА 4

Корреляция и регрессия

Вы слышали, что потребление мороженого связано с нападениями акул? Вероятно, оно возбуждает у акул зверский аппетит. Эта предполагаемая взаимосвязь показана на рис. 4.1.

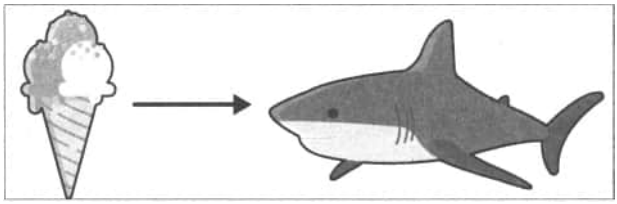


Рис. 4.1. Предполагаемая связь между потреблением мороженого и нападениями акул

«Нет, — можете возразить вы, — не обязательно атаки акул вызваны потреблением мороженого».

«Возможно, — рассуждаете вы, — с повышением температуры на улице потребляется больше мороженого, потому что люди проводят больше времени у океана в теплую погоду, и *именно это* стечение обстоятельств приводит к увеличению количества нападений акул».

«Корреляция не подразумевает причинно-следственную связь»

Вероятно, вы много раз слышали, что «корреляция не подразумевает причинно-следственную связь».

Из *главы 3* вы узнали, что причинно-следственная связь в статистике — очень сложная вещь. В действительности мы отвергаем нулевую гипотезу только потому, что просто не имеем всех данных, необходимых для уверенного заявления о причинно-следственной связи. За исключением семантического различия, имеет ли корреляция какое-либо отношение к причинно-следственной связи? Их связь в стандартном понимании несколько упрощена; почему это так — вы увидите в этой главе, используя инструменты инференциальной статистики, изученные ранее.

Эта последняя глава, в основном посвященная Excel. Изучив ее, вы достаточно овладеете аналитикой, чтобы переходить к языкам R и Python.

Понятие корреляции

До сих пор мы в основном анализировали статистические данные по одной переменной: например, мы вычислили средний балл за чтение или дисперсию цен на жилье. Такой анализ называется *одномерным анализом*.

Также мы осуществляли *двумерный анализ*. Например, с помощью двумерной таблицы частот сравнивали частоты двух категориальных переменных. Кроме того, мы проанализировали непрерывную переменную, сгруппированную по нескольким уровням категориальной переменной, выведя описательную статистику для каждой группы.

Теперь, используя корреляцию, мы произведем двумерный анализ двух непрерывных переменных. Точнее, мы будем использовать *коэффициент корреляции Пирсона* для измерения степени *линейной зависимости* (корреляции) между двумя переменными. Без линейной зависимости корреляция Пирсона использоваться не может.

Итак, как же мы узнаем, что наши данные являются линейными? Проверить это можно более строгими методами, но обычно начинают с визуализации. Мы воспользуемся *диаграммой рассеяния*, чтобы отобразить все наблюдения, основываясь на координатах x и y.

Если через диаграмму рассеяния можно провести линию, которая отображает общую картину, значит, это линейная зависимость (корреляция) и, следовательно, может быть использована корреляция Пирсона. Если для отображения общей картины нужна кривая или иная форма, корреляция Пирсона использоваться не может. На рис. 4.2 показаны одна линейная и две нелинейные зависимости, или корреляции.

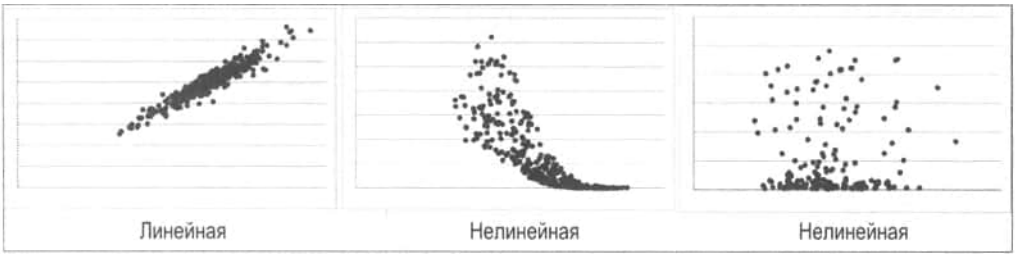


Рис. 4.2. Линейная и нелинейная зависимости (корреляции)

На самом деле на рис. 4.2 представлен пример *положительной линейной зависимости* (корреляции): по мере увеличения значений на оси x, значения на оси y также увеличиваются (с линейной скоростью).

Также возможна *отрицательная линейная зависимость* (корреляция), когда связь отображается отрицательной линией, и *нулевая*, когда корреляция вообще отсутствует и связь отображается плоской кривой. На рис. 4.3 показаны все эти виды линейной корреляции. Помните: для применения корреляции Пирсона все зависимости должны быть линейными.

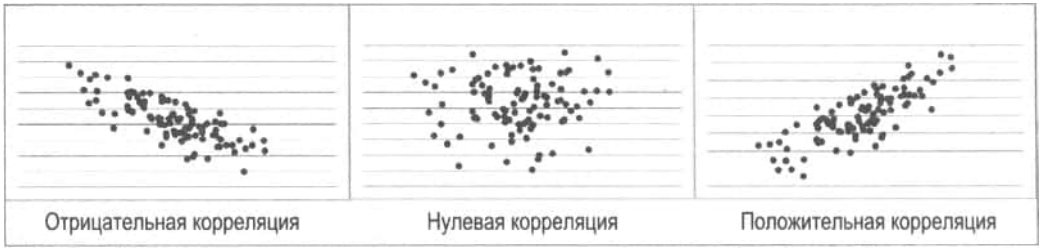


Рис. 4.3. Отрицательная, нулевая и положительная корреляции

Когда мы установили, что имеем дело с линейными данными, можно найти коэффициент корреляции. Он всегда принимает значение между –1 и 1, где –1 — абсолютная отрицательная линейная зависимость, 1 — абсолютная положительная линейная зависимость и 0 — отсутствие линейной зависимости вообще. В табл. 4.1 представлены некоторые эмпирические правила для оценки степени коэффициента корреляции. Они не являются официальным стандартом, но полезны как отправная точка для интерпретации полученных коэффициентов.

Таблица 4.1. *Интерпретация коэффициентов корреляции*

Коэффициент	Интерпретация
–1,0	Абсолютная отрицательная линейная зависимость
–0,7	Сильная отрицательная зависимость
–0,5	Средняя отрицательная зависимость
–0,3	Слабая отрицательная зависимость
0	Отсутствие линейной зависимости
+0,3	Слабая положительная зависимость
+0,5	Средняя положительная зависимость
+0,7	Сильная положительная зависимость
+1,0	Абсолютная положительная линейная зависимость

Опираясь на эту концептуальную основу, проведем корреляционный анализ с использованием Excel. Мы будем использовать набор данных пробега транспортного средства; в репозитории на GitHub найдите файл **mpg.xlsx** в подпапке **mpg** (пробег, миль на галлон) папки **datasets**. Это новый набор данных, поэтому уделите время, чтобы познакомиться с ним и понять, с какими типами переменных мы будем работать. Обработайте и визуализируйте данные, используя инструменты, описанные в *главе 1*. Для упрощения последующего анализа не забудьте добавить индексный столбец и конвертировать набор данных в таблицу, которая будет называться **mpg**.

Для расчета коэффициента корреляции между массивами в Excel существует функция `CORREL()`:

```
CORREL(array1, array2)
```

Давайте используем эту функцию, чтобы найти корреляцию между массивами **weight** (вес) и **mpg** (миль на галлон) в нашем наборе данных:

```
=CORREL(mpg[weight], mpg[mpg])
```

Данная функция возвращает значение между –1 и 1: –0.832. (Вы помните, как это интерпретируется?)

Матрица корреляции, или корреляционная матрица, представляет корреляции по всем парам переменных. Давайте построим ее с помощью пакета инструментов анализа данных (Analysis ToolPak). В верхнем меню выберите последовательно **Data** (Данные) | **Data Analysis** (Анализ данных) | **Correlation** (Корреляция).

Помните, что мы измеряем линейную зависимость между двумя *непрерывными* переменными, поэтому необходимо исключить категориальные переменные, такие как **origin** (регион происхождения), и быть осторожными относительно включения дискретных переменных, таких как **cylinders** (цилиндры) или **model.year** (год модели). Для корректного функционирования пакета инструментов анализа (Analysis ToolPak) все переменные должны находиться в непрерывном диапазоне, поэтому я осторожно включу в него переменную **cylinders**. На рис. 4.4 показано, как должно выглядеть исходное меню **ToolPak**.

В результате получится корреляционная матрица, как на рис. 4.5.

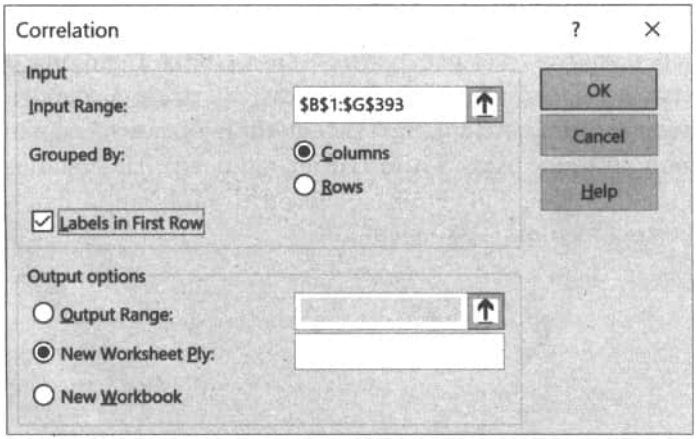


Рис. 4.4. Вставка корреляционной матрицы в Excel

	A	B	C	D	E	F	G
1		*Миль на галлон*	*Цилиндры*	*Литраж*	*Мощность (в л. с.)*	*Вес*	*Ускорение*
2	Миль на галлон	1					
3	Цилиндры	−0.777617508	1				
4	Литраж	−0.805126947	0.950823301	1			
5	Мощность (в л. с.)	−0.778426784	0.842953357	0.897257002	1		
6	Вес	−0.832244215	0.89752734	0.932994404	0.864537738	1	
7	Ускорение	0.423328537	−0.504683379	−0.543800497	−0.68919551	−0.416839202	1
8							
9							
10							

Рис. 4.5. Корреляционная матрица в Excel

В ячейке B6 находится величина –0,83 — это пересечение переменных **weight** (вес) и **mpg** (миль на галлон). Такую же величину мы могли бы увидеть бы в ячейке F2, но эту половину матрицы Excel оставил пустой, т. к. данная информация избыточна. Все величины, расположенные по диагонали, равняются 1, поскольку любая переменная полностью коррелирована сама с собой.

Коэффициент корреляции Пирсона может использоваться только в случае, если зависимость между двумя переменными является линейной.

Проанализировав корреляции переменных, мы значительно продвинулись в наших предположениях относительно их характеристик. Каковы они, по вашему мнению? *Предположим, что их корреляция — линейная*. Давайте проверим это предположение с помощью диаграммы (графика) рассеяния. К сожалению, в базовом пакете Excel нет возможности генерировать диаграммы рассеяния каждой пары переменных одновременно. Для тренировки можно построить их все, но мы попробуем сделать это с переменными **weight** и **mpg**. Выделите эти переменные и затем в верхнем меню выберите последовательно **Insert** (Вставить) | **Scatter** (Рассеяния).

Я добавлю свой заголовок диаграммы и для простоты интерпретации переназначу оси. Чтобы изменить заголовок диаграммы, дважды щелкните на нем. Чтобы переназначить ось, щелкните в любом месте периметра диаграммы и нажмите значок «плюс», который появится для раскрытия меню **Chart Elements** (Элементы диаграммы). На Mac щелкните внутри диаграммы и затем выберите **Chart Design** (Конструктор диаграммы) | **Add Chart Element** (Добавить элемент диаграммы). Выберите название в меню **Axis Titles** (Названия осей). Получившаяся диаграмма рассеяния показана на рис. 4.6. Чтобы облегчить понимание данных диаграммы, стоит добавить на оси единицы измерения.

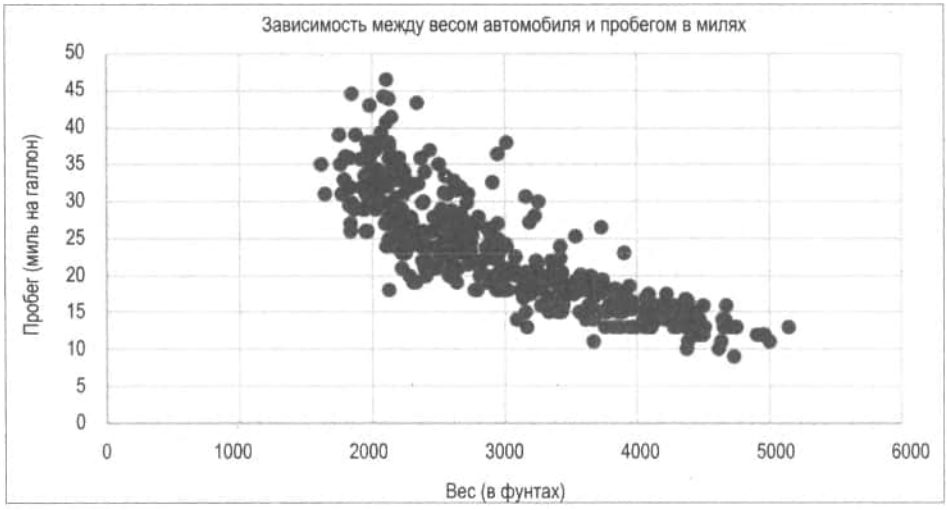


Рис. 4.6. Диаграмма рассеяния веса и пробега

По сути диаграмма на рис. 4.6 выглядит как отрицательная линейная зависимость с бóльшим рассеянием при меньшем весе и большем пробеге. По умолчанию Excel расположил первую переменную выборки данных вдоль оси x, а вторую — вдоль оси y. Но почему не наоборот? Попробуйте изменить порядок этих столбцов на рабочем листе так, чтобы переменная **weight** (вес) находилась в столбце E, а переменная **mpg** (пробег) — в столбце F, и затем вставьте новую диаграмму рассеяния.

На рис. 4.7 показано зеркальное отображение зависимости. Excel — отличный инструмент, но, как и любому инструменту, ему нужно объяснить, что́ делать. Excel вычислит корреляции независимо от того, является ли зависимость линейной. Кроме того, он построит диаграмму рассеяния без учета расположения переменных вдоль той или иной оси.

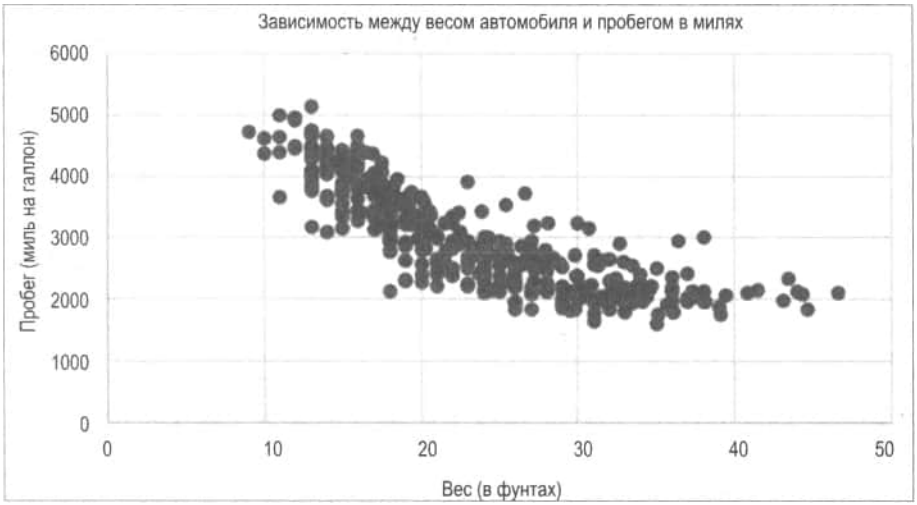


Рис. 4.7. Диаграмма рассеяния пробега и веса

Итак, какая из диаграмм рассеяния является правильной? Имеет ли это значение? По общепринятому соглашению независимая переменная располагается по оси x, а зависимая — по оси y. Подумайте, какими являются эти переменные. Если вы не уверены, имейте в виду, что обычно независимая переменная измеряется в первую очередь.

В данном случае *независимой* переменной является **weight**, т. к. она определяется моделью и сборкой автомобиля. Переменная **mpg** — зависимая, поскольку, по нашему предположению, на нее влияет вес автомобиля. Поэтому переменная **weight** располагается по оси x, а переменная **mpg** — по оси y.

В бизнес-аналитике редко собирают данные только для статистического анализа; например, машины из набора данных **mpg** изготовлены для того, чтобы приносить прибыль, а не для изучения влияния веса автомобиля на пробег в милях. Поскольку не всегда ясно, какие переменные являются независимыми, а какие — зависимыми, тем более важно понимать, *что́* измеряют эти переменные и *ка́к* они измеряются. Именно поэтому следует знать предметную область исследования или хотя бы

характеристики переменных, а также то, каким образом осуществлялся сбор наблюдений.

От корреляции к регрессии

Хотя принято располагать независимую переменную по оси x, это не имеет никакого значения для коэффициента корреляции. Однако тут есть серьезная оговорка, связанная с идеей использовать линию для отображения зависимости, полученной на диаграмме рассеяния. Такая практика несколько расходится с корреляцией, и вы, возможно, уже слышали о ней, — это *линейная регрессия*.

Для корреляции не имеет значения, какую переменную вы назовете независимой, а какую — зависимой; это не учитывается в ее определении как «степени линейного движения двух переменных».

Вместе с тем в линейной регрессии эта зависимость присутствует и определяется как «предполагаемое влияние единичного изменения независимой переменной **X** на зависимую переменную **Y**».

Вы увидите, что линия, которая проходит через диаграмму рассеяния, может быть выражена уравнением; в отличие от коэффициента корреляции, это уравнение зависит от определения независимой и зависимой переменных.

Линейная регрессия и линейные модели

Линейная регрессия часто упоминается как *линейная модель*, которая является всего лишь одной из множества статистических моделей. Так же как модель поезда, которую вы можете построить, статистическая модель может служить приближением реального предмета. В сущности, мы используем статистические модели для понимания взаимосвязи между зависимыми и независимыми переменными. Модели не могут *полностью* описать представляемый объект, но это не значит, что они бесполезны. По известному выражению британского математика Джорджа Бокса, «все модели ошибочны, но некоторые полезны».

Так же как и корреляция, линейная регрессия предполагает, что существует линейная зависимость между двумя переменными. Существуют и другие предположения, которые необходимо принимать во внимание при моделировании данных. Например, нам не нужны предельные наблюдения, которые могут несоразмерно влиять на общую тенденцию линейной зависимости.

В целях демонстрации мы пока не будем принимать во внимание это и другие предположения. Часто такие предположения сложно проверить с помощью Excel. Ваши знания статистического программирования пригодятся при более глубоком погружении в линейную регрессию.

Итак, соберитесь с духом, ибо пришло время еще одного уравнения.

Уравнение 4.1. *Уравнение для линейной регрессии*

$$Y = \beta_0 + \beta_1 \times X + \in$$

Цель уравнения 4.1 — предварительно вычислить зависимую переменную Y. Это левая часть уравнения. Вы, возможно, помните из школьного курса, что линия может быть разбита на точку пересечения (перехват) и наклон. Именно тут появляют-

ся β_0 и $\beta_1 \times X_i$ соответственно. Во второй части уравнения мы умножаем независимую переменную на коэффициент наклона.

Наконец, корреляцию между независимой и зависимой переменными можно объяснить не непосредственно моделью, а некоторым внешним влиянием. Это называется *ошибкой* модели и обозначается ε_i.

Ранее мы использовали t-тест независимых выборок для изучения значимого различия средних значений между двумя группами. Сейчас мы измеряем линейное влияние одной непрерывной переменной на другую. Мы сделаем это, проверив, отличается ли *наклон* линии регрессии соответствия от нуля статистически. Это означает, что наша проверка гипотезы будет работать примерно следующим образом.

- *Нулевая гипотеза* (*H0*): Линейное влияние независимой переменой на зависимую переменную отсутствует (наклон линии регрессии равен нулю).
- *Альтернативная гипотеза* (*Ha*): Линейное влияние независимой переменой на зависимую переменную существует (наклон линии регрессии не равен нулю).

На рис. 4.8 показано несколько примеров значительного и незначительного наклона.

Помните: у нас нет *всех* данных, поэтому мы не знаем, каким будет «истинный» наклон для совокупности значений. Напротив, мы делаем логический вывод, является ли наклон, с учетом нашей выборки, статистически отличным от нуля. Чтобы оценить значимость наклона, можно использовать ту же самую методику p-значения, что и для нахождения разницы средних значений для двух групп. Мы продолжим проводить двусторонние тесты с доверительным интервалом 95 %. Давайте перейдем к поиску результата с использованием Excel.

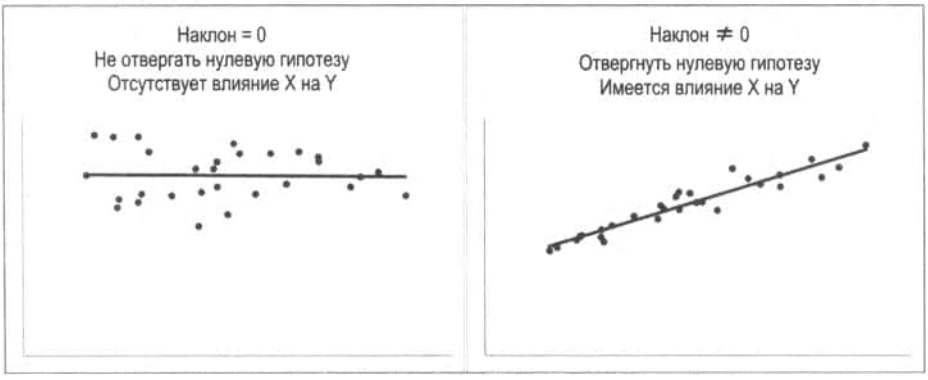


Рис. 4.8. Регрессионные модели со значительным и незначительным наклонами

Линейная регрессия в Excel

В следующем примере линейной регрессии для набора данных **mpg** в Excel мы проверим, оказывает ли вес машины (**weight**) существенное влияние на ее пробег (**mpg**).

Соответственно, наши гипотезы будут выглядеть следующим образом.

H0: линейное влияние веса на пробег отсутствует.

Ha: линейное влияние веса на пробег имеется.

Но сначала запишем уравнение регрессии, используя конкретные искомые переменные (уравнение 4.2).

Уравнение 4.2. *Уравнение регрессии для оценки пробега в милях*

$$\mathbf{mpg} = \beta_0 + \beta_1 \times \mathbf{weight} + \in$$

Начнем с визуализации результатов регрессии. У нас уже есть диаграмма рассеяния (рис. 4.6), и сейчас нам нужно только выполнить наложение на нее («подгонку») линии регрессии. Щелкните на периметре диаграммы, чтобы раскрылось меню **Chart Elements** (Элементы диаграммы). Последовательно выберите **Trendline** (Линия тренда) | **More Options** (Дополнительные параметры). В нижней части меню **Format Trendline** (Формат линии тренда) установите параметр **Display Equation on chart** (Показать уравнение на диаграмме).

Теперь щелкнем на результирующем уравнении, чтобы установить полужирный шрифт и увеличить размер шрифта до 14. Мы сделаем линию тренда сплошной черной линией с шириной 2,5 пункта, щелкнув на ней и затем нажав значок заливки в верхней части меню **Format Trendline** (Формат линии тренда). Мы построили линейную регрессию. Диаграмма рассеяния с линией тренда показана на рис. 4.9. Excel также отображает выражение регрессии из уравнения 4.2, которое мы использовали для оценки пробега в милях с учетом веса автомобиля.

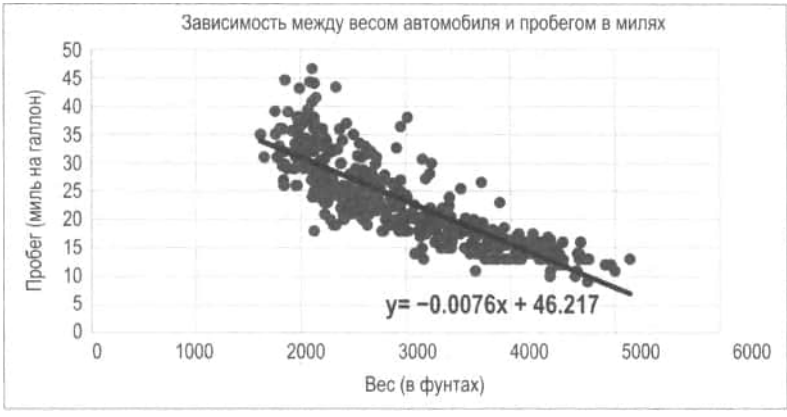


Рис. 4.9. Диаграмма рассеяния с линией тренда и уравнением регрессии для оценки влияния веса на пробег

В нашем уравнении мы можем сделать пересечение перед наклоном, чтобы получить следующее уравнение (4.3).

Уравнение 4.3. *Уравнение регрессии соответствия для оценки пробега*

$$\mathbf{mpg} = 46{,}217 - 0{,}0076 \times \mathbf{weight}$$

Обратите внимание: Excel не включает остаточный член в уравнение регрессии.

Теперь, построив линию регрессии, мы количественно оценили разницу между значениями, которые ожидаем получить из уравнения, и значениями, найденными в данных. Эта разница известна как *остатки* (residual), к ним мы вернемся чуть позже в этой граве. В первую очередь сделаем то, что собирались: установим статистическую значимость.

Замечательно, что Excel настроил для нас линию тренда и выдал результирующее уравнение. Но этой информации недостаточно для выполнения проверки гипотезы: мы все еще не знаем, отличается ли статистически наклон линии от нуля. Чтобы получить эту информацию, снова воспользуемся пакетом инструментов анализа (**Analysis ToolPak**). В верхнем меню выберите последовательно **Data** (Данные) | **Data Analysis** (Анализ данных) | **Regression** (Регрессия). Вам необходимо указать диапазоны **Y** и **X** — это ваши зависимая и независимая переменные соответственно. Не забудьте также указать, что ваши входные данные включают метки (**Labels**), как показано на рис. 4.10.

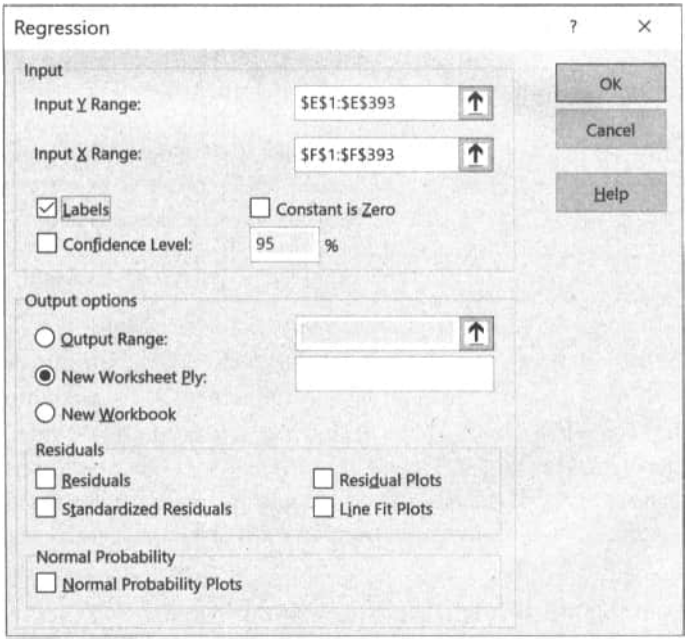


Рис. 4.10. Параметры меню для получения регрессии с помощью **Analysis ToolPak**

В результате мы получим довольно много информации, как показано на рис. 4.11. Давайте пройдемся по ней.

Пропустите пока первый раздел в ячейках `A3:B8` (мы вернемся к нему позже). Второй раздел в ячейках `A10:F14` обозначен как **ANOVA** (analysis of variance), т. е. *дисперсионный анализ*). Он сообщает о том, работает ли лучше регрессия с включенным коэффициентом наклона по сравнению с регрессией только с перехватом[1].

[1] *Перехват* — значение модели, когда все независимые переменные модели равны нулю. Модель «без перехвата» означает, что значение модели для всех независимых переменных равно нулю. — *Ред.*

	A	B	C	D	E	F	G	H	I
1	ИТОГОВЫЕ ВЫХОДНЫЕ ДАННЫЕ								
2									
3	*Regression Statistics*								
4	Multiple R	0.832244215							
5	P Square	0.692630433							
6	Adjusted R Square	0.691842306							
7	Standard Error	4.332712097							
8	Observations	392							
9									
10	Дисперсный анализ (ANOVA)								
11		*df*	*SS*	*MS*	*F*	*Significance F*			
12	Regression	1	16497.75976	16497.75976	878.8308864	6.0153E-102			
13	Residual	390	7321.233706	18.77239412					
14	Total	391	23818.99347						
15									
16		*Coefficients*	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
17	Intercept	46.21652455	0.798672463	57.86668086	1.6231E-193	44.64628231	47.78676679	44.64628231	47.78676679
18	weight	-0.007647343	0.000257963	-29.64508199	6.0153E-102	-0.008154515	-0.00714017	-0.008154515	-0.00714017
19									

Рис. 4.11. Выходные данные регрессии

В табл. 4.2 показано, какие уравнения использовались.

Таблица 4.2. *Сравнение модели без предикторов*[2] *и полной регрессионной модели*

Модель без предикторов (incercept-only model)	**Модель с коэффициентами**
mpg = 46,217	**mpg** = 46,217 – 0,0076 × **weight**

Статистически значимый результат показывает, что коэффициенты действительно улучшили модель. Результаты теста можно определить по p-значению, найденному в ячейке `F12` (см. рис. 4.11). Помните, что это экспоненциальное представление, т. е. следует читать p-значение как 6,01, умноженное на 10 в степени –102, что значительно меньше, чем 0,05. Из этого можно заключить, что вес (**weight**) следует сохранить как коэффициент в данной регрессионной модели.

Это подводит нас к третьему разделу — ячейки `A16:I18`; здесь находится то, что мы изначально искали. Данный диапазон содержит большое количество информации, так что пойдем от столбца к столбцу, начиная с коэффициентов в ячейках `B17:B18`. Они должны выглядеть знакомыми, поскольку это пересечение и наклон линии, которые были даны в уравнении 4.3.

Далее, стандартная ошибка в ячейках `C17:C18`. О ней мы говорили в *главе 3*: это характеристика изменчивости в повторяющихся выборках и в данном случае может рассматриваться как параметр точности наших коэффициентов.

Затем, в ячейках `D17:D18` отображается значение, называемое в Excel «t Stat», иначе известное как t-статистика или t-критерий; эта величина может быть получена

[2] Также известна как *модель только для перехвата* (intercept-only model): предполагает, что вклад переменных-предикторов равен нулю. — *Ред.*

делением коэффициента на стандартную ошибку. Можно сравнить ее с нашим критическим значением, равным 1,96, чтобы установить статистическую значимость с достоверностью 95 %.

Однако чаще интерпретируют и включают в отчеты p-значение, содержащее аналогичную информацию. В нашем распоряжении два p-значения. Первое — коэффициент пересечения в ячейке `E17`. Он говорит о том, действительно ли пересечение значительно отличается от нуля. Значимость пересечения *не является* частью нашей проверки гипотезы, т. е. эта информация нерелевантна в данном случае (еще один удачный пример того, почему не всегда можно принимать на веру выводы Excel).

Хотя большинство статистических инструментов (включая Excel) показывают p-значение пересечения, как правило, эта информация не является релевантной.

Вместо этого нам нужно p-значение переменной **weight** (вес) в ячейке `E18`, связанное с наклоном линии. Величина p-значения значительно меньше 0,05, поэтому мы не можем отвергнуть нулевую гипотезу и сделать вывод, что вес может влиять на пробег. Иными словами, наклон линии значительно отличается от нуля. Так же как в наших предыдущих проверках гипотез, мы не будем делать заключения о том, что «доказали» зависимость или что больший вес *приводит* к меньшему пробегу. Еще раз: мы делаем выводы о совокупности на основе выборки, поэтому неопределенность естественна.

Выходные данные также показывают доверительный интервал перехвата (пересечения) и наклона, равный 95 %, в ячейках `F17:I18`. По умолчанию, он указан дважды: если бы в меню ввода мы запросили еще один доверительный интервал, то получили бы оба.

Теперь, научившись интерпретировать результаты регрессии, попробуем сделать *точечную оценку* на основе линейного уравнения: какой пробег мы ожидаем получить для машины весом 3,021 фунта? Подставим этот вес в уравнение регрессии (уравнение 4.4).

Уравнение 4.4. *Точечная оценка на основе линейного уравнения*

$$\textbf{mpg} = 46{,}217 - 0{,}0076 \times 3021$$

Основываясь на уравнении 4.4, мы ожидаем, что автомобиль весом 3,021 фунта проезжает 23,26 мили на галлон. Посмотрите исходный набор данных: в нем *имеется* наблюдение для веса 3,021 фунта (Ford Maverick, строка 101 в наборе данных) и указаны 18 миль на галлон, а не 23,26. В чем тут дело?

Данное несоответствие представляет собой *остатки*, которые уже упоминались, — это разница между значениями, ожидаемыми в уравнении регрессии и имеющимися в реальных данных. Это и некоторые другие наблюдения показаны на рис. 4.12. Диаграммы рассеяния показывают, какие данные действительно найдены в наборе данных, а линия отражает значения, которые прогнозировались при помощи регрессии.

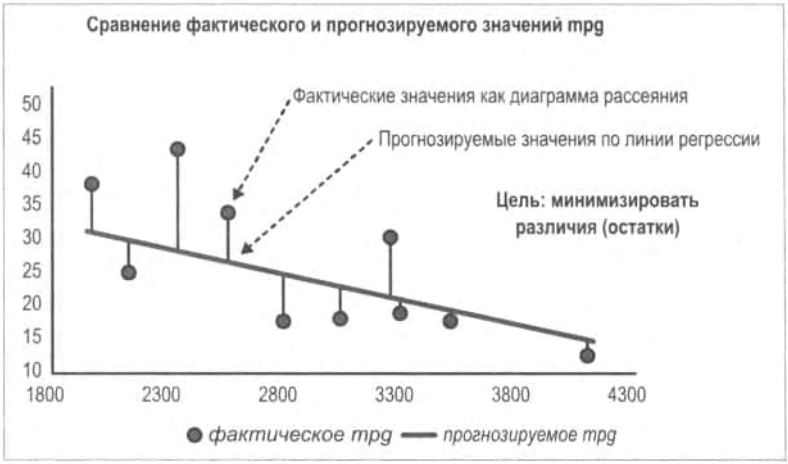


Рис. 4.12. Остатки как разница между фактическими и прогнозируемыми значениями

Отсюда следует, что мы заинтересованы в минимизации разницы между этими значениями. Для этого в Excel и большинстве приложений для расчета регрессии используется *метод наименьших квадратов* (ordinary least squares, OLS). Наша цель использования этого метода — минимизировать остатки, точнее, *сумму квадратов остатков*, так, чтобы негативный и позитивный остатки измерялись одинаково. Чем меньше сумма квадратов остатков, тем меньше разница между фактическими и ожидаемыми значениями и тем лучше уравнение регрессии делает оценку.

Из p-значения наклона мы узнали, что имеется существенная зависимость между независимой и зависимой переменными. Но это не говорит о том, *насколько* изменчивость зависимой переменной объясняется независимой переменной.

Помните: изменчивость — это то, что́ изучают аналитики; переменные различаются, и мы хотим знать, *почему* они различаются. Эксперименты позволяют понять зависимости между независимой и зависимой переменными. Но мы не способны объяснить все касательно зависимой переменной только на основании независимой переменной. Всегда будет присутствовать некоторая необъяснимая ошибка.

R-квадрат (R-squared), или *коэффициент детерминации* (в Excel обозначается как **R-square**), показывает в процентах, какая доля изменчивости зависимой переменной объясняется регрессионной моделью. Например, R-квадрат, равный 0,4, показывает, что 40 % изменчивости **Y** может быть объяснено данной моделью. Это означает, что 1 минус R-квадрат — это изменчивость, которая *не может* объясняться моделью. Если R-квадрат равен 0,4, тогда 60 % изменчивости **Y** не учитывается.

Excel вычислил R-квадрат в первой секции выходных данных регрессии (ячейка B5 на рис. 4.11). Квадратный корень из значения R-квадрат равен множественному R, отображаемому в ячейке B4 выходных данных. Адаптированный R-квадрат (ячейка B6) используется как более осторожная оценка R-квадрата для модели с несколькими независимыми переменными. Этот критерий представляет интерес в случае вы-

числения *множественной линейной регрессии*, рассмотрение которой выходит за рамки данной книги.

Множественная линейная регрессия

В этой главе рассматривается *одномерная линейная регрессия*, т. е. влияние одной независимой переменной на одну зависимую переменную. Также можно построить *множественную*, или многофакторную, *регрессионную модель* для оценки влияния нескольких независимых переменных на зависимую переменную. Независимые переменные могут включать категориальные, а не только непрерывные переменные, взаимодействия переменных и многое другое. Для получения подробной информации о более сложной линейной регрессии в Excel обратитесь к книге Конрада Карлберга «Регрессионный анализ в Microsoft Excel»[3] (Conrad Carlberg. Regression Analysis Microsoft Excel).

Существуют и другие методы оценки результатов регрессии, не только R-квадрат, и Excel включает один из них: стандартная ошибка регрессии в выходных данных (ячейка B7 на рис. 4.11). Это показатель среднего расстояния отклонения наблюдаемых величин от линии регрессии. Некоторые аналитики предпочитают R-квадрату этот и другие методы для оценки регрессионной модели, хотя R-квадрат остается главным из них. Независимо от предпочтений, наилучший результат часто получается при оценивании нескольких элементов в соответствующем контексте, так что нет необходимости слепо доверять или отказываться от того или иного одного метода.

Поздравляю! Вы произвели и интерпретировали полный регрессионный анализ.

Переосмысление результатов: ложные зависимости

Основываясь на временно́м упорядочивании и собственной логике, в примере с пробегом легко прийти к выводу, что переменная **weight** (вес) должна быть независимой, а переменная **mpg** (пробег в милях) — зависимой. Но что произойдет, если мы построим линию регрессии, поменяв эти переменные местами? Давайте попробуем сделать это с помощью пакета инструментов анализа (Analysis ToolPak). Получившееся уравнение регрессии показано ниже (уравнение 4.5).

***Уравнение 4.5**. Уравнение регрессии для оценки веса на основе пробега в милях*

$$\textbf{weight} = 5101{,}1 - 90{,}571 \times \textbf{mpg}$$

Можно поменять местами независимую и зависимую переменные и получить тот же самый коэффициент корреляции. Но если изменить их в регрессии, *коэффициенты изменятся*.

Если бы обнаружилось, что на обе переменные, **mpg** и **weight**, одновременно оказывает влияние какая-то внешняя переменная, ни одна из этих моделей не была бы правильной. Это тот же сценарий, который мы наблюдаем в случае с потреблением

[3] Издана в России. — *Ред.*

мороженого и атаками акул. Смешно говорить, что потребление мороженого хоть как-то влияет на атаки акул только потому, что обе переменные зависят от температуры, как показано на рис. 4.13.

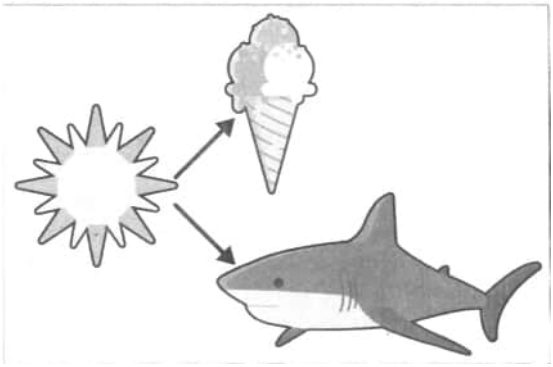


Рис. 4.13. Употребление мороженого и атаки акул: ложная зависимость

Такая зависимость называется *ложной*. Она часто встречается в данных и не всегда так очевидна, как в приведенном примере. Знание данных изучаемой предметной области может быть неоценимым для выявления ложных связей.

Переменные могут быть коррелированными, может даже существовать доказательство причинно-следственной связи. Но причиной зависимости может быть переменная, которую вы даже не учли.

Заключение

Помните выражение:

> «Корреляция не подразумевает причинно-следственную связь»?

Аналитика в высшей степени инкрементна: обычно одна концепция накладывается поверх другой, чтобы выполнять все более сложный анализ. Например, мы всегда начинаем с описательной статистики выборки, прежде чем вывести параметры генеральной совокупности (популяции). Хотя корреляция может не подразумевать причинно-следственную связь, тем не менее эта связь базируется на принципах корреляции. Следовательно, более подходящим выражением о корреляции будет такое:

> Корреляция — это необходимое, но недостаточное условие причинно-следственной связи.

В этой и предыдущих главах мы только слегка коснулись поверхности инференциальной статистики. Существует огромное множество тестов, но все они основаны на тех же принципах *проверки гипотез*, которые мы использовали здесь. Овладейте этим процессом, и вы сможете протестировать все виды зависимостей данных.

Переход к программированию

Надеюсь, вы убедились и согласитесь с тем, что Excel — превосходный инструмент для изучения статистики и аналитики. Вы на практике познакомились со статистическими принципами, лежащими в основе этой книги, и узнали, как изучать и проверять зависимости в реальных наборах данных.

В то же время эффективность Excel снижается, когда речь заходит о более глубокой аналитике. Например, мы проверяли такие свойства, как нормальность и линейность с помощью визуализации; это хорошо для начала, но существуют более надежные способы проверить их (часто с помощью статистического вывода). Эти методы часто опираются на матричную алгебру и различные операции с интенсивными вычислениями, выполнение которых в Excel может быть трудоемким. Хотя существуют надстройки, чтобы компенсировать эти недостатки, они могут быть дорогими и не иметь нужных функций. Вместе с тем есть инструменты с открытым кодом, как языки R и Python, которые являются бесплатными и содержат множество функций, подобных приложениям и называемых *пакетами*, которые подходят практически для любого сценария использования. Эта среда поможет сосредоточиться на концептуальном анализе данных, а не заниматься примитивными вычислениями, но вам придется научиться программировать. Этим инструментам, а также аналитическим инструментам в целом посвящена *глава 5*.

Упражнения

Потренируйтесь в корреляции и регрессии, проанализировав набор данных **ais**, расположенный в папке **datasets** репозитория GitHub. В наборе данных представлены показатели роста, веса и крови австралийских спортсменов различных видов спорта мужского и женского пола.

Выполните следующие действия, используя этот набор данных.

1. Создайте матрицу корреляции значимых переменных в наборе данных.
2. Визуализируйте зависимость между переменными **ht** (рост) и **wt** (вес). Является ли зависимость линейной? Если да, то какая это зависимость — негативная или позитивная?
3. Какая из переменных, **ht** или **wt**, является независимой, а какая — зависимой, по вашему мнению?
   - Имеет ли место существенное влияние независимой переменной на зависимую?
   - Каков наклон получившейся линии регрессии?
   - Какой процент дисперсии зависимой переменной объясняется независимой переменной?
4. В наборе данных содержится переменная **bmi** (body mass index — индекс массы тела). Если вы не знакомы с этим показателем, найдите время, чтобы выяснить, как он вычисляется. Попробуйте проанализировать зависимость между переменными **ht** и **bmi**. Старайтесь полагаться на здравый смысл, а не только на статистические результаты.

ГЛАВА 5

Стек анализа данных

К этому моменту вы уже должны быть хорошо знакомы с ключевыми принципами и методами аналитики, изученными их в Excel. Эта глава служит введением к следующим частям книги, в которых вы будете применять полученные знания в языках R и Python.

Мы продолжим изучать принципы и методы статистики, анализа и обработки данных, а также рассмотрим Excel, R и Python в контексте того, что в аналитике называется *стеком анализа данных*.

Статистика, аналитика и наука о данных

Цель этой книги — помочь вам освоить принципы анализа данных. Но, как вы могли убедиться, статистика настолько важна для аналитики, что часто бывает трудно определить, где заканчивается одна область и начинается другая. Вас также, возможно, заинтересует, как в общую картину вписывается наука о данных. Давайте попробуем разграничить эти понятия.

Статистика

Статистика в первую очередь связана с методами сбора, анализа и представления данных. Мы *многое* почерпнули из этой области: например, делали выводы о генеральной совокупности на основе выборки, представляли распределения и зависимости в данных с помощью гистограмм и диаграмм рассеяния.

Большинство технологий и методов, таких как линейная регрессия и t-тест для независимых выборок, которые мы использовали до сих пор, пришли из статистики. То, что отличает аналитику данных от статистики, является скорее *целью*, а не *средством*.

Аналитика данных

В аналитике данных внимание акцентируется на использовании полученных результатов для достижения некоторых внешних целей, на методах анализа. Результаты могут быть разными: например, хотя некоторые зависимости являются статистически значимыми, для бизнеса они могут не иметь существенного значения.

Аналитика данных также связана с технологиями, необходимыми для ее реализации. Например, может потребоваться очистить наборы данных, разработать ин-

формационные панели, а также быстро и эффективно развернуть эти ресурсы. Хотя основное внимание в этой книге уделяется *статистическим* основам аналитики, существуют также другие, *вычислительные* и *технологические* основы, которые необходимо знать, и мы рассмотрим их далее в этой главе.

Бизнес-аналитика

По сути, аналитика данных используется для достижения бизнес-целей и оказания помощи заинтересованным сторонам; профессионалы в аналитике часто одной ногой стоят в мире коммерческих операций, а другой — в мире информационных технологий. Термин «*бизнес-аналитика*» (business intelligence, BI) часто используется для описания комбинации этих областей.

Примером аналитики данных или бизнес-аналитики может быть анализ проката фильмов. Используя разведочный анализ данных, аналитик может предположить, что комедии особенно хорошо продаются в праздничные и выходные дни. Взаимодействуя с менеджерами проекта или другими заинтересованными людьми, можно провести небольшие эксперименты для сбора данных и проверки этой гипотезы. Элементы этого процесса должны быть вам знакомы из предыдущих глав.

Наука о данных

Наконец, существует наука о данных: еще одна область, неразрывно связанная со статистикой, но наибольшее внимание здесь уделяется уникальным результатам.

Специалисты по анализу и обработке данных также учитывают в работе бизнес-цели, но их сфера деятельности значительно отличается от аналитики данных. Вернемся к примеру с прокатом фильмов: специалист по анализу и обработке данных может построить алгоритмическую систему для рекомендации фильмов отдельным лицам на основе предпочтений подобных им клиентов. Создание и развертывание такой системы требует серьезных инженерных навыков. Хотя несправедливо говорить, что специалисты по анализу и обработке данных не имеют реальных связей с бизнесом, тем не менее часто они больше связаны с техническими науками и информационными технологиями, чем их коллеги — аналитики данных.

Машинное обучение

Итак, основное различие между вышеописанными направлениями заключается в следующем: в то время как аналитика данных занимается *описанием* и *объяснением* зависимостей, наука о данных посвящена созданию *прогнозирующих* систем и продуктов, часто с использованием методов машинного обучения.

Машинное обучение — это методика построения алгоритмов, совершенствующихся с помощью большого количества данных, не будучи для этого специально запрограммированными. Например, в банке можно установить машинное обучение, чтобы определить возможность выполнения клиентом обязательств по кредиту. По мере сбора большего количества данных алгоритм может найти закономерности и

зависимости в данных и использовать их для более точного прогнозирования вероятности выполнения или невыполнения обязательств. Модели машинного обучения могут быть невероятно эффективны для прогнозирования и использоваться в самых различных сценариях. Иногда бывает весьма заманчиво построить сложный алгоритм машинного обучения, тогда как достаточно простого, однако это может привести к осложнениям, связанным с интерпретацией и доверием к модели.

Машинное обучение выходит за рамки этой книги; фантастический обзор этой темы можно найти в книге Орельена Жерона «Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow»[1] (Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow). Эта книга в значительной степени написана для языка Python, поэтому лучше сначала закончить изучение *части III* нашей книги.

Различия без взаимоисключения

Хотя различия между статистикой, аналитикой данных и наукой о данных достаточно велики, не стоит возводить между ними невидимые границы. В любой из этих дисциплин имеет значение разница между категориальной и непрерывной зависимой переменными. Во всех них при формулировании проблемы используется проверка гипотез. За эту общую лексику при работе с данными нужно благодарить статистику.

Роли аналитика данных и специалиста по обработке данных также часто переплетаются. По сути, в настоящей книге вы изучали основы фундаментальной науки о данных: линейную регрессию. Все эти области связывает значительно большее, чем то, что их разделяет. Хотя книга посвящена аналитике данных, вы будете готовы разобраться во всех названных областях, особенно после того, как изучите языки R и Python.

Сейчас, когда мы контекстно увязали аналитику со статистикой и наукой о данных, следует сделать то же самое с Excel, языками R и Python, а также прочими инструментами, которые нужно будет изучить в аналитике.

Значение стека анализа данных

Прежде чем технически овладеть каким-то одним инструментом, профессионал-аналитик должен уметь выбирать и сочетать *различные* методы и инструменты с учетом сильных и слабых сторон каждого из них.

Веб-разработчики и администраторы баз данных обычно имеют собственный «стек» инструментов, используемых для работы. Мы можем использовать эту же идею в аналитике данных. Когда один инструмент или «блок» стека оказывается неэффективным, не имеет смысла искать виновных в этом, надо просто выбрать другой блок или блоки, т. е. воспринимать разные блоки стека как дополняющие, а не заменяющие друг друга.

---

[1] Издана в России. — *Ред.*

На рис. 5.1 показана моя концепция четырех блоков стека анализа данных. Это значительно упрощенное представление инструментов работы с данными, используемых в организациях, — отображение всех аналитических связей может выглядеть слишком сложным. Блоки расположены по порядку, начиная с тех, в которых данные хранятся и обслуживаются отделами информационных технологий (баз данных), и заканчивая теми, где данные применяются и изучаются конечными пользователями (электронные таблицы). Любые из этих блоков могут участвовать в формировании решений совместно в различных сочетаниях.

Рассмотрим каждый блок этого стека, начиная с тех, которые я считаю наиболее знакомыми обычному читателю, и заканчивая наименее знакомыми.

Рис. 5.1. Стек анализа данных

Электронные таблицы

Я не буду тратить слишком много времени на разъяснение сущности и функций электронных таблиц — к этому моменту вы с ними уже достаточно хорошо знакомы. Те же самые принципы применяются и к другим приложениям для работы с электронными таблицами, таким как Google Sheets, LibreOffice и т. д. Поскольку в этой книге мы использовали Excel, сосредоточимся на нем. Вы уже убедились, что электронные таблицы способны визуализировать аналитику и являются отличным инструментом для разведочного анализа данных. Простота в использовании и гибкость делают электронные таблицы прекрасным средством для дистрибуции конечным пользователям.

Но эта гибкость может быть и достоинством, и недостатком. Приходилось ли вам когда-либо создавать табличную модель, в которой вы получали какое-то число, а затем, просто открыв файл спустя несколько часов, увидеть другое число? Иногда возникает ощущение, что вы играете с электронными таблицами в игру «Побей крота»[2], — очень сложно изолировать один слой анализа от других.

[2] «Побей (ударь, убей) крота» (Whack-a-mole) — компьютерная игра, где нужно бить молотком по кротам, — как только ударяешь одного, сразу появляется другой. — *Ред.*

Хорошо спроектированный результат обработки данных выглядит примерно так, как показано на рис. 5.2.

- Исходные данные расположены отдельно и не касаются анализа.
- Данные обрабатываются для соответствующей очистки и анализа.
- Любые результирующие диаграммы или таблицы являются изолированными выходными данными.

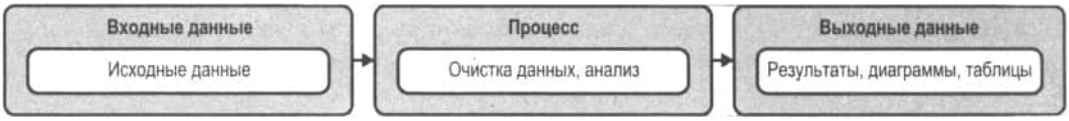


Рис. 5.2. Входные данные, процесс, выходные данные

Хотя в таком подходе к электронным таблицам прослеживаются определенные принципы, эти уровни имеют вероятность превратиться в мешанину: пользователи могут перезаписывать исходные данные или строить вычисления одно поверх другого до такой степени, что становится очень трудно отслеживать ссылки, указывающие на ту или иную ячейку. Даже имея надежную архитектуру рабочей книги, сложно достичь конечной цели модели «вход — процесс — выход» — *воспроизводимости*. Идея тут заключается в том, что одни и те же входные данные и процессы обработки должны раз разом приводить к одним и тем же результатам. Очевидно, что рабочая книга не может быть воспроизводимой, если из-за ошибочных шагов, небрежных вычислений и т. п. нет гарантии получения одного и того же результата при каждом последующем открытии файла.

Беспорядочные или невоспроизводимые рабочие книги породили «страшные» истории в различных областях, начиная от общественного питания и до финансового регулирования: для ознакомления почитайте такие истории на портале Европейской группы по управлению рисками, связанными с электронными таблицами (European Spreadsheet Risks Interest Group, EUSpRig), которая проводит ежегодные конференции. Возможно, анализ, который вы выполняете, не так важен, как торговля акциями или публикация инновационных научных исследований, но никому не нравятся медленные, подверженные ошибкам процессы с ненадежными результатами. Но довольно мрачных предсказаний; как я постоянно подчеркиваю, Excel и другие электронные таблицы занимают свое законное место в аналитике. Давайте рассмотрим некоторые инструменты, позволяющие создавать четкие, воспроизводимые рабочие процессы в Excel.

VBA

Как вы увидите далее, обычно воспроизводимость вычислений достигается с помощью записи каждого шага в виде кода, который можно сохранить и быстро повторно выполнить позже. Excel действительно имеет свой встроенный язык программирования — *Visual Basic для приложений* (Visual Basic for Applications, VBA).

Хотя VBA действительно позволяет записывать процесс в виде кода, в нем недостает многих функций полноценного языка статистического программирования:

в частности, многих бесплатных пакетов для специализированного анализа. Более того, компания Microsoft почти полностью отказалась от VBA, перенаправив ресурсы на свой новый «язык сценариев (скриптов)» (Office Scripts language), являющийся встроенным средством автоматизации Excel, а также на JavaScript и, если верить слухам, Python.

Современный Excel

Я буду использовать термин «современный Excel» (modern Excel) применительно к серии инструментов, ориентированных на бизнес-аналитику (BI), которые компания Microsoft начала включать в Excel, начиная с Excel 2010. Эти инструменты необычайно мощные и интересные в использовании, к тому же они разрушают многие мифы о том, что́ может и чего не может Excel. Давайте взглянем на три приложения, из которых состоит современный Excel.

- Power Query — это инструмент для извлечения данных из различных источников, их преобразования и последующей загрузки в Excel. Эти источники данных могут варьироваться от файлов **.csv** до реляционных баз данных и содержать многие миллионы записей: хотя сама по себе рабочая книга Excel содержит до миллиона строк, при использовании Power Query это число может увеличиться в несколько раз.
- Более того, Power Query *полностью воспроизводится* благодаря языку программирования Microsoft M. Пользователи могут добавлять и редактировать шаги с помощью меню, которое генерирует M-код, или писать его самостоятельно. Power Query — потрясающее продвижение для Excel; он не только устраняет существовавшие ранее ограничения относительно количества данных, которые могут проходить через рабочую книгу, но и делает извлечение и обработку этих данных полностью воспроизводимыми.
- Power Pivot — инструмент для реляционного моделирования данных в Excel. Реляционные модели данных мы обсудим позже в этой главе, когда будем рассматривать базы данных.
- Power View — инструмент для создания интерактивных диаграмм и визуализаций в Excel. Он особенно полезен при работе с информационными панелями.

Я настоятельно рекомендую вам найти время на изучение современного Excel, особенно если ваша работа связана с проведением анализа и подготовкой отчетности. Благодаря этим инструментам многие отрицательные отзывы об Excel, включая его неспособность обрабатывать более миллиона строк или работать с разными источниками данных, уже не соответствуют действительности.

Тем не менее эти инструменты разработаны не только для выполнения статистического анализа, но и для других целей аналитики, таких как построение отчетов и распространение данных. К счастью, существует масса возможностей для сочетания Power Query и Power Pivot с такими инструментами, как R и Python, для создания исключительных информационных продуктов.

Несмотря на очевидное развитие и огромные преимущества Excel, многие аналитики относятся к нему критически из-за неприятностей, к которым может привести

злоупотребление им. Это вызывает вопрос: *почему приложением Excel злоупотребляют чаще всего*? Да потому, что бизнес-пользователи, за недостатком лучших альтернатив и ресурсов, видят в нем интуитивно понятное, гибкое пространство для хранения и анализа данных.

В самом деле, если не можешь справиться с чем-то, присоединяйся: Excel действительно отличный инструмент для изучения данных и взаимодействия с ними. Благодаря новейшим функциям он стал замечательным средством для построения воспроизводимых процессов очистки данных и реляционных моделей данных. Но все же есть аналитические функции, в которых Excel не настолько хорош, например хранение критически важных данных, распространение информационных панелей и отчетов на нескольких платформах и выполнение расширенного статистического анализа. Для этих целей существуют альтернативные варианты, которые мы сейчас рассмотрим.

Базы данных

Базы данных, точнее, *реляционные* базы данных, — это относительно старые технологии в мире аналитики, происхождение которых восходит к началу 1970-х годов. Вы уже встречались с основой реляционных баз данных — это таблица. Пример таблицы показан на рис. 5.3: мы называли столбцы и строки такой таблицы статистическими терминами «переменные» и «наблюдения». На языке баз данных их аналогами являются *поля* и *записи*.

Dept_no	Dept_name	Loc_no
1	Финансы	5
2	Маркетинг	5
3	Информационные технологии	6
4	Управление персоналом	5

→ Столбец, переменная, *поле*

→ Строка, наблюдение, *запись*

Рис. 5.3. Заполненная таблица базы данных

Если требуется связать данные вместе, можно воспользоваться функцией Excel **VLOOKUP()**, используя общие столбцы как «поля поиска» для передачи данных из одной таблицы в другую (рис. 5.4). Это далеко не все возможности, но суть *реляционных моделей данных* в том, чтобы использовать связи между данными в таблицах для эффективного хранения и управления. Я называю функцию `VLOOKUP()` «клейкой лентой Excel» из-за ее способности связывать наборы данных. Если функция `VLOOKUP()` — это клейкая лента, то реляционные модели данных — «сварщики».

Система управления реляционными базами данных (СУБД) (relational database management system, RDBMS) предназначена для использования этой базовой концепции для хранения и извлечения больших наборов данных. Когда вы размещаете заказ в магазине или подписываетесь на рассылку, эти данные, скорее всего, пропускаются через СУБД. Хотя Power Pivot построен на тех же концепциях, он больше подходит для бизнес-анализа и построения отчетов, а не для полнофункциональной СУБД.

Empno	Ename	Deptno	Sal
805	Адамс	2	1600
210	Кларк	1	1500
879	Джонс	4	1100

Loc_no	City	State	Zip
5	Кливленд	Огайо	44106
6	Денвер	Колорадо	80205

Dept_no	Dept_name	Loc_no
1	Финансы	5
2	Маркетинг	5
3	Информационные технологии	6
4	Управление персоналом	5

Рис. 5.4. Связи между полями и таблицами в реляционной базе данных

Язык структурированных запросов, или *SQL* (Structured Query Language), традиционно используется для взаимодействия с базами данных. Это исключительно важная тема в аналитике, но она выходит за рамки нашей книги; чтобы получить о ней представление, прочтите книгу Алана Болье «Изучаем SQL»[3] (Alan Beaulieu. Learning SQL). Имейте в виду, что хотя SQL (или «сиквел») часто используется как общее название языка, существует несколько его «диалектов», в зависимости от СУБД. Некоторые из них, такие как Microsoft или Oracle, являются защищенными, другие, такие как PostgreSQL или SQLite, имеют открытый исходный код.

Классическая аббревиатура для операций, которые может выполнять язык SQL, — CRUD: Create (создать), Read (читать), Update (обновить) и Delete (удалить). Как аналитик данных, вы, скорее всего, будете *читать* данные из базы, а не изменять их. Для подобных операций разница в диалектах SQL на разных платформах ничтожна.

Платформы бизнес-аналитики (BI)

Этот, по общему признанию, обширный набор инструментов, вероятно, наиболее неоднозначный блок стека. Я имею в виду корпоративные инструменты, позволяющие пользователям собирать, моделировать и отображать данные. Инструменты для хранения данных, такие как MicroStrategy и SAP BusinessObjects, являются здесь лидерами, поскольку предназначены для самостоятельного сбора и анализа данных. Но зачастую они отличаются ограниченной визуализацией и интерактивной информационной панелью.

На помощь приходят такие инструменты, как Power BI, Tableau и Looker. Эти платформы (почти все они с закрытым исходным кодом) позволяют пользователям создавать модели данных, информационные панели и отчеты с минимальным кодом. Важно и то, что они упрощают процесс распространения и обновления информации на всем предприятии. Эти ресурсы часто могут развертываться в разных

[3] Издана в России. — *Ред.*

форматах на планшетах и смартфонах. Многие организации перенесли свои программы для отчетности и создания информационных панелей из электронных таблиц в эти инструменты бизнес-аналитики.

Несмотря на все преимущества, платформы бизнес-аналитики обычно не обладают гибкостью в обработке и визуализации данных. Их основная цель — быть простыми и понятными для бизнес-пользователей, но сложными для взлома, при этом они часто не имеют достаточного набора функций, которые требуются опытным аналитикам данных для эффективного решения их задач. Кроме того, они бывают довольно дороги и могут предоставлять годовые пользовательские лицензии, которые стоят сотни или даже тысячи долларов. Здесь важно отметить, что элементы современного Excel (Power Query, Power Pivot и Power View) также доступны для набора программ Power BI. Более того, можно создавать визуализации в Power BI с помощью кода R и Python. Другие системы бизнес-аналитики обладают теми же свойствами. Я обратил особое внимание на инструментарии Power BI только потому, что ранее мы сосредоточились на Excel.

Языки программирования для анализа данных

Итак, мы подошли к последнему блоку стека анализа данных: языки программирования. Под ними я подразумеваю скриптовые приложения, используемые специально для аналитики данных. Многие специалисты по аналитике выполняют потрясающую работу, не используя этот блок стека. Более того, многие профессиональные утилиты для сложной аналитики переходят на решения с минимальным кодом или без него.

Принимая во внимание все сказанное, я настоятельно рекомендую вам научиться писать код. Это даст вам более глубокое понимание процессов обработки данных и позволит более полно управлять рабочим процессом, чем применение графического интерфейса пользователя, ГИП (graphical user interface, GUI), или интерактивного программного обеспечения.

Для аналитики данных подходят два языка программирования с открытым кодом: R и Python, которым посвящена остальная часть этой книги. Каждый включает в себя невообразимое множество бесплатных пакетов, предназначенных для любых целей, начиная с автоматизации социальных сетей до геопространственного анализа. Изучение этих языков открывает двери в расширенную аналитику и науку о данных. Если вы считаете, что Excel — это мощный инструмент для исследования и анализа данных, подождите, пока не освоите языки R и Python.

Кроме того, эти инструменты идеально подходят для воспроизводимого исследования. Вспомните рис. 5.2 и сложности, с которыми мы столкнулись при разделении шагов исследовательского процесса в Excel. Являясь языками программирования, R и Python умеют записывать все шаги, предпринятые в процессе анализа. Такая последовательность действий оставляет нетронутыми исходные данные, сначала считывая их из внешних источников, а *затем* работая с копией данных. Это также упрощает отслеживание изменений и дополнений в файлы посредством процесса, известного как *контроль версий*, о котором мы будем говорить в *главе 14*.

R и Python — это языки программирования с открытым исходным кодом; их исходный код находится в свободном доступе для любого, кто хочет изучать, развивать, распространять или вносить в код какой-либо вклад. Это значительно отличает R и Python от Excel, являющегося защищенным продуктом. Системы с открытым исходным кодом, как и закрытые, имеют свои преимущества и недостатки. В случае с R и Python возможность свободно разрабатывать исходный код привела к созданию целой «экосистемы» пакетов и приложений. Это также снизило барьер для привлечения новичков.

В то же время зачастую критически важные части инфраструктуры с открытым исходным кодом поддерживаются разработчиками в свободное время без какой-либо компенсации. Возможно, не следует полагаться на постоянное развитие и поддержку инфраструктуры, не имеющей коммерческой защиты. Этот риск можно снизить: многие компании фактически существуют исключительно для поддержки, сопровождения и развития систем с открытым исходным кодом. Вы сможете увидеть эту связь далее, когда мы будем обсуждать R и Python; возможно, вас удивит, что зарабатывать деньги можно, предоставляя исключительно услуги на основе свободно доступного кода.

Блок стека «языки программирования для анализа данных», вероятно, является самым сложным среди всех: в конце концов, это подразумевает буквально изучение нового языка. Изучение *одного* такого языка — это уже довольно напряженно, тогда как и зачем учить сразу два?

Прежде всего, как уже говорилось в начале этой книги, вы начинаете не с нуля. Вы владеете достаточными знаниями о программировании и работе с данными. Поэтому отнеситесь спокойно к тому, что вы *научились* программировать... Примерно так.

Быть полиглотом в языках программирования так же выгодно, как и в случае с обычными языками. С практической точки зрения работодатели могут использовать любой из этих языков, так что разумно учесть оба варианта. Но вы не просто ставите галочку, изучая оба языка: каждый из них имеет свои уникальные особенности, и вам может быть проще использовать тот или иной язык для каждого конкретного случая. Как и с разными блоками стека, разумно рассматривать инструменты одного блока как дополняющие, а не заменяющие друг друга.

Заключение

Аналитики данных часто задумываются над тем, какие инструменты им следует изучить и какими овладеть в совершенстве. Я бы посоветовал не стараться стать экспертом по какому-то одному инструменту, а изучать различные инструменты из каждого блока стека настолько, чтобы грамотно и с пониманием ситуации выбирать между ними. С этой точки зрения нет смысла заявлять, что один блок стека уступает другому. Они должны дополнять, а не заменять друг друга.

В действительности множество наиболее мощных аналитических инструментов происходят от *комбинации* различных блоков стека. Например, можно использовать Python для автоматизации создания отчетов на основе Excel или для выгрузки

данных из СУБД в панель управления платформы BI (бизнес-аналитики). Хотя эти варианты использования выходят за рамки данной книги, вывод таков: *не следует игнорировать Excel*. Это важный блок стека технологий, который будет только дополнен вашими навыками в языках R и Python.

В этой книге мы уделяем основное внимание электронным таблицам (Excel) и языкам программирования для анализа данных (R и Python). Эти инструменты особенно подходят для способов анализа данных на основе статистики, которые, как мы видели, частично совпадают с традиционной статистикой и наукой о данных. Также мы обсуждали, что аналитика включает в себя больше, чем просто статистический анализ, — и реляционные базы данных, и инструменты бизнес-аналитики могут быть полезны для решения таких задач. Изучив эту книгу, подумайте о дополнении своих знаний о стеке анализа данных темами, представленными в настоящей главе.

Что будет дальше

Получив общее представление об аналитике и приложениях для анализа данных, мы перейдем к более глубокому изучению новых инструментов.

Начнем с языка R, потому что, как мне кажется, для пользователей Excel это более естественная стартовая площадка в программирование данных. Вы узнаете, как осуществлять большую часть того же разведочного анализа данных и проверку гипотез с помощью R, что и в Excel. Это позволит вам выполнять более глубокую аналитику. Затем то же самое вы научитесь делать с помощью Python. На каждом этапе пути я буду помогать связывать новые знания с уже имеющимися, — и вы увидите, насколько вам знакомы многие концепции. До встречи в *главе 6*!

Упражнения

Эта глава имеет скорее концептуальный, чем прикладной характер, поэтому в ней нет упражнений. Советую вам возвращаться к ней по мере того, как вы знакомитесь с другими областями аналитики и связываете их друг с другом. Когда вы сталкиваетесь с новым инструментом работы с данными в своей деятельности, при просмотре социальных сетей или тематических публикаций, спросите себя, к какому блоку стека анализа данных он относится, имеется ли у него открытый исходный код и т. д.

ЧАСТЬ II

От Excel к R

ГЛАВА 6

Первые шаги в R для пользователей Excel

В *главе 1* вы научились выполнять разведочный анализ данных в Excel. Там же упоминалось, что популяризацию методики разведочного анализа данных приписывают Джону Тьюки. Подход Тьюки к данным вдохновил разработку нескольких языков статистического программирования, включая язык S в легендарных лабораториях Белла (Bell Laboratories). Язык S, в свою очередь, вдохновил разработку языка R. Название языка R, разработанного в начале 1990-х Россом Айхэкой (Ross Ihaka) и Робертом Джентльменом (Robert Gentleman), происходит от первых букв имен его авторов, а также отсылает к его происхождению от языка S. R имеет открытый исходный код и поддерживается организацией «Фонд R для статистических вычислений» (R Foundation for Statistical Computing). Разработанный прежде всего для статистических вычислений и графики, язык R наиболее популярен среди исследователей, статистиков и специалистов по анализу данных.

Язык R был разработан специально для статистического анализа.

Загрузка R

Чтобы начать работу, перейдите на сайт R Project. Для загрузки языка R щелкните ссылку в верхней части страницы. Вам предложат выбрать зеркало в *комплексной R-архивной сети* (Comprehensive R Archive Network, CRAN). Это сеть серверов для распространения исходного кода R, пакетов и документации. Выберите зеркало рядом для загрузки R для вашей операционной системы.

Начало работы с RStudio

Итак, вы установили язык R, но чтобы оптимизировать процесс написания кода, необходимо выполнить еще одну загрузку. Из *главы 5* вы знаете, что если программное обеспечение имеет открытый исходный код, любой может его развивать, распространять или вносить в него какой-либо вклад. Например, для работы с кодом поставщики могут предложить IDE (integrated development environment) — *интегрированную среду разработки*. IDE RStudio сочетает под единым интерфейсом инструменты для редактирования кода, графические средства, документацию и многое другое. За десять лет пребывания на рынке она стала основной средой раз-

работки для программирования на R, в которой пользователи создают всё, начиная от интерактивных информационных панелей (Shiny) до аналитических отчетов (R Markdown).

Вас может заинтересовать, зачем устанавливать R, если *RStudio является таким замечательным инструментом*? На самом деле это две различные загрузки: мы загрузили R в качестве *базы кода*, а RStudio — как интегрированную среду разработки (IDE) *для работы с кодом*. Такое разделение приложений может быть незнакомо вам как пользователю Excel, но в мире программного обеспечения с открытым исходным кодом это весьма распространено.

RStudio представляет собой платформу для *работы с кодом* на языке R, но не саму базу кода. Сначала необходимо загрузить язык R из репозитория CRAN (Comprehensive R Archive Network), а затем загрузить RStudio.

Для загрузки RStudio перейдите на страницу загрузки на ее веб-сайте. Вы увидите, что RStudio предлагается по гибкой системе цен; выберите бесплатную версию **RStudio Desktop** (RStudio — отличный пример создания прибыльного бизнеса на базе программного обеспечения с открытым исходным кодом). RStudio вам очень понравится, но вначале может показаться слишком большой и запутанной из-за обилия панелей и функций. Чтобы преодолеть начальный дискомфорт, совершим ознакомительный тур.

Сначала войдите в главное меню и выберите **File** (Файл) | **New File** (Новый файл) | **R Script**. Вы увидите окно, подобное изображенному на рис. 6.1. На нем имеется множество необязательных дополнительных функций; идея IDE RStudio (интегрированная среда разработки RStudio) состоит в том, чтобы все инструменты, необ-

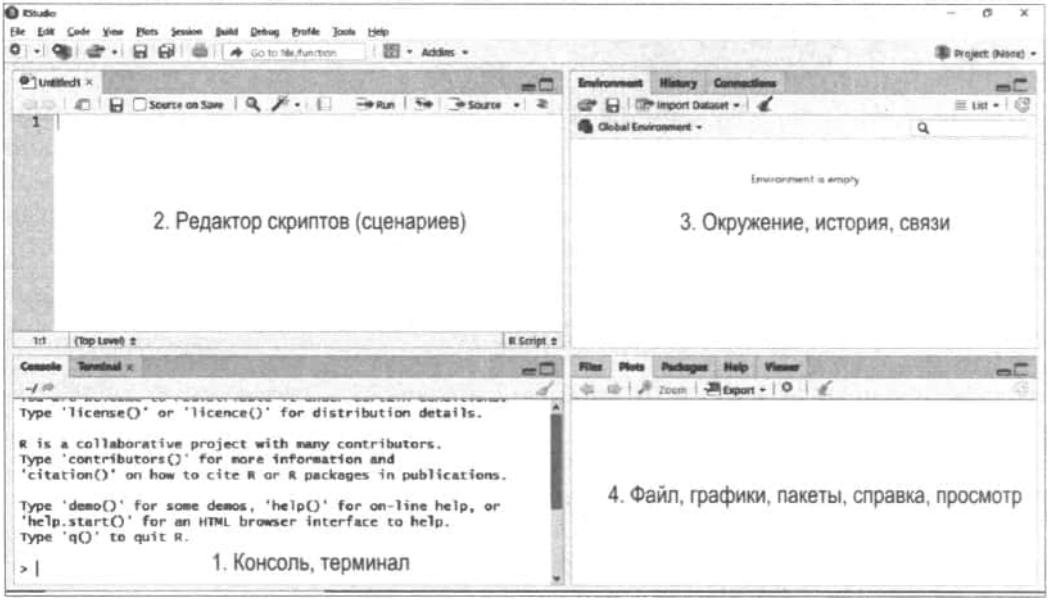


Рис. 6.1. Интегрированная среда разработки RStudio

ходимые для разработки кода, находились в одном месте. В каждой из четырех панелей рассмотрим функции, которые следует знать, чтобы начать работать.

Консоль (console), расположенная в левом нижнем углу RStudio, — это то место, откуда команды отправляются в R на выполнение. Здесь вы видите символ «>», за которым следует мигающий курсор. Здесь можно вводить операции и затем нажимать <Enter> для их выполнения. Давайте начнем с элементарной операции, например найдем 1 + 1, как показано на рис. 6.2.

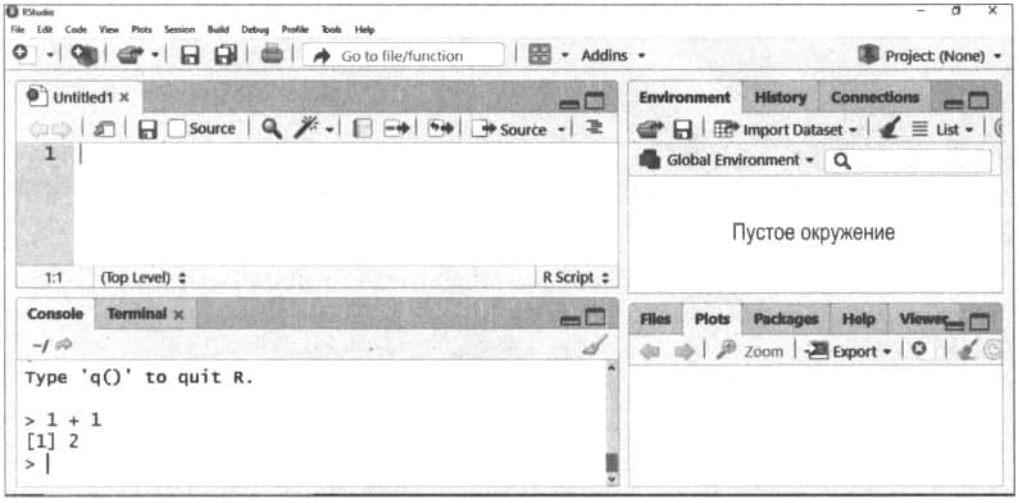


Рис. 6.2. Написание кода в RStudio

Вы можете заметить, что `[1]` появляется перед результатом `2`. Чтобы понять происходящее, введите и выполните в консоли `1:50`. Оператор `:` в R генерирует все числа с шагом 1 в заданном диапазоне, подобно маркеру заполнения в Excel. Вы увидите нечто подобное:

```
1:50
#> [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
#> [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
#> [47] 47 48 49 50
```

В квадратных скобках указана числовая позиция первого значения в каждой строке выходных данных.

Хотя вы можете продолжить работу непосредственно в консоли, рекомендуется сначала записать ваши команды в *скрипт* (сценарий), а затем отправить их в консоль. Таким образом вы сохраните запись кода, который запускали. Редактор скриптов (сценариев) находится в панели непосредственно над консолью. Введите в нем пару строк обычной арифметики, как показано на рис. 6.3.

Поместите курсор в строку 1, наведите указатель мыши на значки в верхней части редактора скриптов и найдите значок с подсказкой: **Run the current line or selection** (Запустите текущую строку или выбор). Щелкните этот значок, и произойдут две вещи. Во-первых, активная строка кода будет выполнена в консоли.

Курсор также переместится на следующую строку в редакторе скриптов. Можно отправить в консоль несколько строк сразу, выделив их и щелкнув этот значок. Сочетания клавиш для этой операции: <Ctrl>+<Enter> в ОС Windows и <Cmd>+ +<Return> для Mac. Как пользователь Excel, вы, скорее всего, любите использовать сочетания клавиш. В RStudio их множество, вы можете их посмотреть, выбрав **Tools** (Инструменты) | **Keyboard Shortcuts Help** (Справка по сочетанию клавиш).

Рис. 6.3. Работа с редактором скриптов в RStudio

Теперь сохраним наш скрипт. Выберите **File** (Файл) | **Save** (Сохранить). Назовите файл **ch-6**. Файлы скриптов R имеют расширение **.r**. Процесс открытия, сохранения и закрытия скриптов R напоминает работу с документом в текстовом редакторе; в конце концов, и то и другое является текстовой записью.

Теперь перейдем к нижней правой панели. Здесь вы увидите пять вкладок: **Files** (Файлы), **Plots** (Графики), **Packages** (Пакеты), **Help** (Справка), **Viewer** (Просмотр). Для языка R имеется огромное количество справочной документации, просмотреть которую можно в этой панели. Например, можно ознакомиться с подробной информацией о функции языка R с помощью оператора `?`.

Как пользователь Excel, вы хорошо знакомы с такими функциями, как `VLOOKUP()` или `SUMIF()`. Некоторые функции языка R очень похожи на функции Excel; например, функция R «квадратный корень», `sqrt()`. Введите в новой строке скрипта следующий код и запустите его с помощью значка меню или сочетания клавиш:

```
?sqrt
```

Документ с названием **Miscellaneous Mathematical Functions** (Различные математические функции) отобразится в окне **Help** (Справка). В нем содержится важная информация о функции `sqrt()`, аргументах, которые она может принимать, и т. д. Также там находится следующий пример использования этой функции:

```
require(stats) # для сплайна
require(graphics)
xx <- -9:9
```

```
plot(xx, sqrt(abs(xx)), col = "red")
lines(spline(xx, sqrt(abs(xx)), n=101), col = "pink")
```

Не пытайтесь понять смысл этого кода прямо сейчас; просто выделите, скопируйте и вставьте весь фрагмент в свой скрипт и запустите его. Вы увидите график, как на рис. 6.4. Я изменил размер панелей RStudio, чтобы увеличить график. Как строить графики в R, вы узнаете в *главе 8*.

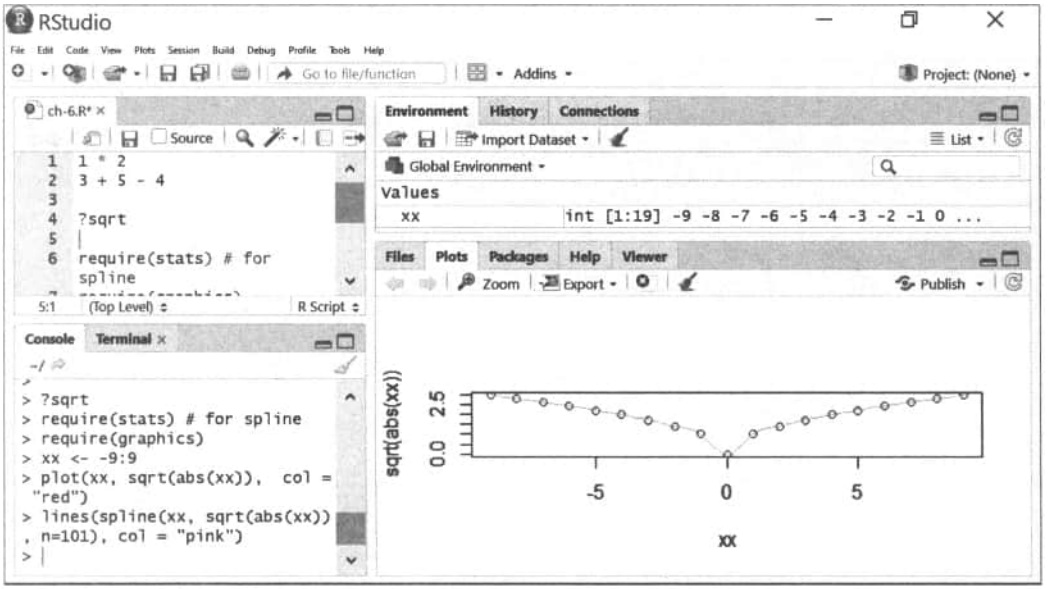


Рис. 6.4. Ваш первый график в R

Теперь посмотрите на правую верхнюю панель, где вы увидите следующее: **Environment** (Окружение), **History** (История), **Connections** (Связи). На вкладке **Environment** есть нечто с названием **xx** рядом с набором целых чисел. Что это такое? Оказывается, *вы* создали это с помощью кода, который я попросил вас запустить из документации `sqrt()`. На самом деле большая часть того, что мы делаем в R, связана с тем, что показано здесь, — *объектом*.

Вы могли заметить, что в этом туре по RStudio мы не рассмотрели несколько панелей, значков и пунктов меню. Это исключительно многофункциональная интегрированная среда разработки (IDE): смело исследуйте ее, экспериментируйте и используйте поисковую систему, чтобы узнать как можно больше. Но на данный момент вы уже достаточно много знаете о том, как работать в RStudio, чтобы приступить к изучению программирования на языке R. Вы уже видели, что R можно использовать как калькулятор. В табл. 6.1 перечислены некоторые распространенные арифметические операторы языка R.

Возможно, два последних оператора из табл. 6.1 вам мало знакомы: оператор `%%`, «остаток» (modulo), возвращает остаток от деления, а «целочисленное деление» (floor division) округляет результат деления до ближайшего целого числа.

Таблица 6.1. *Распространенные арифметические операторы R*

Оператор	Описание
+	Сложение (Addition)
-	Вычитание (Subtraction)
*	Умножение (Multiplication)
/	Деление (Division)
^	Степень (Exponent)
%%	Остаток (Modulo)
%/%	Целочисленное деление (Floor division)

Так же как и Excel, R следует порядку операций в арифметике.

```
# Умножение перед сложением
3 * 5 + 6
#> [1] 21

# Деление перед вычитанием
2 / 2 - 7
#> [1] -6
```

Что делать со строками, содержащими решетку (#) и текст? Это комментарии, используемые для инструкций и справочной информации о коде. Комментарии помогают другим пользователям, а позже будут помогать и нам, понять и запомнить, для чего используется код. Язык R не исполняет комментарии: эта часть скрипта предназначена для программиста, а не для компьютера. Хотя комментарии могут располагаться справа от кода, предпочтительно размещать их над кодом:

```
1 * 2 # Этот комментарий возможен
#> [1] 2

# Этот комментарий предпочтителен
2 * 1
#> [1] 2
```

Нет необходимости использовать комментарии, чтобы объяснять *всё,* что делает код, но имеет смысл описывать ваши допущения и аргументы. Рассматривайте их как комментарии. Я буду использовать комментарии в примерах везде, где это актуально и полезно.

Возьмите за правило включать комментарии для документирования ваших целей, допущений и аргументов при написании кода.

Как уже говорилось, функции в R по большей части работают так же, как Excel, и часто выглядят очень похоже. Например, так мы можем получить абсолютное значение –100:

```
# Чему равно абсолютное значение - 100?
abs(-100)
#> [1] 100
```

Однако имеются некоторые важные отличия при работе с функциями в R, как видно из следующих примеров.

```
# Это не будет работать
ABS(-100)
#> Error in ABS(-100) : could not find function "ABS"
Abs(-100)
#> Error in Abs(-100) : could not find function "Abs"
```

[1]

В Excel можно ввести функцию `ABS()` в нижнем регистре, `abs()`, или с заглавной буквы, `Abs()`. Однако в языке R функция `abs()` *должна быть* введена в нижнем регистре по причине того, что язык R *чувствителен* к регистру. Это главное различие между Excel и R, и вы, скорее всего, рано или поздно споткнетесь об него.

Язык R чувствителен к регистру, например функция `SQRT()` не то же самое, что `sqrt()`.

Некоторые функции в R, такие как `sqrt()`, предназначены для работы с числами аналогично Excel; другие, такие как `toupper()`, работают с символами:

```
# Преобразовать в верхний регистр
toupper('I love R')
#> [1] "I LOVE R"
```

Давайте рассмотрим еще один случай, когда R ведет себя так же, как Excel, за одним исключением, которое будет иметь огромные последствия: *операторы сравнения*. Речь идет о сравнении некоторых отношений между двумя значениями, например больше ли одно число, чем другое.

```
# 3 больше 4?
3 > 4
#> [1] FALSE
```

Язык R, так же как Excel, возвращает `TRUE` (истинно) или `FALSE` (ложно) в качестве результата любого оператора сравнения. В табл. 6.2 перечислены операторы сравнения языка R.

Большинство из этих операторов, вероятно, выглядят для вас знакомыми, за одним исключением... Вы обратили внимание на последний оператор? Совершенно верно, в R мы проверяем равенство двух величин друг другу с помощью не одного знака равенства, а с помощью *двух*, поскольку один знак равенства используется в R для *присваивания объектов*.

---

[1] Ошибка в ABS(-100) / в Abs(-100): невозможно найти функцию "ABS" / "Abs". — *Ред.*

Таблица 6.2. *Операторы сравнения в R*

Оператор	Значение
>	Больше (Greater than)
<	Меньше (Less than)
>=	Больше или равно (Greater than or equal to)
<=	Меньше или равно (Less than or equal to)
!=	Не равно (Not equal to)
==	Равно (Equal to)

Объекты и переменные

Хранимые объекты иногда еще называют *переменными*, т. к. они могут перезаписываться и изменять значения. Однако в этой книге мы уже употребляли термин «переменные» в статистическом смысле. Чтобы избежать путаницы в терминологии, продолжим употреблять термин «объекты» относительно программирования и «переменные» — относительно статистики.

Если вы еще не вполне уверены в том, что поняли разницу, давайте рассмотрим еще один пример. Присвоим объекту абсолютное значение –100 и назовем его **my_first_object**.

```
# Присваивание объекта в R
my_first_object = abs(-100)
```

Можно воспринимать объект как обувную коробку, в которую мы помещаем часть информации. С помощью оператора = мы сохранили результат функции `abs(-100)` в «обувной коробке» под названием **my_first_object**. Можно «открыть обувную коробку», распечатав ее. В R это можно сделать, просто запустив имя объекта:

```
# Печать объекта в R
my_first_object
#> [1] 100
```

Другой способ присваивания объектов в R — использование оператора <-. Он даже более предпочтителен, чем оператор =, отчасти потому, что позволяет не путать его с оператором ==. Попробуйте присвоить другой объект с помощью этого оператора, а затем распечатайте его. Комбинации клавиш для такой задачи следующие: <Alt>+<–> в ОС Windows и <Option+<–> в Mac. Можно подойти к функциям и операциям творчески, как это сделал я:

```
my_second_object <- sqrt(abs(-5 ^ 2))
my_second_object
#> [1] 5
```

Имена объектов в R должны начинаться с буквы или точки и содержать только буквы, цифры, подчеркивания и точки. Также имеется несколько запрещенных ключевых слов. Все это оставляет много места для «творческого» именования объектов. Правильные имена объектов указывают на данные, которые в них хранятся,

так же как этикетка на коробке с обувью указывает, какой тип обуви находится внутри.

R и руководства по стилю программирования

Некоторые люди и организации объединили соглашения по программированию в так называемые руководства по стилю, так же как газета может иметь руководство по стилю для написания статей. Эти руководства по стилю рассказывают, какие операторы присваивания использовать, как именовать объекты и многое другое. Одно из таких руководств по стилю программирования в R разработано компанией Google и доступно в Интернете.

Объекты могут содержать разные типы, или *режимы, данных*, так же как у вас могут быть разные категории обувных коробок. В табл. 6.3 перечислены некоторые распространенные типы данных.

Таблица 6.3. *Распространенные типы данных в R*

Тип данных	Пример
Символьные (Character)	`'R', 'Mount', 'Hello, world'`
Числовые (Numeric)	`6.2, 4.13, 3`
Целочисленные (Integer)	`3L, -1L, 12L`
Логические (Logical)	`TRUE, FALSE, T, F`

Давайте создадим несколько объектов разных типов (режимов). Во-первых, символьные данные часто заключаются в одиночные кавычки для удобочитаемости, хотя двойные кавычки также подходят и могут быть особенно полезны, если вы предполагаете использовать одиночные кавычки как часть входных данных.

```
my_char <- 'Hello, world'
my_other_char <- "We're able to code R![2]"
```

Числа могут быть как десятичные, так и целые:

```
my_num <- 3
my_other_num <- 3.21
```

Но целые числа также можно хранить как отдельный целочисленный тип данных. Символ `L` во входных данных означает *литерал*; этот термин пришел из информатики и используется для обозначения фиксированных значений:

```
my_int <- 12L
```

Символы `T` и `F` по умолчанию рассматриваются как логические данные `TRUE` (истинно) и `FALSE` (ложно), соответственно:

```
my_logical <- FALSE
my_other_logical <- F
```

[2] Мы умеем программировать на R. — *Ред.*

Можно использовать функцию `str()` для характеристики структуры объекта, например его типа и информации, содержащейся внутри:

```
str(my_char)
#> chr "Hello, world"
str(my_num)
#> num 3
str(my_int)
#> int 12
str(my_logical)
#> logi FALSE
```

Присвоив объекты, мы можем использовать их в различных операциях:

```
# my_num равно 5.5?
my_num == 5.5
#> [1] FALSE

# Количество символов в my_char
nchar(my_char)
#> [1] 12
```

Объекты можно даже использовать в качестве входных данных при присваивании других объектов или переназначить их:

```
my_other_num <- 2.2
my_num <- my_num/my_other_num
my_num
#> [1] 1.363636
```

«Что же из этого? — спросите вы. — Я работаю с множеством данных; каким образом присвоение каждого числа его собственному объекту может мне помочь?» К счастью, как вы увидите в *главе 7*, можно объединять несколько значений в один объект, аналогично тому, как вы можете делать это с диапазонами и рабочими листами в Excel. Но сначала ненадолго сменим тему и познакомимся с *пакетами*.

Пакеты в R

Представьте, что вы не можете загружать приложения на свой смартфон. Вы попрежнему достаточно удобно можете лишь совершать телефонные звонки, просматривать страницы в Интернете и делать заметки и т. д. Но настоящая сила смартфона заключается в его приложениях. Язык R поставляется подобно смартфону в заводской конфигурации: он и в таком виде достаточно полезен, с его помощью вы могли бы выполнять почти все необходимые действия. Но часто более эффективно выполнить в отношении R *установку пакета*, что эквивалентно установке приложения.

Поставляемая в заводской конфигурации версия R называется «базовым R». Пакеты, выполняющие в R роль приложений, являются разделяемыми единицами кода,

включающими в себя функции, наборы данных, документацию и многое другое. Эти пакеты создаются на основе базового R для улучшения имеющихся функций и и добавления новых.

Ранее вы загрузили базовый R из сети CRAN. В этой сети также размещено более 10 000 пакетов, предоставленных обширной пользовательской базой R и проверенных добровольцами CRAN. Это ваш «магазин приложений» для R. Хотя можно загрузить пакеты из других источников, начинающим лучше использовать те, что размещены в сети CRAN. Чтобы установить пакет из сети CRAN, вы можете выполнить команду `install.packages()`.

Представление задач CRAN

Новичкам может быть сложно определить, какие пакеты R им нужны. К счастью, команда `CRAN` с помощью *представления задач CRAN* предоставляет нечто вроде «рекомендованного плейлиста» пакетов. Это наборы пакетов, предназначенные для того, чтобы помочь во всем, от эконометрии до генетики, и обеспечить пользователей полезными пакетами R. Продвигаясь в изучении языка, вы научитесь находить и подбирать пакеты под свои требования.

В этой книге мы будем использовать пакеты для решения таких задач, как обработка и визуализация данных. В частности, мы будем использовать библиотеку **tidyverse**, представляющую собой коллекцию пакетов, предназначенных для совместного использования. Чтобы установить эту коллекцию, выполните в консоли следующую команду:

```
install.packages('tidyverse')
```

Итак, вы установили ряд полезных пакетов; один из них, `dplyr` (обычно произносится как «ди-плаер», «d-plier»), включает функцию `arrange()`. Если вы попытаетесь открыть документацию для этой функции, то получите ошибку:

```
?arrange
#> No documentation for 'arrange' in specified packages and libraries:
#> you could try '??arrange'[3]
```

Чтобы понять, почему R не может найти функцию `tidyverse`, вернемся к аналогии со смартфоном: хотя вы и установили приложение, прежде чем использовать, его надо открыть. То же самое с R: мы установили пакет с помощью команды `install.packages()`, но теперь нам надо вызвать его в нашей сессии с помощью функции `library()`:

```
# Вызвать tidyverse в нашей сессии
library(tidyverse)
#> -- Attaching packages ------------------------- tidyverse 1.3.0 --
#> v ggplot2 3.3.2     v purrr 0.3.4
#> v tibble 3.0.3      v dplyr 1.0.2
#> v tidyr  1.1.2      v stringr 1.4.0
#> v readr  1.3.1      v forcats 0.5.0
```

[3] Нет документации для 'arrange' в указанных пакетах и библиотеках: вы можете попробовать '??arrange'. — *Ред.*

```
#> -- Conflicts ------------------------------ tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
```

Пакеты библиотеки `tidyverse` теперь доступны для всех сессий R; теперь можно выполнить упомянутое действие без ошибки.

Пакеты *устанавливаются* один раз, но *вызываются* для каждой сессии.

Обновление R, RStudio и пакетов R

RStudio, пакеты R и сам язык R постоянно совершенствуются, поэтому их следует периодически проверять на наличие обновлений. Чтобы обновить RStudio, найдите и выберите в меню **Help** (Справка) | **Check for Updates** (Проверить наличие обновлений). Если необходимо обновление, RStudio проведет вас через шаги обновления.

Чтобы обновить все пакеты из сети CRAN, вы можете выполнить следующую функцию и ряд предложенных шагов:

```
update.packages()
```

Также можно обновить пакеты из меню RStudio, выбрав последовательно **Tools** (Инструменты) | **Check for Package Updates** (Проверить наличие обновлений для пакетов). Появится меню **Update Packages** (Обновление пакетов); выберите все пакеты, которые следует обновить. Также можно установить пакеты в меню **Tools** (Инструменты).

Обновление самого R, к сожалению, более сложно. На компьютере с ОС Windows можно использовать функцию `updateR()` из пакета `installr` и выполнить следующие инструкции:

```
# Обновление R для Windows
install.packages('installr')
library(installr)
updateR()
```

На ПК с Mac установку последней версии R можно выполнить с сайта CRAN.

Заключение

В этой главе вы узнали, как работать с объектами и пакетами в R и приобрели навыки работы в RStudio. Вы многое узнали; полагаю, пришло время прерваться. Сохраните ваши скрипты R и закройте RStudio, выбрав **File** (Файл) | **Quit Session** (Завершить сессию). После этого вы увидите вопрос: **Save workspace image to ~/.RData?** (Сохранить образ рабочей области в ~/.RData?) Чаще всего сохранять образ вашей рабочей области не нужно. Если вы сделаете это, копия всех ваших

объектов будет сохранена, так что они будут доступны для вас в следующих сессиях. Хотя это *звучит правильно*, хранение этих объектов и отслеживание причин, по которым вы их сохранили, может оказаться обременительным.

Вместо этого положитесь на скрипты R для повторного создания объектов в следующей сессии. В конце концов, преимущество языка программирования в том, что он является воспроизводимым: не нужно таскать за собой объекты, если при необходимости можно их создать.

Предпочтительно не сохранять образ вашей рабочей области: вы должны уметь повторно создавать любые объекты из предыдущей сессии, используя свой скрипт.

Чтобы предотвратить сохранение RStudio вашего рабочего пространства между сессиями, перейдите в главное меню и выберите **Tools** (Инструменты) | **Global Options** (Глобальные параметры). В меню **General** (Общее) измените две настройки в области **Workspace** (Рабочее пространство), как показано на рис. 6.5.

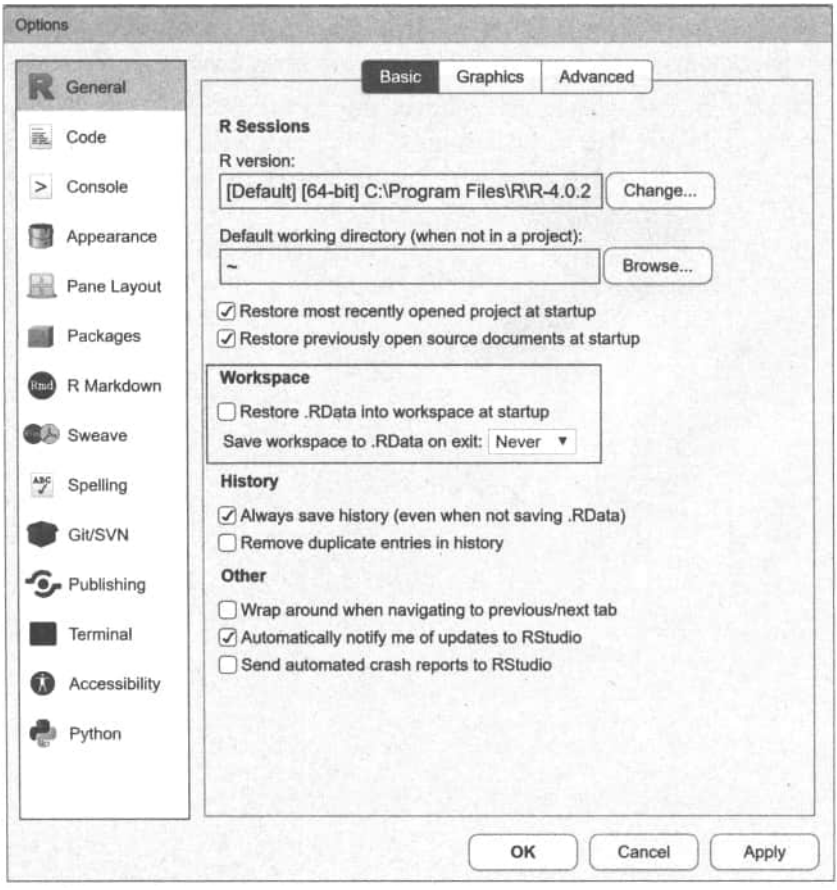


Рис. 6.5. Настраиваемое рабочее пространство в RStudio

Упражнения

Следующие упражнения предназначены для дополнительной практики и понимания работы с объектами, пакетами и RStudio.

1. Помимо богатого набора инструментов, в RStudio существует огромное количество настроек внешнего вида. Выберите в меню **Tools** (Инструменты) | **Global Options** (Глобальные параметры) | **Appearance** (Внешний вид) и настройте шрифт и тему редактора. Например, вы можете выбрать тему «Темный режим» и пр.
2. Используя скрипт в RStudio, выполните следующие действия:
   - назначьте сумму 1 и 4 как `a`;
   - назначьте квадратный корень из `a` как `b`;
   - назначьте `b` минус 1 как `d`;
   - ответьте на вопрос: какой тип данных сохранен в `d`?
   - ответьте на вопрос: `d` больше 2?
3. Установите пакет `psych` из CRAN и загрузите его в свою сессию. С помощью комментариев объясните разницу между установкой и загрузкой пакета.

Наряду с этими упражнениями рекомендую немедленно начать использовать язык R в повседневной работе. На первых порах он будет служить вам всего лишь в качестве калькулятора, но и это позволит вам освоиться с использованием языка R и RStudio.

ГЛАВА 7

Структуры данных в R

В конце *главы 6* вы узнали, как работать с пакетами в R. Как правило, все необходимые пакеты загружаются в начале скрипта, чтобы это не потребовалось делать в дальнейшем. Следуя этой традиции, мы сейчас вызовем все скрипты, которые понадобятся в этой главе. Возможно, некоторые из них вам придется установить; чтобы вспомнить, как это делается, обратитесь к *главе 6*. Я буду давать пояснения по этим пакетам по мере того, как мы будем с ними работать.

```
# Для импорта и экспорта данных
library(tidyverse)

# Для чтения из файлов Excel
library(readxl)

# Для описательной статистики
library(psych)

# Для записи данных в Excel
library(writexl)
```

Векторы

В *главе 6* вы также узнали о вызове функций для данных различных режимов и присвоении данных объектам:

```
my_number <- 8.2
sqrt(my_number)
#> [1] 2.863564

my_char <- 'Hello, world'
toupper(my_char)
#> [1] "HELLO, WORLD"
```

Скорее всего, обычно вы работаете более чем с одним элементом данных в каждый момент времени, поэтому присвоение каждого элемента собственному объекту нецелесообразно. В Excel можно поместить данные в смежные ячейки, называемые *диапазоном*, и легко работать с этими данными. На рис. 7.1 показаны простые примеры работы с диапазонами чисел и текста в Excel.

	A	B	C	D	E	F	G	H
1	Билли	БИЛЛИ	=UPPER(A1)		5	8	2	7
2	Джек	ДЖЕК	=UPPER(A2)		2.236067977	2.828427125	1.414213562	2.645751311
3	Джилл	ДЖИЛЛ	=UPPER(A3)		=SQRT(E1)	=SQRT(F1)	=SQRT(G1)	=SQRT(H1)
4	Джонни	ДЖОННИ	=UPPER(A4)					
5	Сьюзи	СЬЮЗИ	=UPPER(A5)					
6								

Рис. 7.1. Работа с диапазонами в Excel

Ранее я связывал *режим объекта* с определенным типом обуви в обувной коробке. Можно сказать, что *структура объекта* — это форма, размер и конфигурация «обувной коробки». Вы уже находили структуру объекта R с помощью функции `str()`.

Язык R содержит несколько структур объектов: можно хранить порцию данных и работать с ней, помещая ее в специальную структуру, называемую *вектором*. Векторы — это наборы из одного или нескольких элементов данных того же типа. Оказывается, мы уже пользовались векторами, что можно подтвердить с помощью функции `is.vector()`:

```
is.vector(my_number)
#> [1] TRUE
```

Хотя `my_number` является вектором, он содержит всего один элемент — нечто вроде одной ячейки в Excel. В R мы бы сказали, что этот вектор имеет длину, равную 1:

```
length(my_number)
#> [1] 1
```

С помощью функции `c()` можно создать вектор из нескольких элементов, наподобие диапазона в Excel. Эта функция имеет такое название, поскольку служит для *объединения* нескольких элементов в один вектор. Давайте испробуем ее:

```
my_numbers <- c(5, 8, 2, 7)
```

Этот объект действительно является вектором — он содержит числовые данные и имеет длину, равную 4:

```
is.vector(my_numbers)
#> [1] TRUE

str(my_numbers)
#> [1] num [1:4] 5 8 2 7

length(my_numbers)
#> [1] 4
```

Теперь посмотрим, что произойдет, когда мы вызовем функцию для `my_numbers`:

```
sqrt(my_numbers)
#> [1] 2.236068 2.828427 1.414214 2.645751
```

Мы достигли определенного прогресса. Аналогично можно работать с символьным вектором:

```
roster_names <- c('Jack', 'Jill', 'Billy', 'Susie', 'Johnny')
toupper(roster_names)
#> [1] "JACK"   "JILL"   "BILLY"   "SUSIE"   "JOHNNY"
```

Объединив элементы данных в векторы посредством функции `c()`, мы смогли легко воспроизвести в языке R то, что показано на рис. 7.1 в Excel. А если элементы разных типов присвоены одному и тому же вектору? Давайте попробуем и узнаем, что произойдет:

```
my_vec <- c('A', 2, 'C')
my_vec
#> [1] "A" "2" "C"

str(my_vec)
#> chr [1:3] "A" "2" "C"
```

R *приведет* все элементы к одному типу, чтобы их можно было объединить в вектор; например, числовой элемент 2 из предыдущего примера преобразован в символ.

Индексирование и подмножества векторов

В Excel функция `INDEX()` предназначена для того, чтобы найти позицию элемента в диапазоне. Например, я буду использовать функцию `INDEX()` для извлечения элемента в третьей позиции диапазона `roster_names` (ячейки `A1:A5`) (рис. 7.2).

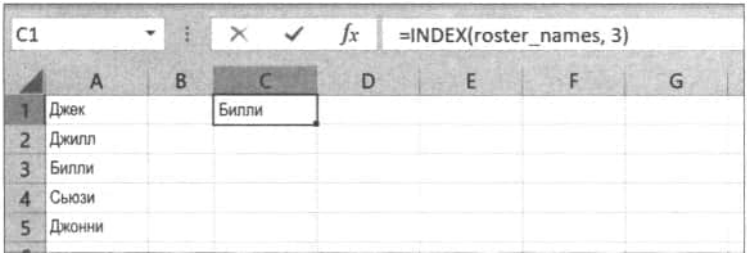


Рис. 7.2. Функция **INDEX()** для диапазона в Excel

Аналогичным образом можно проиндексировать вектор в R, привязав желаемую позицию индекса внутри скобок к имени объекта:

```
# Получить третий элемент вектора roster_names
roster_names[3]
#> [1] "Billy"
```

Используя ту же нотацию, можно выбрать несколько элементов по порядковому номеру, которые мы будем называть *подмножеством*. Мы снова используем оператор :, чтобы выбрать все элементы между позициями 1 и 3:

```
# Получить с первого по третий элементы
roster_names[1:3]
#> [1] "Jack"   "Jill"   "Billy"
```

Также при этом можно использовать функции. Помните функцию `length()`? С ее помощью можно получить в векторе все элементы до последнего:

```
# Получить со второго по последний элементы
roster_names[2:length(roster_names)]
#> [1] "Jill"   "Billy"   "Susie"   "Johnny"
```

Мы можем даже использовать функцию `c()`, чтобы проиндексировать непоследовательные элементы вектора:

```
# Получить второй и пятый элементы
roster_names[c(2, 5)]
#> [1] "Jill"   "Johnny"
```

От таблиц Excel к кадрам данных R

«Все это прекрасно, — можете подумать вы, — но я просто не работаю с такими малыми диапазонами. А как насчет целых таблиц данных?» В конце концов, в *главе 1* вы узнали все о важности организации данных в переменные и наблюдения, как в наборе данных **star** (рис. 7.3). Это пример двумерной структуры данных.

	A	B	C	D	E	F	G	H	I
1	id	tmathssk	treadssk	classk	totexpk	sex	freelunk	race	schidkn
2	1	473	447	small.class	7	girl	no	white	63
3	2	536	450	small.class	21	girl	no	black	20
4	3	463	439	regular.with.aide	0	boy	yes	black	19
5	4	559	448	regular	16	boy	no	white	69
6	5	489	447	small.class	5	boy	yes	white	79
7	6	454	431	regular	8	boy	yes	white	5
8	7	423	395	regular.with.aide	17	girl	yes	black	16
9	8	500	451	regular	3	girl	no	white	56
10	9	439	478	small.class	11	girl	no	black	11
11	10	528	455	small.class	10	girl	no	white	66

Рис. 7.3. Двумерная структура данных в Excel

В то время как вектор в R является одномерным, кадр, или рамка (также фрейм — от *frame*), данных позволяет хранить данные как в строках, так и в столбцах. Это делает кадр (рамку) данных R эквивалентом таблицы в Excel. Формально кадр данных — двумерная структура данных, в которой записи в каждом столбце имеют одинаковый режим, а столбцы — одинаковую длину. В R, так же как и в Excel, принято присваивать имена, или метки, каждому столбцу и строке.

Можно создать кадр данных с нуля с помощью функции `data.frame()`. Давайте создадим и затем распечатаем кадр данных **roster**:

```
roster <- data.frame(
name = c('Jack', 'Jill', 'Billy', 'Susie', 'Johnny'),
```

```
height = c(72, 65, 68, 69, 66),
injured = c(FALSE, TRUE, FALSE, FALSE, TRUE))

roster
#>   name   height   injured
#> 1 Jack     72       FALSE
#> 2 Jill     65        TRUE
#> 3 Billy    68       FALSE
#> 4 Susie    69       FALSE
#> 5 Johnny   66        TRUE
```

Ранее мы использовали функцию `c()`, чтобы объединить элементы в вектор. Действительно, кадр данных можно рассматривать как *набор векторов* одинаковой длины. Имея три переменные и пять наблюдений, `roster` представляет собой довольно миниатюрный кадр данных. К счастью, кадр данных не всегда требуется создавать с нуля, как в этом случае. R поставляется с множеством наборов данных, чтобы просмотреть их список, воспользуйтесь этой функцией:

```
data()
```

Меню **R data sets** (Наборы данных R) появится как новое окно на панели скриптов. Многие, но не все, из этих наборов структурированы как кадры данных. Например, вы могли ранее встречаться с известным набором данных **iris**, встроенным в R.

Как и любой объект, набор данных **iris** можно распечатать; однако 150 строк данных быстро перегрузят вашу консоль (представьте, как усугубится эта проблема с тысячами и миллионами строк). Вместо этого, как правило, распечатывают несколько первых строк, используя функцию `head()`:

```
head(iris)
#> Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
#> 1          5.1         3.5          1.4         0.2    setosa
#> 2          4.9         3.0          1.4         0.2    setosa
#> 3          4.7         3.2          1.3         0.2    setosa
#> 4          4.6         3.1          1.5         0.2    setosa
#> 5          5.0         3.6          1.4         0.2    setosa
#> 6          5.4         3.9          1.7         0.4    setosa
```

Убедиться в том, что `iris` действительно является кадром данных, можно с помощью функции `is.data.frame()`:

```
is.data.frame(iris)
#> [1] TRUE
```

Еще один способ познакомиться с новым набором данных, помимо печати, — использовать функцию `str()`:

```
str(iris)
#> 'data.frame':        150 obs. of 5 variables:
#> $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
#> $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
```

```
#> $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
#> $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
#> $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 ...
```

Выходные данные возвращают размер кадра данных и некоторую информацию о его столбцах. Как видите, четыре из них — числовые. Последний, **Species**, является *фактором*. Факторы — это особый способ хранения переменных, принимающих ограниченное количество значений. Они особенно полезны для хранения *категориальных переменных*: вы увидите далее, что **Species** имеет три *уровня* (термин, который мы использовали в статистическом смысле при описании категориальных переменных).

Хотя факторы выходят за рамки настоящей книги, отмечу, что они обладают множеством преимуществ при работе с категориальными переменными, например позволяют более эффективно хранить данные. Чтобы больше узнать о факторах, найдите в справочной документации по R описание функции `factor()` (можно сделать это с помощью оператора `?`). В библиотеку `tidyverse` также включен пакет `forcats` как основной пакет для работы с факторами.

В дополнение к наборам данных, предварительно загруженных в R, многие пакеты содержат собственные данные. Узнать о них можно при помощи функции `data()`. Давайте посмотрим, включает ли пакет `psych` какие-либо наборы данных:

```
data(package = 'psych')
```

Меню **R data sets** (Наборы данных R) снова запустится в новом окне; на этот раз появится дополнительный раздел **Data sets in package** `psych` (Наборы данных в пакете `psych`). Один из наборов данных называется `sat.act`. Чтобы этот набор данных стал доступен в нашей сессии R, можно снова использовать функцию `data()`. Теперь это присвоенный объект R, который вы можете найти в меню **Environment** (Окружение) и использовать как любой другой объект. Давайте убедимся, что это кадр данных:

```
data('sat.act')
str(sat.act)
#> 'data.frame':          700 obs. of 6 variables:
#> $ gender    :  int 2 2 2 1 1 1 2 1 2 2 ...
#> $ education :  int 3 3 3 4 2 5 5 3 4 5 ...
#> $ age       : int 19 23 20 27 33 26 30 19 23 40 ...
#> $ ACT       : int 24 35 21 26 31 28 36 22 22 35 ...
#> $ SATV      : int 500 600 480 550 600 640 610 520 400 730 ...
#> $ SATQ      : int 500 500 470 520 550 640 500 560 600 800 ...
```

Импорт данных в R

При работе в Excel принято хранить, анализировать и представлять данные в одной рабочей книге. Хранение данных внутри R-скрипта, напротив, является редкостью. Как правило, данные импортируются из внешних источников, начиная от текстовых файлов и баз данных до веб-страниц и программных интерфейсов (API), изо-

бражений и аудио, и только затем анализируются в R. Результаты анализа затем часто экспортируются в различные источники. Мы начнем этот процесс с чтения данных, что неудивительно, из рабочих книг Excel (расширение файла **.xlsx**) и файлов с разделителями-запятыми (расширение **.csv**).

Базовый R и библиотека tidyverse

В *главе 6* вы узнали о взаимосвязи между базовым R и пакетами R. Хотя пакеты способны помочь выполнить действия, довольно сложные для базового R, иногда они предлагают альтернативные варианты. Например, базовый R включает в себя функции для чтения **.csv**-файлов (но не файлов Excel), а также функции для построения графиков. Мы будем использовать функции библиотеки **tidyverse** для этих и других информационных нужд. С учетом имеющихся целей и задач нет ничего плохого в использовании аналогов базового R. Я решил здесь сделать упор на инструменты библиотеки **tidyverse**, поскольку их синтаксис более понятен пользователям Excel.

Чтобы импортировать данные в R, важно понимать, как работают пути к файлам и каталоги. Каждый раз, когда вы используете программу, вы работаете из «главного окна» на своем компьютере или *рабочего каталога*. Предполагается, что любые файлы, на которые вы ссылаетесь из R, например при импорте набора данных, расположены относительно этого рабочего каталога. Функция `getwd()` выводит на печать путь к файлу рабочего каталога. Работая с ОС Windows, вы увидите результат, подобный следующему:

```
getwd()
#> [1] "C:/Users/User/Documents"
```

На Mac он выглядит примерно так:

```
getwd()
#> [1] "/Users/user"
```

Язык R имеет глобальный рабочий каталог по умолчанию, один и тот же при запуске каждой сессии. Я полагаю, что вы запускаете файлы из загруженной или клонированной копии репозитория, прилагаемого к книге (на GitHub), и также работаете из R-скрипта в той же папке. В таком случае лучше всего установить рабочий каталог в эту папку, используя функцию `setwd()`. Если вы не привыкли использовать пути к файлам, возможно, будет сложно правильно заполнить поле; к счастью, для этого в RStudio имеются возможности, ориентированные на меню.

Чтобы изменить рабочий каталог на ту же папку, в которой находится текущий R-скрипт, выберите последовательно: **Session** (Сессия) | **Set Working Directory** (Установить рабочий каталог) | **To Source File Location** (К расположению исходных файлов). Вы должны увидеть результаты работы функции `setwd()`, появившиеся в консоли. Еще раз запустив функцию `getwd()`, вы увидите, что находитесь в другом рабочем каталоге.

Теперь, когда рабочий каталог установлен, потренируемся в работе с файлами относительно этого каталога. Я поместил файл **test-file.csv** в главную папку репозитория на GitHub. Чтобы проверить, можно ли его найти, используем функцию `file.exists()`:

```
file.exists('test-file.csv')
#> [1] TRUE
```

Кроме того, я поместил копию этого файла в подпапку **test-folder** репозитория. На этот раз необходимо указать, в какой подпапке нужно искать:

```
file.exists('test-folder/test-file.csv')
#> [1] TRUE
```

Что произойдет, если понадобится перейти на папку уровнем *выше*? Попробуйте поместить копию файла **test-file** в любую папку на один уровень выше текущего каталога. Можно использовать оператор .., чтобы R понял, где нужно искать (на одну папку выше):

```
file.exists('../test-file.csv')
#> [1] TRUE
```

Проекты RStudio

В репозитории на GitHub вы найдете файл с именем **aia-book.Rproj**. Это файл проекта RStudio. Проект — отличный способ сохранить вашу работу; например, при выходе из RStudio в проекте сохранится конфигурация окон и файлов, которые вы открыли. Кроме того, проект автоматически установит ваш рабочий каталог в *каталог проекта*, так что вам не потребуется встроенная функция `setwd()` для каждого скрипта.

При работе с R в этом репозитории попробуйте использовать файл **.Rproj**. Вы можете открыть любой файл на панели **Files** (Файлы) в нижней правой области RStudio.

Теперь, когда вы умеете находить файлы в R, давайте считаем некоторые из них. В репозитории к этой книге имеется папка **datasets**, под которой расположена подпапка **star**. Она, кроме прочих, содержит два файла: **districts.csv** и **star.xlsx**.

Чтобы считать файлы **.csv**, можно воспользоваться функцией `read_csv()` из пакета `readr`. Этот пакет входит в состав `tidyverse`, следовательно, нет необходимости устанавливать или загружать что-то новое. Мы передадим местонахождение файла в функцию. (Теперь вы понимаете, что́ такое рабочие каталоги и пути к файлам и насколько они полезны?)

```
read_csv('datasets/star/districts.csv')
#>-- Column specification --------------------------
#> cols(
#>  schidkn = col_double(),
#>  school_name = col_character(),
#>  county = col_character()
#> )
#>
#> # Тиббл: 89 x 3
#>   schidkn school_name    county
#>     <dbl>  <chr>          <chr>
#> 1       1 Rosalia        New Liberty
#> 2       2 Montgomeryville Topton
#> 3       3 Davy           Wahpeton
```

```
#> 4      4 Steelton        Palestine
#> 5      5 Bonifay         Reddell
#> 6      6 Tolchester      Sattley
#> 7      7 Cahokia         Sattley
#> 8      8 Plattsmouth     Sugar Mountain
#> 9      9 Bainbridge      Manteca
#>10     10 Bull Run        Manteca
#> # ... и еще 79 строк
```

Мастер импорта наборов данных в RStudio (Import Dataset Wizard)

Если вам сложно импортировать набор данных, попробуйте инструмент импорта данных, доступный в RStudio с помощью меню. Выберите последовательно **File** (Файл) | **Import Dataset** (Импортировать набор данных). Вам будет представлен ряд последовательных шагов, которые проведут вас через весь процесс импорта, включая возможность переходить к исходному файлу с помощью проводника файлов вашего компьютера.

Результатом будет получение достаточно большого количества выходных данных. Во-первых, определены столбцы и показаны функции, использовавшиеся для анализа данных в R. Во-вторых, несколько первых строк данных перечислены в виде *тиббла* (tibble). Тиббл — это новая форма кадра (рамки, фрейма) данных; это по-прежнему кадр данных, и ведет он себя в основном как кадр данных, но имеет некоторые усовершенствования, облегчающие работу, особенно в `tidyverse`.

Хотя мы могли считывать данные в R, с ними мало что можно было делать, мы не могли работать с ними, не присвоив их объекту:

```
districts <- read_csv('datasets/star/districts.csv')
```

Среди множества преимуществ тиббла есть одна приятная особенность: можно распечатать данные, не беспокоясь о перегрузке вывода консоли; на печать выводятся только первые 10 строк:

```
districts
#> # Тиббл: 89 x 3
#>   schidkn school_name       county
#>     <dbl> <chr>             <chr>
#> 1       1 Rosalia           New Liberty
#> 2       2 Montgomeryville   Topton
#> 3       3 Davy              Wahpeton
#> 4       4 Steelton          Palestine
#> 5       5 Bonifay           Reddell
#> 6       6 Tolchester        Sattley
#> 7       7 Cahokia           Sattley
#> 8       8 Plattsmouth       Sugar Mountain
#> 9       9 Bainbridge        Manteca
#> 10      10 Bull Run         Manteca
#> # ... и еще 79 строк
```

Пакет `readr` не включает в себя возможность импортировать рабочие книги Excel; для этого мы будем использовать пакет `readxl`. Хотя этот пакет является частью

библиотеки `tidyverse`, он не загружается с основным набором пакетов, как пакет `readr`, и поэтому мы импортировали его отдельно в начале этой главы.

Для импорта файла **star.xlsx** в виде тиббла используем функцию `read_xlsx()`:

```
star <- read_xlsx('datasets/star/star.xlsx')
head(star)
#> # Тиббл: 6 x 8
#>   tmathssk treadssk  classk     totexpk  sex   freelunk  race   schidkn
#>     <dbl>     <dbl>  <chr>       <dbl> <chr> <chr>     <chr>    <dbl>
#> 1      473       447  small.class  7     girl   no       white       63
#> 2      536       450  small.class  21    girl   no       black       20
#> 3      463       439  regular.wit~ 0     boy    yes      black       19
#> 4      559       448  regular      16    boy    no       white       69
#> 5      489       447  small.class  5     boy    yes      white       79
#> 6    454       431  regular       8     boy    yes      white       5
```

С помощью пакета `readxl` можно делать еще многое, например считывать файлы **.xls** или **.xlsm**, а также отдельные рабочие страницы или диапазоны рабочих книг. Для более подробного изучения обратитесь к документации к этому пакету

Исследование кадра данных

Ранее вы познакомились с функциями `head()` и `str()`, используемыми для определения размеров кадра (рамки) данных. Но есть еще несколько полезных функций. Прежде всего это функция `View()` из RStudio, выходные данные которой будут весьма полезны вам как пользователю Excel:

```
View(star)
```

После вызова этой функции в новом окне на панели **Scripts** (Скрипты) появится средство просмотра, похожее на электронную таблицу. Вы можете сортировать, фильтровать и изучать наборы данных так же, как и в Excel. Однако в соответствии со своим названием[1] функция `View()` предназначена *только* для просмотра. В этом окне вы не можете изменять данные.

Другой способ распечатать несколько записей кадра данных вместе с названиями и типами столбцов — функция `glimpse()`. Эта функция поставляется с пакетом `dplyr`, который входит в `tidyverse`. Мы будем активно использовать пакет `dplyr` в последующих главах для управления данными.

```
glimpse(star)
#> Rows: 5,748
#> Columns: 8
#> $ tmathssk <dbl> 473, 536, 463, 559, 489,...
#> $ treadssk <dbl> 447, 450, 439, 448, 447,...
#> $ classk   <chr> "small.class", "small.cl...
#> $ totexpk  <dbl> 7, 21, 0, 16, 5, 8, 17, ...
```

[1] *View* — просмотр. — *Ред.*

```
#> $ sex      <chr> "girl", "girl", "boy", "...
#> $ freelunk <chr> "no", "no", "yes", "no",...
#> $ race     <chr> "white", "black", "black...
#> $ schidkn  <dbl> 63, 20, 19, 69, 79, 5, 1...
```

В базовом R также существует функция `summary()`, которая создает сводки по различным объектам R. Когда кадр данных передается в функцию `summary()`, предоставляется некоторая описательная статистика:

```
summary(star)
#>     tmathssk        treadssk       classk            totexpk
#> Min.   :320.0   Min.   :315.0   Length:5748      Min.   : 0.000
#> 1st Qu.:454.0   1st Qu.:414.0   Class :character 1st Qu.: 5.000
#> Median :484.0   Median :433.0   Mode :character  Median : 9.000
#> Mean   :485.6   Mean   :436.7                    Mean : 9.307
#> 3rd Qu.:513.0   3rd Qu.:453.0                    3rd Qu.:13.000
#> Max.   :626.0 Max.      :627.0                   Max. :27.000
#>     sex               freelunk           race
#> Length:5748        Length:5748      Length:5748
#> Class :character   Class :character Class :character
#> Mode :character    Mode  :character Mode  :character
#>    schidkn
#> Min.   : 1.00
#> 1st Qu.:20.00
#> Median :39.00
#> Mean   :39.84
#> 3rd Qu.:60.00
#> Max.   :80.00
```

Множество других пакетов включают в себя собственные версии описательной статистики; мне больше всего нравится функция `describe()` из пакета `psych`:

```
describe(star)
#>           vars    n   mean     sd median  trimmed    mad min max range  skew
#> tmathssk     1 5748 485.65  47.77    484   483.20  44.48 320 626   306  0.47
#> treadssk     2 5748 436.74  31.77    433   433.80  28.17 315 627   312  1.34
#> classk*      3 5748   1.95   0.80      2     1.94   1.48   1   3     2  0.08
#> totexpk      4 5748   9.31   5.77      9     9.00   5.93   0  27    27  0.42
#> sex*         5 5748   1.49   0.50      1     1.48   0.00   1   2     1  0.06
#> freelunk*    6 5748   1.48   0.50      1     1.48   0.00   1   2     1  0.07
#> race*        7 5748   2.35   0.93      3     2.44   0.00   1   3     2 -0.75
#> schidkn      8 5748  39.84  22.96     39    39.76  29.65   1  80    79  0.04
#>           kurtosis   se
#> tmathssk     0.29 0.63
#> treadssk     3.83 0.42
#> classk*     -1.45 0.01
#> totexpk     -0.21 0.08
#> sex*        -2.00 0.01
#> freelunk*   -2.00 0.01
```

```
#> race*          -1.43 0.01
#> schidkn        -1.23 0.30
```

Если вам незнакома вся эта описательная статистика, вы знаете, что делать: изучите документацию к функции.

Индексирование и подмножества кадров данных

Ранее в этой главе мы создали небольшой кадр данных `roster`, содержащий сведения об именах и росте пятерых человек (Джек, Джилл, Билли, Сьюзи и Джонни). Сейчас, используя этот объект, мы увидим основные методы управления кадром данных.

В Excel для того, чтобы сослаться на позиции строк и столбцов в таблице, можно использовать функцию `INDEX()`, как показано на рис. 7.4:

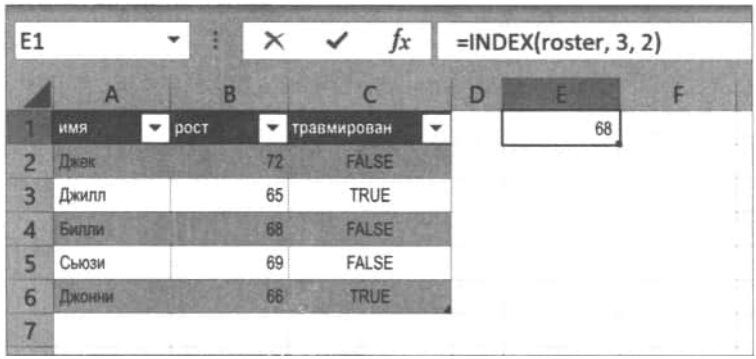


Рис. 7.4. Функция INDEX() в таблице Excel

В R это работает аналогичным образом. Мы будем использовать такую же запись со скобками, что и для индексных векторов, но на этот раз будем ссылаться на позиции строк и столбцов:

```
# Третья строка, второй столбец кадра данных
roster[3, 2]
#> [1] 68
```

И снова мы можем использовать оператор :, чтобы извлечь все элементы в данном диапазоне:

```
# От второй по четвертую строки, от первого по  третий столбец
roster[2:4, 1:3]
#> name height injured
#> 2 Jill  65 TRUE
#> 3 Billy 68 FALSE
#> 4 Susie 69 FALSE
```

Также можно выбрать целую строку или столбец, оставив их индекс пустым, или использовать функцию `c()` для выбора подмножества непоследовательных элементов:

```
# Только вторая и третья строки
roster[2:3,]
#>      name height injured
#> 2    Jill     65    TRUE
#> 3   Billy     68   FALSE

# Только первый и третий столбцы
roster[, c(1,3)]
#>     name injured
#> 1   Jack   FALSE
#> 2   Jill    TRUE
#> 3  Billy   FALSE
#> 4  Susie   FALSE
#> 5 Johnny    TRUE
```

Если нужен доступ только к одному столбцу кадра данных, можно использовать оператор `$`. Интересно, что результатом будет *вектор*:

```
roster$height
#> [1] 72 65 68 69 66
is.vector(roster$height)
#> [1] TRUE
```

Все это подтверждает, что кадр данных на самом деле является списком векторов одинаковой длины.

Другие структуры данных в R

Мы сосредоточились на векторах и кадрах данных R — структурах, эквивалентных диапазонам и таблицам в Excel, с которыми вы, скорее всего, будете работать в процессе анализа данных. Однако в базовом R существуют и другие структуры данных, такие как *матрицы* и *списки*. Чтобы узнать больше об этих структурах и их связях с векторами и кадрами данных, обратитесь к книге Хэдли Уикхэма «Расширенный R» (Hadley Wickham. Advanced R).

Запись кадров данных

Как уже упоминалось, принято считывать данные в R, обрабатывать их и затем экспортировать результаты куда-то еще. Чтобы записать кадр данных в файл **.csv**, можно использовать функцию `write_csv()` из пакета `readr`:

```
# Запишите кадр данных roster в csv
write_csv(roster, 'output/roster-output-r.csv')
```

Если у вас установлен рабочий каталог из репозитория на GitHub, вы должны найти этот файл в папке **output**.

К сожалению, пакет `readxl` не содержит функции, позволяющей записать данные в рабочую книгу Excel. Но можно использовать пакет `writexl` и его функцию `write_xlsx()`:

```
# Запишите кадр данных roster в csv
write_xlsx(roster, 'output/roster-output-r.xlsx')
```

Заключение

В этой главе вы перешли от элементарных объектов к бóльшим, векторам и кадрам данных. Хотя мы будем работать с кадрами данных до конца книги, полезно помнить, что они являются наборами векторов и ведут себя в основном так же. Далее вы узнаете, как анализировать, визуализировать и, наконец, тестировать связи в кадрах данных R.

Упражнения

Выполните следующие упражнения, чтобы проверить свои знания о структурах данных в R.

1. Создайте символьный вектор из пяти элементов, затем получите доступ к его первому и четвертому элементам.
2. Создайте два вектора, x и y, имеющих длину 4, один из которых содержит числовые, а другой — логические значения. Перемножьте их и передайте результат в `z`. Какой результат вы получили?
3. Загрузите пакет `nycflights13` из CRAN. Сколько наборов данных включено в этот пакет?
   - Один из этих наборов данных называется **airports** (аэропорты). Распечатайте несколько первых строк из этого кадра данных, а также описательную статистику.
   - Другой набор имеет название **weather** (погода). Найдите строки с 10-й по 12-ю и столбцы с 4-го по 7-й. Запишите результаты в файл **.csv** и в рабочую книгу Excel.

ГЛАВА 8

Обработка и визуализация данных в R

Американский статистик Рональд Тистед однажды пошутил: «Необработанные данные подобны сырому картофелю — перед использованием они обычно требуют очистки». Обработка данных требует времени, и вы наверняка испытывали сложности, если когда-либо вам приходилось выполнять следующие действия.

- Выбор, удаление или создание вычисляемых столбцов.
- Сортировка или фильтрация строк.
- Группировка и суммирование по категориям.
- Объединение нескольких наборов данных общим полем.

Скорее всего, всё это вы делали в Excel... и *много* раз, и вам, вероятно, пришлось покопаться в знаменитой функции `VLOOKUP()` и в сводных таблицах, чтобы выполнить эти задачи. В настоящей главе вы познакомитесь с эквивалентами этих методов в R, в частности со справочной системой (**Help**) пакета `dplyr`.

Обработка данных часто идет рука об руку с визуализацией: как уже упоминалось, люди прекрасно справляются с визуальной обработкой информации, и это отличный способ оценить набор данных. Вы научитесь визуализировать данные с помощью прекрасного пакета `ggplot2`, который, так же как и пакет `dplyr`, является частью библиотеки `tidyverse`. Это обеспечит вам прочную основу для исследования и проверки взаимосвязей между данными с помощью R — подробнее вы узнаете об этом в *главе 9*. Мы начнем с вызова соответствующих пакетов; будем использовать набор данных **star**, который находится в репозитории к данной книге, поэтому импортируем его прямо сейчас:

```
library(tidyverse)
library(readxl)

star <- read_excel('datasets/star/star.xlsx')
head(star)
#> # Тиббл: 6 x 8
#> tmathssk treadssk classk        totexpk  sex   freelunk  race   schidkn
```

Обработка данных с помощью пакета *dplyr*

`dplyr` — это популярный пакет, предназначенный для обработки табличных структур данных. Многие его функции, или *глаголы*, работают одинаково и легко могут

использоваться вместе. В табл. 8.1 указаны некоторые часто используемые функции пакета `dplyr` и их назначение (все они будут рассмотрены в данной главе).

Таблица 8.1. *Часто используемые функции пакета* ***dplyr***

Функция	Назначение
`select()`	Выбрать заданные столбцы
`mutate()mutate()`	Создать новые столбцы на основе существующих
`rename()`	Переименовать заданные столбцы
`arrange()`	Упорядочить строки по заданному критерию
`filter()`	Выбрать строки по заданному критерию
`group_by()`	Сгруппировать строки по заданным столбцам
`summarize()`	Суммировать значения для каждой группы
`left_join()`	Включить соответствующие строки из таблицы B в таблицу A; если в таблице B соответствие не найдено, результатом будет NA[1]

Для краткости я не буду описывать все функции пакета `dplyr` или все способы использования тех функций, которые мы рассмотрим. Чтобы лучше познакомиться с пакетом dplyr, прочтите книгу Хэдли Уикхема и Гаррета Гролемунда «R для науки о данных» (Hadley Wickham, Garrett Grolemund. R for Data Science). Также вы можете получить доступ к полезным подсказкам, описывающим, как множество функций пакета `dplyr` работают друг с другом, выбрав в RStudio последовательно **Help** (Справка) | **Cheatsheets** (Подсказки) | **Data Transformation with dplyr** (Преобразование данных с dplyr).

Постолбцовые операции

Выбор и исключение столбцов в Excel часто требует их скрытия или удаления. Но проверять или воспроизводить их достаточно трудно, поскольку скрытые столбцы можно легко пропустить, а удаленные — нелегко восстановить. В R для выбора заданных столбцов из кадра данных можно использовать функцию `select()`. Первый аргумент этой функции `select()`, как и в остальных из этих функций, указывает, с каким кадром данных надо работать. Затем следуют дополнительные аргументы для управления нужными данными в кадре данных. Например, можно выбрать столбцы **tmathssk**, **treadssk** и **schidkin** из набора данных **star** следующим образом:

```
select(star, tmathssk, treadssk, schidkn)
#> # Тиббл: 5,748 x 3
#>   tmathssk  treadssk  schidkn
#>      <dbl>     <dbl>    <dbl>
#> 1      473       447       63
#> 2      536       450       20
```

[1] *NA* — специальная величина в R, указывающая на отсутствие значения. — *Ред.*

```
#> 3      463       439       19
#> 4      559       448       69
#> 5      489       447       79
#> 6      454       431        5
#> 7      423       395       16
#> 8      500       451       56
#> 9      439       478       11
#> 10     528       455       66
#> # ... и еще 5,738 строк
```

Также можно использовать оператор - с функцией `select()`, чтобы *исключить* заданные столбцы:

```
select(star, -tmathssk, treadssk, schidkn)
#> # Тиббл: 5,748 x 5
#>   classk           totexpk   sex   freelunk     race
#>   <chr>              <dbl>  <chr>  <chr>        <chr>
#> 1 small.class            7  girl   no           white
#> 2 small.class           21  girl   no           black
#> 3 regular.with.aide      0  boy    yes          black
#> 4 regular               16  boy    no           white
#> 5 small.class            5  boy    yes          white
#> 6 regular                8  boy    yes          white
#> 7 regular.with.aide     17  girl   yes          black
#> 8 regular                3  girl   no           white
#> 9 small.class           11  girl   no           black
#> 10 small.class          10  girl   no           white
```

Более элегантная альтернатива — передача всех нежелательных столбцов в вектор, а *затем* его удаление:

```
select(star, -c(tmathssk, treadssk, schidkn))
#> # Тиббл: 5,748 x 5
#>   classk           totexpk   sex   freelunk     race
#>   <chr>              <dbl>  <chr>  <chr>        <chr>
#> 1 small.class            7  girl   no           white
#> 2 small.class           21  girl   no           black
#> 3 regular.with.aide      0  boy    yes          black
#> 4 regular               16  boy    no           white
#> 5 small.class            5  boy    yes          white
#> 6 regular                8  boy    yes          white
#> 7 regular.with.aide     17  girl   yes          black
#> 8 regular                3  girl   no           white
#> 9 small.class           11  girl   no           black
#> 10 small.class          10  girl   no           white
```

Имейте в виду, что в приведенных примерах мы только вызывали функции, а не присваивали выходные данные объекту.

Еще один способ сокращения записи для функции `select()` – использование оператора :, чтобы выбрать все данные между двумя столбцами включительно. На этот

раз я присвою результаты выборки всех данных между столбцами **tmathssk** и **totexpk** обратно объекту **star**:

```
star <- select(star, tmathssk:totexpk)
head(star)
#> # Тиббл: 6 x 4
#>   tmathssk treadssk  classk        totexpk
#>      <dbl>    <dbl> <chr>           <dbl>
#> 1      473      447  small.class         7
#> 2      536      450  small.class        21
#> 3      463      439  regular.with.aide   0
#> 4      559      448  regular            16
#> 5      489      447  small.class         5
#> 6      454      431  regular             8
```

Вероятно, вы создавали вычисляемые столбцы в Excel; в R то же самое делает функция `mutate()`. Давайте создадим столбец **new_column**, объединяющий баллы по чтению и математике. *Сначала* с помощью функции `mutate()` укажем имя нового столбца, затем знак равенства и, наконец, вычисление. Мы можем сослаться на другие столбцы как на часть этой формулы:

```
star <- mutate(star, new_column = tmathssk + treadssk)
head(star)
#> # Тиббл: 6 x 5
#>   tmathssk  treadssk  classk         totexpk  new_column
#>      <dbl>     <dbl> <chr>            <dbl>       <dbl>
#> 1      473       447  small.class          7         920
#> 2      536       450  small.class         21         986
#> 3      463       439  regular.with.aide    0         902
#> 4      559       448  regular             16        1007
#> 5      489       447  small.class          5         936
#> 6      454       431  regular              8         885
```

Функция `mutate()` позволяет легко получать более сложные вычисляемые столбцы, такие как логарифмическое преобразование или запаздывающие (лаговые) переменные; более подробную информацию можно найти в справочной документации.

Название **new_column** на самом деле не очень подходит для итоговой суммы баллов. К счастью, есть функция `rename()`, которая позволяет присвоить нужное имя. Укажем, какое имя надо дать новому столбцу вместо старого:

```
star <- rename(star, ttl_score = new_column)
head(star)
#> # Тиббл: 6 x 5
#>   tmathssk  treadssk  classk         totexpk  new_column
#>      <dbl>     <dbl> <chr>            <dbl>       <dbl>
#> 1      473       447  small.class          7         920
#> 2      536       450  small.class         21         986
#> 3      463       439  regular.with.aide    0         902
#> 4      559       448  regular             16        1007
```

```
#> 5      489       447  small.class          5         936
#> 6      454       431  regular              8         885
```

Построчные операции

До сих пор мы работали со *столбцами*. Теперь перейдем к операциям со *строками*; а именно — к сортировке и фильтрации. В Excel можно выполнять сортировку по нескольким столбцам с помощью меню **Custom Sort** (Настраиваемая сортировка). Допустим, мы хотим отсортировать наш кадр данных по столбцу **classk**, затем по столбцу **treadssk**, в обоих случаях по возрастанию. В Excel это выглядело бы так, как показано на рис. 8.1.

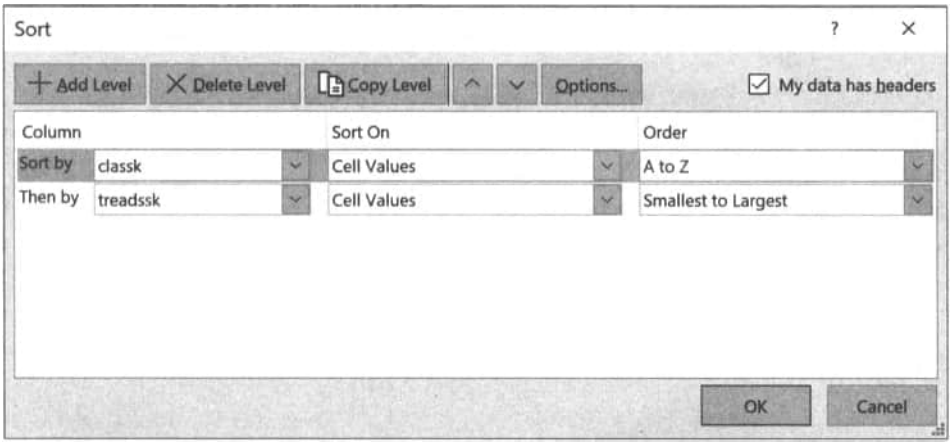


Рис. 8.1. Меню **Custom Sort** в Excel

Это можно воспроизвести с помощью пакета `dplyr`, используя функцию `arrange()`, указывая каждый столбец в том порядке, в котором мы хотим отсортировать кадр данных:

```
arrange(star, classk, treadssk)
#> # Тиббл: 5,748 x 5
#> tmathssk  treadssk  classk   totexpk  ttl_score
#>    <dbl>     <dbl>  <chr>      <dbl>      <dbl>
#> 1    320       315  regular        3        635
#> 2    365       346  regular        0        711
#> 3    384       358  regular       20        742
#> 4    384       358  regular        3        742
#> 5    320       360  regular        6        680
#> 6    423       376  regular       13        799
#> 7    418       378  regular       13        796
#> 8    392       378  regular       13        770
#> 9    392       378  regular        3        770
#> 10   399       380  regular        6        779
#> # ... и еще 5,738 строк
```

Если необходимо отсортировать столбец по убыванию, можно передать этому столбцу функцию `desc()`.

```
# Сортировать classk по убыванию, treadssk по возрастанию
arrange(star, desc(classk), treadssk)
#> # Тиббл: 5,748 x 5
#> tmathssk treadssk classk       totexpk ttl_score
#>    <dbl>    <dbl> <chr>          <dbl>     <dbl>
#> 1    412      370 small.class       15       782
#> 2    434      376 small.class       11       810
#> 3    423      378 small.class        6       801
#> 4    405      378 small.class        8       783
#> 5    384      380 small.class       19       764
#> 6    405      380 small.class       15       785
#> 7    439      382 small.class        8       821
#> 8    384      384 small.class       10       768
#> 9    405      384 small.class        8       789
#> 10   423      384 small.class       21       807
```

В таблицах Excel имеются полезные раскрывающиеся меню для фильтрации любого столбца по заданным критериям. Чтобы отфильтровать кадр данных в R, используем соответствующую функцию `filter()`. Давайте отфильтруем кадр данных **star** так, чтобы оставить только записи, в которых значение переменной **classk** равно `small.class`. Помните, что поскольку мы проверяем равенство, а не присваиваем объект, необходимо использовать оператор `==`, а не `=`:

```
filter(star, classk == 'small.class')
#> # Тиббл: 1,733 x 5
#>  tmathssk treadssk classk      totexpk ttl_score
#>     <dbl>    <dbl> <chr>        <dbl>     <dbl>
#>  1      473      447 small.class      7       920
#>  2      536      450 small.class     21       986
#>  3      489      447 small.class      5       936
#>  4      439      478 small.class     11       917
#>  5      528      455 small.class     10       983
#>  6      559      474 small.class      0      1033
#>  7      494      424 small.class      6       918
#>  8      478      422 small.class      8       900
#>  9      602      456 small.class     14      1058
#> 10      439      418 small.class      8       857
#> #... и еще 1,723 строки
```

В тиббле выходных данных мы видим, что операция `filter()` влияет *только* на количество строк, но не столбцов. Теперь давайте найдем записи, в которых значение `treadssk` *не менее* 500:

```
filter(star, treadssk >= 500)
#> # Тиббл: 233 x 5
```

```
#>   tmathssk treadssk classk          totexpk  ttl_score
#>      <dbl>    <dbl> <chr>              <dbl>      <dbl>
#>  1     559      522 regular                8       1081
#>  2     536      507 regular.with.aide      3       1043
#>  3     547      565 regular.with.aide      9       1112
#>  4     513      503 small.class            7       1016
#>  5     559      605 regular.with.aide      5       1164
#>  6     559      554 regular               14       1113
#>  7     559      503 regular               10       1062
#>  8     602      518 regular               12       1120
#>  9     536      580 small.class           12       1116
#> 10     626      510 small.class           14       1136
#> #... и еще 223 строки
```

Можно фильтровать по нескольким критериям, используя оператор `&` как «и» и оператор `|` как «или». Давайте объединим два критерия из предыдущего примера с помощью оператора &:

```
# Получить записи, в которых classk равен small.class и
# treadssk не менее 500
filter(star, classk == 'small.class' & treadssk >= 500)
#> # Тиббл: 84 x 5
#>   tmathssk treadssk classk      totexpk  ttl_score
#>      <dbl>    <dbl> <chr>          <dbl>      <dbl>
#>  1     513      503 small.class        7       1016
#>  2     536      580 small.class       12       1116
#>  3     626      510 small.class       14       1136
#>  4     602      518 small.class        3       1120
#>  5     626      565 small.class       14       1191
#>  6     602      503 small.class       14       1105
#>  7     626      538 small.class       13       1164
#>  8     500      580 small.class        8       1080
#>  9     489      565 small.class       19       1054
#> 10     576      545 small.class       19       1121
#> # ... и еще 74 more строки
```

Агрегирование и объединение данных

Мне нравится называть сводные таблицы «WD-40 от Excel»[2], т. к. они позволяют «вертеть» данные в различных направлениях для облегчения их анализа. Например, давайте восстановим сводную таблицу, показывающую средний балл по математике в зависимости от размера класса из набора данных **star** (рис. 8.2).

Как видно из рис. 8.2, в сводной таблице есть два элемента. Сначала я сгруппировал данные по переменной **classk**. Затем свел данные, взяв среднее значение

[2] *WD-40* — американская компания и торговая марка известного аэрозольного препарата, первоначально разработанного как водоотталкивающее средство, предотвращающее коррозию; позже обнаружилось множество других полезных свойств и возможностей применения. — *Ред.*

Рис. 8.2. Работа сводных таблиц в Excel

tmathssk. В R это дискретные шаги, выполняемые с помощью разных функций `dplyr`. Прежде всего мы сгруппируем данные с помощью функции `group_by()`. Выходные данные включают строку `# Groups: classk [3]`, указывающую, что `star_grouped` разбито на три группы по переменной **classk**:

```
star_grouped <- group_by(star, classk)
head(star_grouped)
#> # Тиббл: 6 x 5
#> # Groups:  classk [3]
#>  tmathssk treadssk classk          totexpk  ttl_score
#>     <dbl>   <dbl> <chr>              <dbl>      <dbl>
#> 1      473     447 small.class            7        920
#> 2      536     450 small.class           21        986
#> 3      463     439 regular.with.aide      0        902
#> 4      559     448 regular               16       1007
#> 5      489     447 small.class            5        936
#> 6      454     431 regular                8        885
```

Мы *сгруппировали* данные по одной переменной; теперь давайте сведем данные по другой переменной с помощью функции `summarize()` (`summarise()` тоже будет работать). Теперь укажем, как назвать результирующий столбец и как его вычислить. В табл. 8.2 перечислены некоторые популярные агрегатные функции.

Таблица 8.2.** Полезные агрегатные функции пакета **dplyr

Функция	Тип агрегирования
`sum()`	Сумма
`n()`	Количество значений
`mean()`	Среднее значение
`max()`	Максимальное значение
`min()`	Минимальное значение
`sd()`	Стандартное отклонение

Мы можем получить средний балл по математике в зависимости от размера класса, выполнив функцию `sum()` для сгруппированного кадра данных:

```
summarize(star_grouped, avg_math = mean(tmathssk))
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # Тиббл: 3 x 2
#>  classk          avg_math
#>  <chr>              <dbl>
#> 1 regular            483.
#> 2 regular.with.aide  483.
#> 3 small.class        491.
```

Ошибка `summarise()`ungrouping output является предупреждением, означающим, что вы разгруппировали сгруппированный тиббл путем его агрегирования. За вычетом некоторых различий в форматировании мы имеем те же результаты, что и на рис. 8.2.

Если сводные таблицы — это WD-40 в Excel, то VLOOKUP() — это «скотч», позволяющий легко объединять, связывать данные из разных источников. В исходном наборе данных **star** переменная **schidkin** является индикатором школьного округа. Ранее мы опустили этот столбец, так что давайте снова обратим на него внимание. Но как быть, если в дополнение к номеру индикатора мы хотели бы знать названия школьных округов? К счастью, эта информация имеется в файле **districts.csv** в репозитории на GitHub — давайте считаем ее и определим стратегию объединения данных:

```
star <- read_excel('datasets/star/star.xlsx')
head(star)
#> # Тиббл: 6 x 8
#>  tmathssk treadssk classk          totexpk sex   freelunk race  schidkn
#>     <dbl>   <dbl> <chr>              <dbl> <chr> <chr>   <chr>  <dbl>
#> 1      473     447 small.class            7 girl  no      white     63
#> 2      536     450 small.class           21 girl  no      black     20
#> 3      463     439 regular.with.aide      0 boy   yes     black     19
#> 4      559     448 regular               16 boy   no      white     69
#> 5      489     447 small.class            5 boy   yes     white     79
#> 6      454     431 regular                8 boy   yes     white      5

districts <- read_csv('datasets/star/districts.csv')

#> -- Column specification --------------------------------------------
#> cols(
#>  schidkn = col_double(),
#>  school_name = col_character(),
#>  county = col_character()
#> )

head(districts)
#> # Тиббл: 6 x 3
#>  schidkn school_name      county
#>    <dbl> <chr>           <chr>
#> 1       1 Rosalia         New Liberty
#> 2       2 Montgomeryville Topton
```

```
#> 3      3 Davy           Wahpeton
#> 4      4 Steelton       Palestine
#> 5      6 Tolchester     Sattley
#> 6      7 Cahokia        Sattley
```

Похоже, нужно нечто вроде функции `VLOOKUP()`: мы хотим «считать» переменную **school_name** (название школы) (и, возможно, переменную **county** (округ)) из набора данных **districts** в **star**, с учетом общей переменной **schidkn**. Чтобы выполнить это в R, мы будем использовать принцип объединения, пришедший из реляционных баз данных, — тема, которую мы затронули в *главе 5*. Ближе всего к функции *VLOOKUP()* левое внешнее соединение, которое в пакете `dplyr` может быть выполнено с помощью функции `left_join()`. Сначала мы представим «базовую» таблицу (**star**), а затем — «поисковую» таблицу (**districts**). Функция будет искать и возвращать соответствия в таблице **districts** для каждой записи в **star** или, если соответствие не найдено, возвращать NA. Чтобы не перегружать вывод консоли, я оставил только некоторые столбцы из набора **star**:

```
# Левое внешнее соединение star по districts
left_join(select(star, schidkn, tmathssk, treadssk), districts)
#> Joining, by = "schidkn"
#> # Тиббл: 5,748 x 5
#>   schidkn tmathssk treadssk school_name    county
#>     <dbl>   <dbl>   <dbl> <chr>           <chr>
#>  1      63     473     447 Ridgeville      New Liberty
#>  2      20     536     450 South Heights   Selmont
#>  3      19     463     439 Bunnlevel       Sattley
#>  4      69     559     448 Hokah           Gallipolis
#>  5      79     489     447 Lake Mathews    Sugar Mountain
#>  6       5     454     431 NA              NA
#>  7      16     423     395 Calimesa        Selmont
#>  8      56     500     451 Lincoln Heights Topton
#>  9      11     439     478 Moose Lake      Imbery
#> 10      66     528     455 Siglerville     Summit Hill
#> # ... и еще 5,738 строк
```

Функция `left_join()` достаточно умна: она знала, что надо выполнять объединение по значению **schidkn** и искала не только название школы (**school_name**), но и **округ** (county). Чтобы узнать больше об объединении данных, ознакомьтесь со справочной документацией.

Отсутствующее наблюдение в R представлено специальной величиной `NA`. Например, похоже, что не найдено соответствий для названия округа 5. В `VLOOKUP()` это привело бы к ошибке `#N/A`. NA *не означает*, что наблюдение равно нулю, а лишь только то, что его значение отсутствует. При R-программировании вы столкнетесь и с другими специальными величинами, такими как `NaN` или `NULL`; чтобы узнать о них больше, обратитесь к справочной документации.

dplyr и оператор *pipe* (%>%)

Как видите, функции пакета `dplyr` достаточно мощные и интуитивно понятны для тех, кто работает с данными, в том числе в Excel. И каждый, кто работал с данными, знает, что редко удается подготовить данные должным образом за один шаг. К примеру, возьмем типичную задачу, которую вы можете выполнить с набором **star**.

> Найти средний балл по чтению с учетом типа класса с сортировкой от большего к меньшему.

Зная, что́ нужно сделать с данными, можно разбить задачу на три отдельных шага:

1. Сгруппировать данные по типу класса.
2. Найти средний балл по чтению для каждой группы.
3. Отсортировать результаты от большего к меньшему.

Это можно сделать с помощью пакета `dplyr` следующим образом:

```
star_grouped <- group_by(star, classk)
star_avg_reading <- summarize(star_grouped, avg_reading = mean(treadssk))

#> `summarise()` ungrouping output (override with `.groups` argument)
#>
star_avg_reading_sorted <- arrange(star_avg_reading, desc(avg_reading))
star_avg_reading_sorted
#>
#> # Тиббл: 3 x 2
#>   classk          avg_reading
#>   <chr>                 <dbl>
#> 1 small.class             441.
#> 2 regular.with.aide       435.
#> 3 regular                 435.
```

Мы получили ответ, но для этого потребовалось довольно много шагов, к тому же можно легко запутаться в различных функциях и именах объектов. Альтернатива — объединить эти функции с помощью оператора `%>%`, называемого также «трубой» (pipe). Это позволит передавать выходные данные одной функции непосредственно в другую, так что мы будем избавлены от необходимости постоянно переименовывать входные и выходные данные. Клавиатурная комбинация по умолчанию для этого оператора <Ctrl>+<Shift>+<M> для Windows и <Cmd>+ +<Shift>+<M> для Mac.

Давайте воспроизведем предыдущие шаги, но на этот раз с помощью оператора `pipe`. Разместим каждую функцию в отдельной строке, объединив их с помощью `%>%`. Хотя нет необходимости помещать каждый шаг на отдельной строке, но лучше это делать для удобства чтения. Также при использовании оператора pipe необязательно выделять весь блок кода для его выполнения; просто поместите курсор в любом месте следующего фрагмента и выполните:

```
 star %>%
  group_by(classk) %>%
  summarise(avg_reading = mean(treadssk)) %>%
  arrange(desc(avg_reading))
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # Тиббл: 3 x 2
#>  classk          avg_reading
#>  <chr>                 <dbl>
#> 1 small.class           441.
#> 2 regular.with.aide     435.
#> 3 regular               435.
```

Поначалу может быть непривычно, что не нужно явно включать источник данных в качестве аргумента в каждой функции. Но сравните последний блок кода с предыдущим, и вы увидите, насколько эффективнее этот подход. Более того, оператор pipe можно использовать с функциями, не принадлежащими пакету `dplyr`. Например, давайте назначим первые несколько строк результирующей операции, включив функцию `head()` в конце pipe:

```
# Средний балл по математике и чтению
# для каждого школьного округа
star %>%
  group_by(schidkn) %>%
  summarise(avg_read = mean(treadssk), avg_math = mean(tmathssk)) %>%
  arrange(schidkn) %>%
  head()
#> `summarise()` ungrouping output (override with `.groups` argument)
#> # Тиббл: 6 x 3
#>  schidkn avg_read avg_math
#>    <dbl>   <dbl>    <dbl>
#> 1       1    444.     492.
#> 2       2    407.     451.
#> 3       3    441      491.
#> 4       4    422.     468.
#> 5       5    428.     460.
#> 6       6    428.     470.
```

Преобразование данных с помощью *tidyr*

Несмотря на то что функция `group_by()` совместно с функцией `summarize()` является в R эквивалентом сводных таблиц, эти функции не способны делать все, что может делать в Excel сводная таблица. А если вместо того, чтобы просто агрегировать данные, мы хотим *преобразовать* их, т. е. изменить организацию строк и столбцов? Например, кадр данных **star** имеет два отдельных столбца для баллов по математике и чтению — **tmathssk** и **treadssk** соответственно. Я хочу объединить их в один столбец с названием **score** (балл), и еще с одним, под названием **test_type**, указывающим, относится ли каждое наблюдение к математике или чтению. Также я хочу сохранить индикатор школы **schidkn** как часть анализа.

На рис. 8.3 показано, как это может выглядеть в Excel. Обратите внимание: я переименовал поле **Values** (значения) из **tmathssk** и **treadssk** в **math** и **reading** соответственно. Если вы хотите более подробно ознакомиться с этой сводной таблицей, ее можно найти в репозитории к книге, файл **ch-8.xlsx**. Здесь я снова использую индексный столбец; в противном случае сводная таблица будет пытаться «свернуть» все значения по величине **schidkn**.

	A	B	C	D
1				
2				
3	id	schidkn	Values	Total
4	1	63	reading	447
5	1	63	math	473
6	2	20	reading	450
7	2	20	math	536
8	3	19	reading	439
9	3	19	math	463
10	4	69	reading	448
11	4	69	math	559
12	5	79	reading	447

Рис. 8.3. Преобразование набора данных **star** в Excel

Для преобразования набора данных **star** можно использовать основной пакет `tidyr` библиотеки `tidyverse`. Добавление индексного столбца будет полезно при преобразовании в R, так же как и в Excel. Мы можем создать его с помощью функции `row_number()`:

```
star_pivot <- star %>%
              select(c(schidkn, treadssk, tmathssk)) %>%
              mutate(id = row_number())
```

Чтобы преобразовать кадр данных, будем использовать функции `pivot_longer()` и `pivot_wider()` из пакета `tidyr`. Посмотрите на рис. 8.3 и представьте, что произойдет с набором данных, если мы объединим баллы из столбцов **tmathssk** и **treadssk** в один столбец? Станет ли набор данных длиннее или шире? Поскольку мы добавляем строки, набор данных станет длиннее. Чтобы использовать функцию `pivot_longer()`, укажем с помощью аргумента `cols`, по каким столбцам надо удлинить набор, и используем функцию `values_to`, чтобы дать имя результирующему столбцу. Также применим функцию `names_to` для присвоения имени столбцу, указывающему, относится ли каждый балл к математике или чтению:

```
star_long <- star_pivot %>%
             pivot_longer(cols = c(tmathssk, treadssk),
                          values_to = 'score', names_to = 'test_type')
head(star_long)
#> # Тиббл: 6 x 4
#>   schidkn    id test_type score
#>     <dbl> <int> <chr>     <dbl>
#> 1      63     1 tmathssk    473
#> 2      63     1 treadssk    447
```

```
#> 3     20    2 tmathssk   536
#> 4     20    2 treadssk   450
#> 5     19    3 tmathssk   463
#> 6     19    3 treadssk   439
```

Отлично. Но существует ли способ переименовать **tmathssk** и **treadssk** в **math** и **reading** соответственно? Да, существует, с помощью функции `recode()` — еще одной полезной функции пакета `dplyr`, которую можно использовать вместе с функцией `mutate()`. Функция `recode()` работает несколько иначе, чем другие функции этого пакета: названия «старых» значений ставятся *перед* знаком равенства, а новых — за ним. Функция `distinct()` из пакета `dplyr` подтвердит, что все строки были названы либо **math**, либо **reading**:

```
# Переименовать tmathssk и treadssk в math and reading
star_long <- star_long %>%
   mutate(test_type = recode(test_type,
                             'tmathssk' = 'math', 'treadssk' = 'reading'))
distinct(star_long, test_type)
#> # Тиббл: 2 x 1 #>  test_type
#>  <chr> #> 1 math
#> 2 reading
```

Теперь, когда кадр данных удлинен, можно сделать его шире с помощью функции `pivot_wider()`. На это раз с помощью функции `values_from` я укажу, какой столбец содержит в своих строках значения, которые должны быть столбцами, и с помощью функции `names_from` — как должны называться результирующие столбцы:

```
star_wide <- star_long %>%
               pivot_wider(values_from = 'score', names_from = 'test_type')
head(star_wide)
#> # Тиббл: 6 x 4
#>  schidkn   id  math reading
#>    <dbl> <int> <dbl>  <dbl>
#> 1     63     1  473    447
#> 2     20     2  536    450
#> 3     19     3  463    439
#> 4     69     4  559    448
#> 5     79     5  489    447
#> 6      5     6  454    431
```

Преобразование данных в R — достаточно сложная операция, поэтому, если вы сомневаетесь, спросите себя: *я делаю эти данные шире или длиннее? Как я сделал бы это в сводной таблице?* Если вы можете логически прикинуть, что нужно для достижения желаемого конечного состояния, программирование будет намного проще.

Визуализация данных с помощью *ggplot2*

Пакет `dplyr` может сделать очень многое, чтобы помочь нам обрабатывать данные, но сейчас давайте обратим внимание на визуализацию данных. А именно: рассмотрим другой пакет библиотеки `tidyverse` — `ggplot2`. Названный и созданный в соответствии с «грамматикой графики» ученого-компьютерщика Леланда Уилкинсона (Leland Wilkinson), пакет `ggplot2` предлагает упорядоченный подход к построению графиков. Эта структура основана на принципах объединения элементов речи в предложения, отсюда и название — «грамматика» графики.

Здесь я расскажу о некоторых основных элементах и типах графиков пакета `ggplot2`. Дополнительную информацию об этом пакете можно получить, прочитав книгу автора идеи пакета Хэдли Уикхэма «ggplot2: элегантная графика для анализа данных» (Hadley Wickham. ggplot2: Elegant Graphics for Data Analysis). Также можно почерпнуть полезные сведения, выбрав в RStudio последовательно **Help** (Справка) | **Cheatsheets** (Подсказки) | **Data Visualization with ggplot2** (Визуализация данных с ggplot2). В табл. 8.3 указаны основные элементы пакета `ggplot2`. Также существуют и другие элементы, дополнительную информацию о которых можно получить, воспользовавшись ресурсами, упомянутыми ранее.

Таблица 8.3.** Основные элементы пакета **ggplot2

Элемент	Описание
`data`	Исходные данные
`aes`	Эстетическое соответствие данных визуальным свойствам (оси x и y, цвет, размер и т. д.)
`geom`	Типы геометрических объектов, отображаемых на графике (линии, точки, гистограммы и т. д.)

Начнем с визуализации в виде гистограммы количества наблюдений для каждого уровня **classk**. Начнем с функции `ggplot()` и укажем три элемента из табл. 8.3:

```
ggplot(data = star,              (1)
       aes(x = classk)) +        (2)
  geom_bar()                     (3)
```

(1) Источник данных задается с помощью аргумента `data`.

(2) Эстетические соответствия данных и визуализации задаются с помощью функции `aes()`. Здесь мы указываем, например, что **classk** должен располагаться по оси x конечного графика.

(3) С помощью функции `geom_bar()` построим геометрический объект, опираясь на указанные данные и эстетические соответствия. Результат показан на рис. 8.4.

Как и в случае с оператором pipe, необязательно размещать каждый слой графика на отдельной линии, хотя это предпочтительно для удобства. Кроме того, можно построить весь график целиком, поместив курсор в любом месте внутри кода и запустив его.

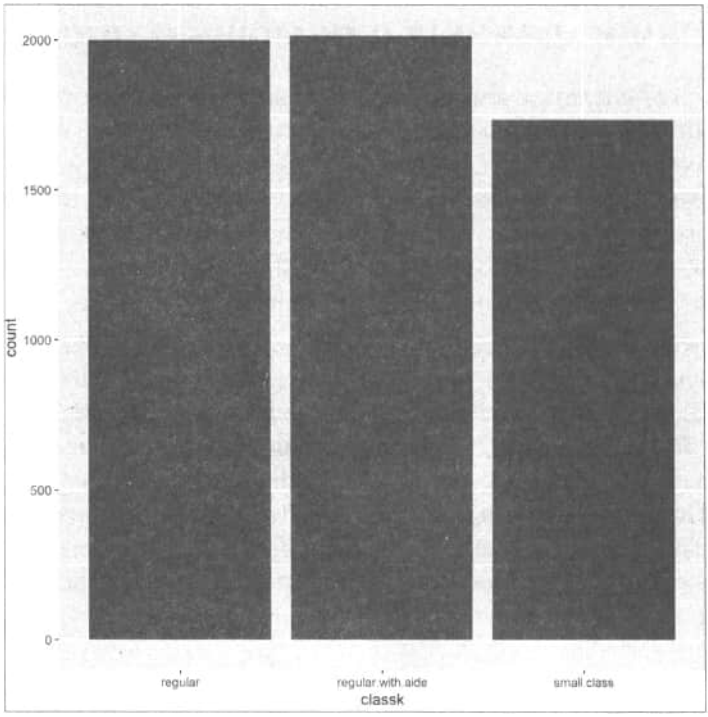


Рис. 8.4. Гистограмма в **ggplot2**

Благодаря модульному принципу с помощью пакета `ggplot2` легко выполнять итерации в визуализациях. Например, можно перейти с нашего графика к гистограмме **treadssk**, изменив сопоставление с осью x и построив результаты с помощью функции `geom_histogram()`. В результате мы получим гистограмму, как на рис. 8.5:

```
ggplot(data = star, aes(x = treadssk)) +
  geom_histogram()

#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Существует множество способов *модифицировать* диаграммы в ggplot2. Возможно, вы заметили, что в выходном сообщении для предыдущей диаграммы было указано: в гистограмме использовалось 30 столбиков. Давайте изменим это число на 25 и с помощью нескольких дополнительных аргументов в функции `geom_histogram()` используем заполнение каким-либо цветом, например розовым (pink). В результате получится гистограмма выбранного цвета с соответствующим числом столбиков (рис. 8.6).

```
ggplot(data = star, aes(x = treadssk)) +
  geom_histogram(bins = 25, fill = 'pink')
```

Для создания блочной диаграммы («боксплот») можно использовать функцию `geom_boxplot()`, результат показан на рис. 8.7:

```
ggplot(data = star, aes(x = treadssk)) +
  geom_boxplot()
```

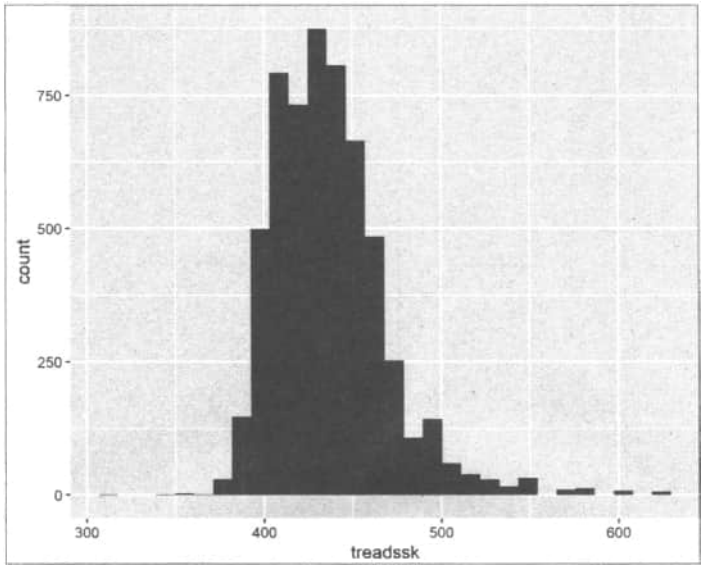


Рис. 8.5. Гистограмма в **ggplot2**

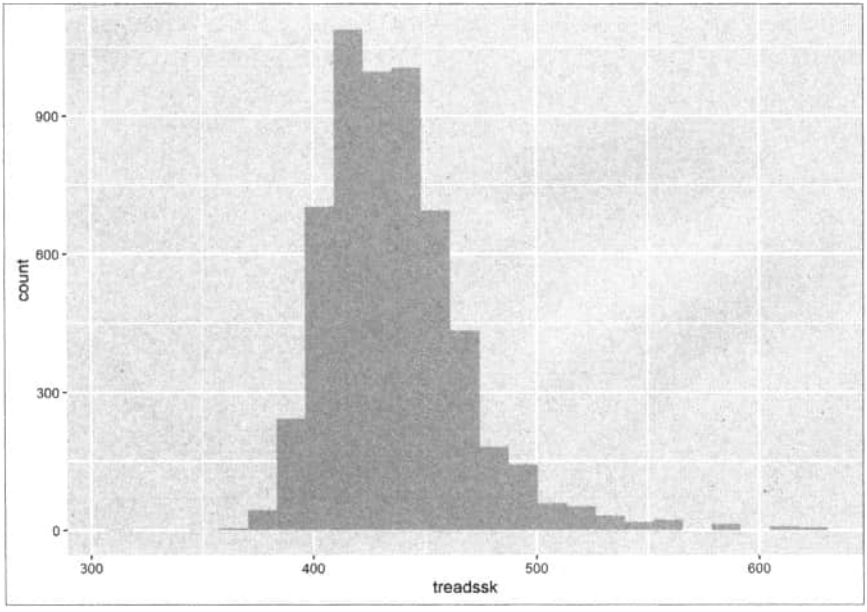


Рис. 8.6. Модифицированная гистограмма **ggplot2**

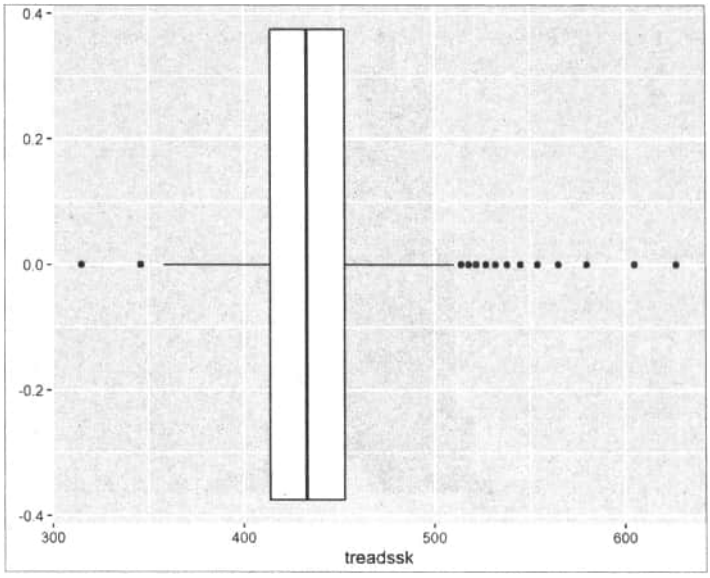


Рис. 8.7. Блочная диаграмма

В любом из случаев, рассмотренных до сих пор, мы могли «перевернуть» диаграмму, включив интересующую нас переменную в отображение на оси y вместо оси x. Давайте сделаем это с блочной диаграммой. На рис. 8.8 показан результат такого преобразования:

```
ggplot(data = star, aes(y = treadssk)) +
  geom_boxplot()
```

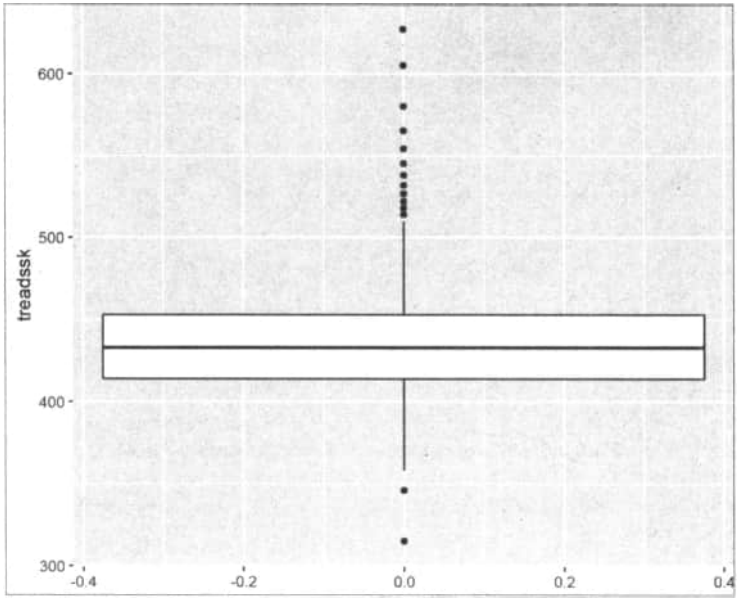


Рис. 8.8. «Перевернутая» блочная диаграмма

Теперь давайте построим блочную диаграмму для каждого уровня класса, сопоставив **classk** с осью x и **treadssk** — с осью y. В результате получим блочную диаграмму, показанную на рис. 8.9:

```
ggplot(data = star, aes(x = classk, y = treadssk)) +
  geom_boxplot()
```

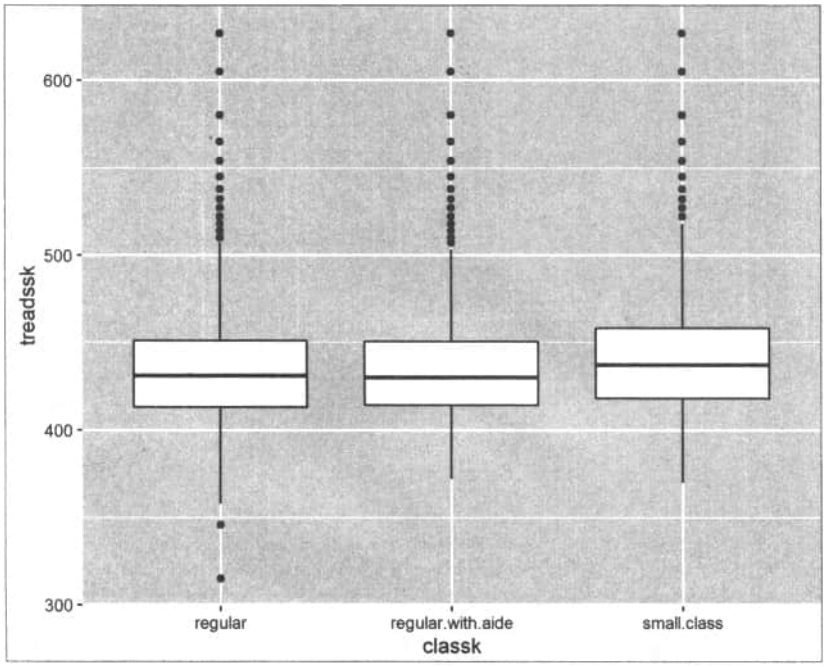


Рис. 8.9. Блочная диаграмма по группам

Аналогичным образом можно использовать функцию `geom_point()` для построения графика взаимосвязи между **tmathssk** и **treadssk** на оси x и оси y соответственно, в виде диаграммы рассеяния. Результат показан на рис. 8.10:

```
ggplot(data = star, aes(x = tmathssk, y = treadssk)) +
    geom_point()
```

Можно использовать некоторые другие функции `ggplot2` для нанесения меток на оси x и y, а также для заголовка диаграммы. Результат показан на рис. 8.11:

```
ggplot(data = star, aes(x = tmathssk, y = treadssk)) +
  geom_point() +
  xlab('Math score') + ylab('Reading score') +
  ggtitle('Math score versus reading score')[3]
```

[3] Средний балл по математике и чтению.— *Ред.*

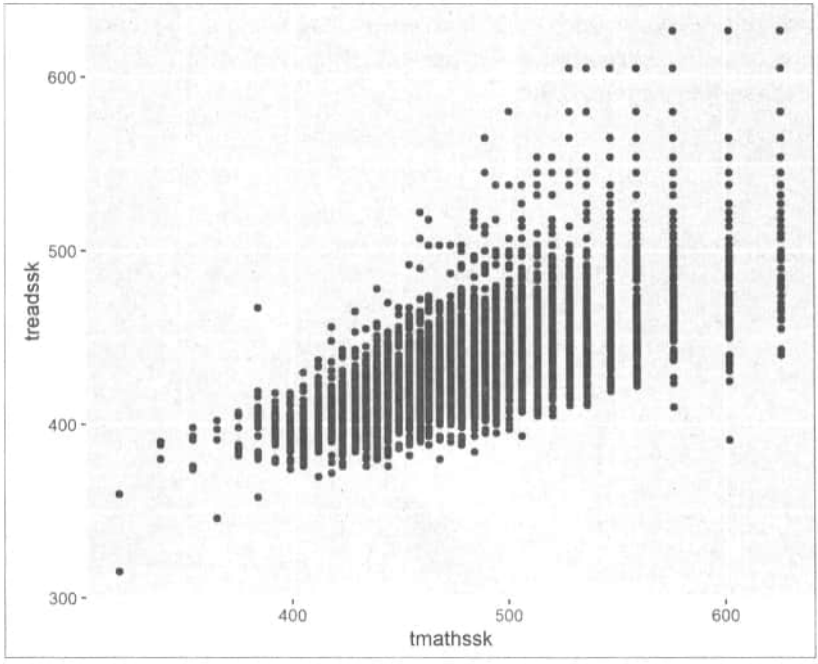


Рис. 8.10. Диаграмма рассеяния

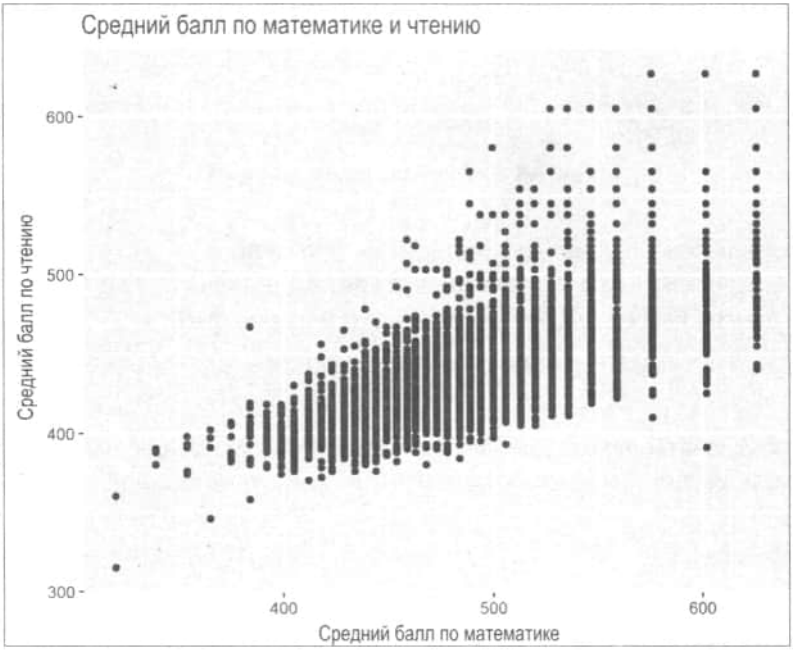


Рис. 8.11. Диаграмма рассеяния с модифицированными метками осей и заголовком

Заключение

Пакеты `dplyr` и `ggplot2` могут делать значительно больше, чем вы увидели здесь, но этого достаточно, чтобы вы могли приступить к выполнению глобальной задачи — изучению и тестированию взаимосвязей данных, чему посвящена *глава 9*.

Упражнения

В репозитории на GitHub имеются два файла, расположенные в подпапке **census** (численность населения) папки **datasets**: **census.csv** и **census-divisions.csv**. Считайте их в R и выполните следующие задания.

1. Отсортируйте данные по регионам (по возрастанию), по подразделениям (по возрастанию) и по численности населения (по убыванию) (для выполнения вам потребуется объединить наборы данных). Запишите результаты в рабочий лист Excel.
2. Исключите поле почтового кода из объединенного набора данных.
3. Создайте новый столбец **density** (плотность населения), содержащий расчет численности населения, деленной на площадь суши.
4. Визуализируйте зависимость между площадью суши и численностью населения для всех наблюдений в 2015 году.
5. Найдите общую численность населения для каждого региона в 2015 году.
6. Создайте таблицу, содержащую названия штатов и данные о численности населения за каждый год в промежутке с 2010 по 2015, расположенные в отдельном столбце.

ГЛАВА 9

Кульминация: R для анализа данных

В этой главе мы применим полученные знания об анализе и визуализации данных в R для изучения и тестирования зависимостей в уже знакомом нам наборе данных **mpg**. Здесь вы познакомитесь с несколькими новыми методами в R, включая t-тест и линейную регрессию. Начнем с вызова необходимых пакетов, считывания файла **mpg.csv** из подпапки **mpg** папки **datasets** репозитория к книге и выбора интересующих нас столбцов. Мы еще не использовали библиотеку `tidymodels`, поэтому вам может понадобиться ее установить.

```
library(tidyverse)
library(psych)
library(tidymodels)

# Считать данные, выбрать только необходимые столбцы
mpg <- read_csv('datasets/mpg/mpg.csv') %>%
  select(mpg, weight, horsepower, origin, cylinders)

#> -- Column specification--------------------------------------------------------
#> cols(
#>  mpg = col_double(),
#>   cylinders = col_double(),
#>   displacement = col_double(),
#>   horsepower = col_double(),
#>   weight = col_double(),
#>   acceleration = col_double(),
#>   model.year = col_double(),
#>   origin = col_character(),
#>   car.name = col_character() #> )
#>   )

head(mpg)
#> # Тиббл: 6 x 5
#>     mpg weight horsepower origin cylinders
#>   <dbl>  <dbl>      <dbl> <chr>       <dbl>
#> 1   18   3504        130  USA             8
#> 2   15   3693        165  USA             8
#> 3   18   3436        150  USA             8
#> 4   16   3433        150  USA             8
#> 5   17   3449        140  USA             8
#> 6   15   4341        198  USA             8
```

Разведочный анализ данных

Исследование данных следует начать с описательной статистики. Мы используем функцию `describe()` из пакета `psych`:

```
describe(mpg)
#>            vars   n    mean     sd  median trimmed    mad  min
#> mpg           1 392   23.45   7.81   22.75   22.99   8.60    9
#> weight        2 392 2977.58 849.40 2803.50 2916.94 948.12 1613
#> horsepower    3 392  104.47  38.49   93.50   99.82  28.91   46
#> origin*       4 392    2.42   0.81    3.00    2.53   0.00    1
#> cylinders     5 392    5.47   1.71    4.00    5.35   0.00    3
#>                max  range  skew kurtosis    se
#> mpg           46.6   37.6  0.45    -0.54  0.39
#> weight      5140.0 3527.0  0.52    -0.83 42.90
#> horsepower   230.0  184.0  1.08     0.65  1.94
#> origin*        3.0    2.0 -0.91    -0.86  0.04
#> cylinders      8.0    5.0  0.50    -1.40  0.09
```

Поскольку переменная **origin** (регион происхождения) является категориальной, при интерпретации ее описательной статистики следует соблюдать осторожность (в пакете `psych` это предупреждение обозначается звездочкой — `*`). Однако мы можем проанализировать ее одномерную таблицу частот с помощью новой функции `count()` пакета `dplyr`:

```
mpg %>%
  count(origin)
#> # Тиббл: 3 x 2
#>   origin     n
#>   <chr>  <int>
#> 1 Asia      79
#> 2 Europe    68
#> 3 USA      245
```

Из результирующего столбца **n** видно, что, хотя большинство наблюдений относится к американским автомобилям, наблюдения азиатских и европейских автомобилей также являются репрезентативными выборками из подмножеств.

Далее разобьем эти подсчеты по переменной **cylinders**, чтобы получить двумерную таблицу частот. Чтобы распределить значение `cylinders` по столбцам, используем функцию `count()` в комбинации с функцией `pivot_wider()`:

```
mpg %>%
  count(origin, cylinders) %>%
  pivot_wider(values_from = n, names_from = cylinders)
#> # A tibble: 3 x 6
#>   origin     `3`   `4`   `6`   `5`     `8`
#>   <chr>  <int> <int> <int> <int> <int>
#> 1 Asia       4    69     6    NA     NA
#> 2 Europe    NA    61     4     3     NA
#> 3 USA       NA    69    73    NA    103
```

Помните: NA в R указывает на отсутствие значения, в данном случае — т. к. не было найдено никаких наблюдений для некоторых из этих срезов данных.

Немногие из автомобилей имеют трех- или пятицилиндровые двигатели, и *только* американские автомобили имеют восемь цилиндров. При анализе данных часто встречаются несбалансированные наборы данных, где на некоторых уровнях имеется непропорционально большое количество наблюдений. Для моделирования таких данных часто требуются специальные методы. Чтобы узнать больше о работе с несбалансированными данными, ознакомьтесь с книгой «Практическая статистика для специалистов Data Science» Питера Брюса и соавт., 2-е изд.[1] (Peter Bruce et al. Practical Statistics for Data Scientists, 2nd edition).

Мы также можем вывести описательную статистику для каждого уровня переменной **origin**. Сначала используем функцию `select()`, чтобы выбрать интересующую нас переменную, затем — функцию `describeBy()` пакета `psych`, присваивающую `groupBy` переменной **origin**:

```
mpg %>%
  select(mpg, origin) %>%
  describeBy(group = 'origin')

#>  Descriptive statistics by group
#> origin: Asia
         vars  n  mean   sd median trimmed  mad min  max range
#> mpg        1 79 30.45 6.09  31.6   30.47 6.52  18 46.6  28.6
#> origin*    2  79  1.00 0.00   1.0    1.00 0.00   1  1.0   0.0
         skew kurtosis   se
#> mpg      0.01    -0.39  0.69
#> origin*  NaN      NaN  0.00

#> origin: Europe
        vars  n mean   sd median trimmed  mad  min  max range
#> mpg        1 68 27.6 6.58     26   27.1  5.78 16.2 44.3  28.1
#> origin*    2 68  1.0 0.00      1    1.0  0.00  1.0  1.0   0.0
        skew kurtosis  se
#> mpg     0.73    0.31 0.8
#> origin*  NaN     NaN 0.0

#> origin: USA
        vars   n  mean   sd median trimmed  mad min max range
#> mpg        1 245 20.03 6.44  18.5    19.37 6.67  9   39    30
#> origin*    2 245  1.00 0.00   1.0     1.00 0.00  1    1     0
        skew kurtosis   se
#> mpg     0.83    0.03 0.41
#> origin*  NaN     NaN 0.00
```

[1] Издана в России: БХВ-Петербург, 2021: **https://bhv.ru/product/prakticheskaya-statistika-dlya-spetsialistov-data-science-2-e-izd.** — *Ред.*

Давайте подробнее рассмотрим потенциальную взаимосвязь между переменными **origin** (регион происхождения) и **mpg** (пробег). Начнем с визуализации распределения величины **mpg** с помощью гистограммы (рис. 9.1).

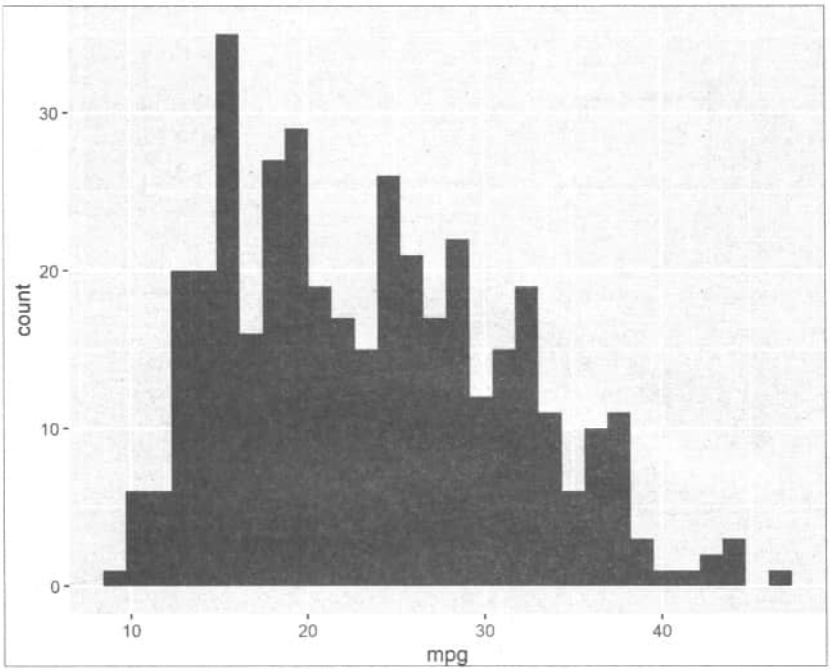


Рис. 9.1. Распределение величины **mpg**

Теперь можем заняться распределением **mpg** по **origin**. Наложение всех трех уровней величины **origin** на одну гистограмму может привести к беспорядку, поэтому более подходящей может быть блочная диаграмма («боксплот»), подобная той, что изображена на рис. 9.2.

```
ggplot(data = mpg, aes(x = origin, y = mpg)) +
  geom_boxplot()
```

Визуализировать эти переменные в виде гистограмм, не создавая беспорядок, в R можно с помощью *фасетной* гистограммы. Чтобы разбить гистограмму `ggplot2` на подгистограммы, или *фасеты*, воспользуемся функцией `facet_wrap()`. Начнем с оператора `~`, или тильда, за которым следует имя переменной. Знак «тильда» в R следует воспринимать как «по». Например, мы разбиваем гистограмму на фасеты по `origin`, в результате чего получаем гистограммы, показанные на рис. 9.3:

```
# Гистограмма mpg, распределенная по origin
ggplot(data = mpg, aes(x = mpg)) +
  geom_histogram() +
  facet_grid(~ origin)
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

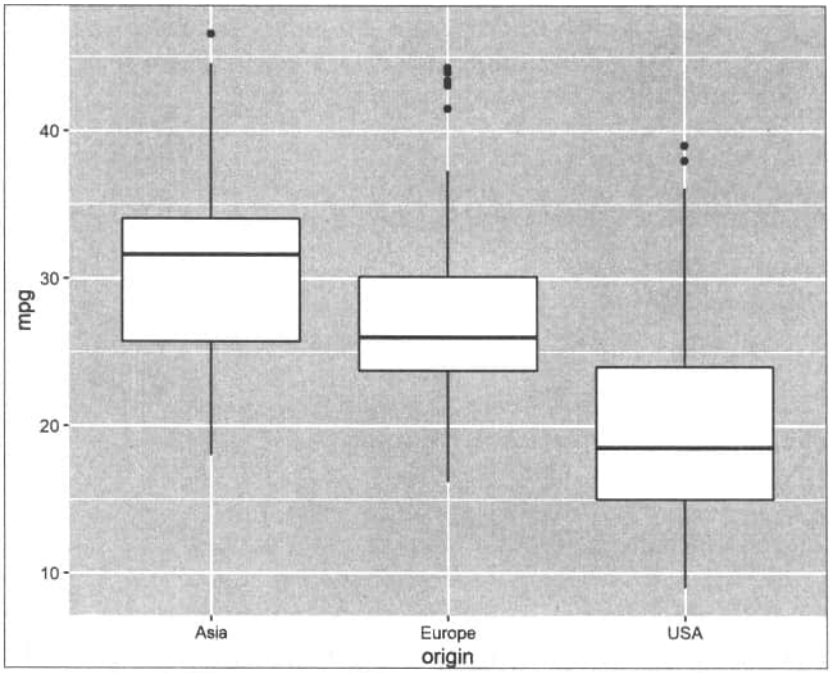


Рис. 9.2. Распределение **mpg** по **origin**

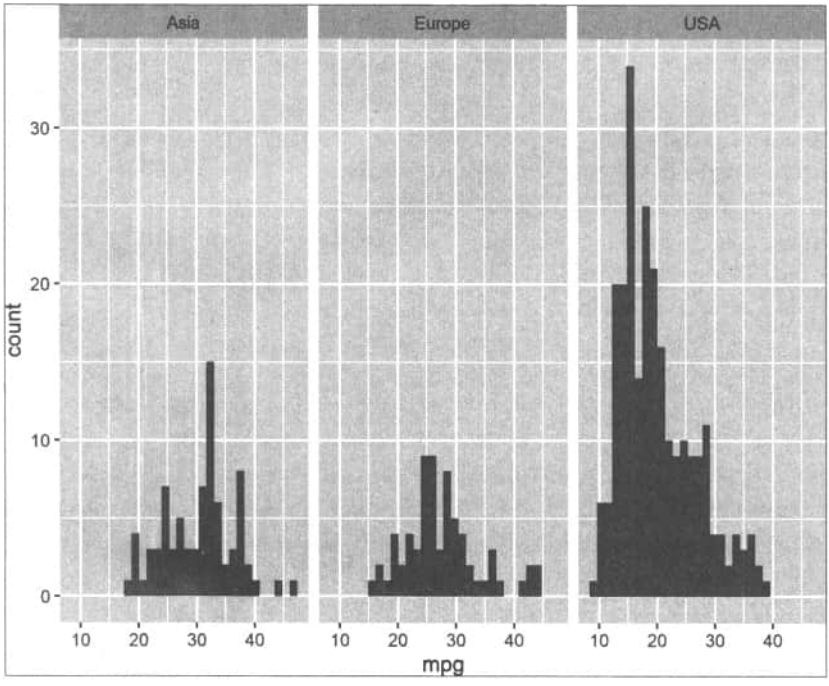


Рис. 9.3. Распределение **mpg** по **origin**

Проверка гипотез

Можно продолжить изучение данных с помощью этих методов, но мы перейдем к проверке гипотез. Итак, я хотел бы знать, имеется ли существенная разница в пробеге между американскими и европейскими автомобилями. Давайте создадим новый кадр данных, содержащий только эти наблюдения; мы будем его использовать для выполнения t-теста.

```
mpg_filtered <- filter(mpg, origin=='USA' | origin=='Europe')
```

Проверка зависимостей между несколькими группами

Мы могли бы осуществить проверку гипотез относительно разницы в пробеге между американскими, европейскими и азиатскими автомобилями; это другой статистический тест, называемый *дисперсионным анализом*, или ANOVA. Вы сможете изучить его в вашем следующем аналитическом путешествии.

t-тест для независимых выборок

R включает стандартную функцию `t.test()`: необходимо указать, откуда поступают данные, с помощью аргумента *data*, а также указать, какую *формулу* тестировать. Для этого установим зависимость между независимой и зависимой переменными с помощью оператора `~`. Независимая переменная стоит перед символом ~, а зависимая — после. Напомню: эту запись следует интерпретировать как анализ изменения **mpg** по **origin**.

```
# Зависимая переменная ~ («по») независимой переменной
t.test(mpg ~ origin, data = mpg_filtered)
#>  Welch Two Sample t-test
#>
#>      data: mpg by origin
#>      t = 8.4311, df = 105.32, p-value = 1.93e-13
#>      alternative hypothesis: true difference in means is not equal to 0[2]
#>      95 percent confidence interval[3]:
#>      5.789361 9.349583
#>      sample estimates[4]:
#>      mean in group Europe   mean in group USA[5]
#>                  27.60294             20.03347
```

Разве не замечательно, что R не только явно указывает, какова наша альтернативная гипотеза, но *и* включает доверительный интервал наряду с p-значением? (Вы можете сказать, что эта программа создана для статистического анализа.) Основы-

---

[2] Альтернативная гипотеза: истинная разница в средних значениях не равна 0. — *Ред.*

[3] Доверительный интервал — 95 %. — *Ред.*

[4] Выборочные оценки. — *Ред.*

[5] Значение в группе «Европа»; значение в группе «США». — *Ред.*

ваясь на p-значении, мы отвергнем нулевую гипотезу; очевидно, имеется доказательство разницы между средними.

Теперь давайте обратим внимание на зависимости между непрерывными переменными. Сначала используем функцию `cor()` из базового R для печати матрицы корреляции. Сделаем это только для непрерывных переменных в наборе **mpg**:

```
select(mpg, mpg:horsepower) %>%
cor()
#>                 mpg      weight   horsepower
#> mpg         1.0000000 -0.8322442 -0.7784268
#> weight     -0.8322442  1.0000000  0.8645377
#> horsepower -0.7784268  0.8645377  1.0000000
```

Мы можем использовать `ggplot2` для визуализации, например, зависимости между весом (**weight**) и пробегом (**mpg**), как показано на рис. 9.4:

```
ggplot(data = mpg, aes(x = weight,y = mpg)) +
geom_point() + xlab('weight (pounds)') +
ylab('mileage (mpg)') + ggtitle('Relationship between weight and mileage')[6]
```

В качестве альтернативы мы могли бы использовать функцию `pairs()` из базового R, чтобы создать парный график всех комбинаций переменных, построенный

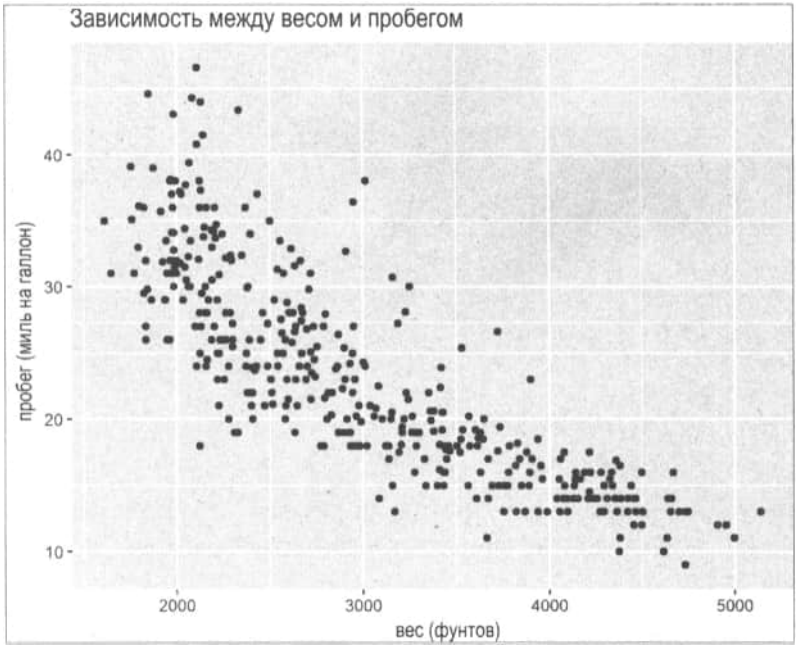


Рис. 9.4. Диаграмма рассеяния

[6] Зависимость между весом и пробегом — *Ред.*

аналогично матрице корреляции. На рис. 9.5 показан парный график выбранных переменных из набора **mpg**:

```
select(mpg, mpg:horsepower) %>%
  pairs()
```

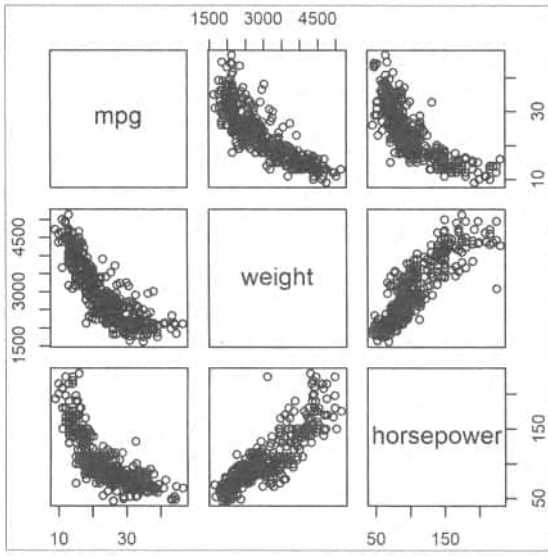


Рис. 9.5. Парный график

Линейная регрессия

Теперь мы готовы к созданию модели линейной регрессии с помощью функции `lm()` (сокращение от «линейная модель») из базового R. Так же как в функции `t.test()`, укажем набор данных и формулу. Линейная регрессия возвращает значительно больше результатов, чем t-тест, поэтому обычно сначала присваивают результаты новому объекту в R, а затем по отдельности исследуют его различные элементы. Функция `summary()`, в частности, дает полезный обзор регрессионной модели:

```
mpg_regression <- lm(mpg ~ weight, data = mpg)
summary(mpg_regression)

#>     Call:
#>     lm(formula = mpg ~ weight, data = mpg)
#>
#>     Residuals:
#>         Min      1Q  Median      3Q     Max
#>     -11.9736 -2.7556  -0.3358  2.1379  16.5194
#>
#>     Coefficients:
#>                 Estimate Std. Error t value Pr(>|t|)
```

```
#>     (Intercept) 46.216524   0.798673   57.87   <2e-16 ***
#>     weight      -0.007647   0.000258  -29.64   <2e-16 ***
#>     ---
#>     Signif. codes:   0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>     Residual standard error: 4.333 on 390 degrees of freedom
#>     Multiple R-squared: 0.6926,      Adjusted R-squared:  0.6918
#>     F-statistic: 878.8 on 1 and 390 DF,  p-value: < 2.2e-16
```

Вам должны быть знакомы такие выходные данные. Здесь наряду с другими числами вы найдете коэффициенты, p-значения и R-квадрат (коэффициент детерминации). Кроме того, действительно наблюдается значительное влияние веса на пробег.

Наконец, что не менее важно, можно поместить эту линию регрессии поверх диаграммы рассеяния, включив функцию `geom_smooth()` в функцию `ggplot()` и присвоив `method` значение `lm`. Результат показан на рис. 9.6:

```
ggplot(data = mpg, aes(x = weight, y = mpg)) +
geom_point() + xlab('weight (pounds)') +
ylab('mileage (mpg)') + ggtitle('Relationship between weight and mileage') +
geom_smooth(method = lm)
#> `geom_smooth()` using formula 'y ~ x'
```

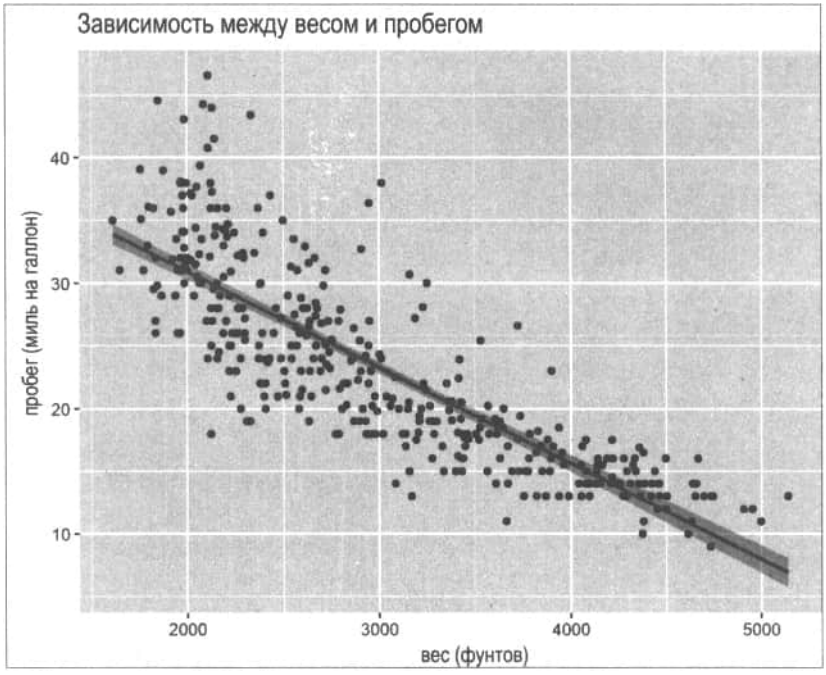


Рис. 9.6. Диаграмма рассеяния с линией регрессии

Доверительный интервал и линейная регрессия

Взгляните на затененную область вдоль линии на рис. 9.6. Это доверительный интервал наклона линии регрессии, указывающий с 95 %-ной достоверностью, где, по нашему мнению, должна быть истинная оценка генеральной совокупности для каждого значения x.

Разделение и проверка данных для обучения и тестирования

В *главе 5* мы кратко рассмотрели, как машинное обучение взаимосвязано с работой с данными в широком смысле. Метод, популяризированный машинным обучением, с которым вы можете столкнуться в аналитической работе, — это *разделение на обучение и тестирование* (train/test split). Идея заключается в том, чтобы обучить модель на подмножестве данных, а затем протестировать ее на другом подмножестве. Это гарантирует, что модель работает не только на одной конкретной выборке наблюдений, но и может быть распространена на более широкую совокупность данных. Специалистов по обработке данных часто особенно интересует, насколько хорошо модель делает прогнозы на основе тестирования данных.

Давайте разобьем наш набор данных **mpg** в R, обучим модель линейной регрессии на одной части данных и затем протестируем ее на остальной части. Мы будем использовать пакет `tidymodels`, который, хотя и не является частью библиотеки `tidyverse`, но построен на тех же принципах и поэтому хорошо с ней работает.

Возможно, вы помните из *главы 2*, что в связи с использованием случайных чисел результаты, полученные в вашей рабочей книге, отличались от тех, что были в ней задокументированы. Поскольку здесь мы снова будем разбивать набор данных случайным образом, то можем столкнуться с той же проблемой. Чтобы этого избежать, можно установить начальное значение *генератора* случайных чисел R — это приведет к генерации одной и той же серии случайных чисел каждый раз. Используем с этой целью функцию `set.seed()`. Вы можете задать любое число, вполне обычным является `1234`:

```
set.seed(1234)
```

Чтобы начать разделение, можно использовать функцию с соответствующим названием `initial_split()`[7]; далее мы разделим данные на части для обучения и для тестирования с помощью функций `training()` и `testing()` соответственно:

```
dim(mpg_train)
#> [1] 294 5
dim(mpg_test)
#> [1] 98 5
```

При 294 и 98 наблюдениях размеры наших выборок для обучения и тестирования должны быть достаточно велики для рефлективного статистического вывода. Хотя это не часто принимается во внимание для больших наборов данных, используемых

[7] *Initial* — начальный, исходный; *split* — разделять. — *Ред.*

в машинном обучении, при разделении данных адекватный размер выборки может служить ограничением.

Можно разделить данные в пропорциях, отличных от 75/25, использовать специальные методы разделения данных и т. д. Для получения более подробной информации ознакомьтесь с документацией по `tidymodels`; до тех пор, пока вы не освоитесь с регрессионным анализом, значения по умолчанию вполне подойдут.

Чтобы построить модель обучения, мы сначала *укажем*, к какому типу она относится, с помощью функции `linear_reg()` , а затем *совместим* ее с данными для обучения. Входные данные функции `fit()` должны быть вам знакомы, только на этот раз мы используем только обучающее подмножество набора **mpg**.

```
# Указать тип модели
lm_spec <- linear_reg()

# Подобрать модель под данные
lm_fit <- lm_spec %>%
fit(mpg ~ weight, data = mpg_train)
#> Warning message:
#> Engine set to `lm`.
```

Из вывода консоли видно, что функция `lm()` базового R, с которой вы уже работали ранее, использовалась в качестве *механизма* для подбора модели.

С помощью функции `tidy()` мы можем получить коэффициенты и p-значения нашей модели для обучения, а с помощью функции `glance()` — ее показатели производительности (например, R-квадрат).

```
tidy(lm_fit)
#> # Тиббл: 2 x 5
#>   term        estimate std.error statistic   p.value
#>   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
#> 1 (Intercept) 47.3        0.894     52.9  1.37e-151
#> 2 weight      -0.00795 0.000290    -27.5  6.84e- 83 #>
glance(lm_fit)
#> # Тиббл: 1 x 12
#>    r.squared adj.r.squared sigma statistic   p.value     df logLik   AIC
#>        <dbl>         <dbl> <dbl>     <dbl>     <dbl>  <dbl> <dbl>   <dbl>
#> 1      0.721         0.720   4.23      754.   6.84e-83      1 -840.   1687.
#> # ... еще 4 переменные: BIC <dbl>, deviance <dbl>,
#> #  df.residual <int>, nobs <int>
```

Все это замечательно, но *на самом деле* нас интересует, насколько хорошо эта модель будет работать, когда мы применим ее к новому набору данных; вот тут-то и приходит время тестовой части данных. Чтобы делать прогнозы на подмножестве `mpg_test`, будем использовать функцию `predict()` . Также используем функцию `bind_cols()`, чтобы добавить столбец прогнозируемых значений Y в кадр данных. По умолчанию этот столбец будет иметь имя **.pred**.

```
mpg_results <- predict(lm_fit, new_data = mpg_test) %>%
  bind_cols(mpg_test)

mpg_results
#> # Тиббл: 98 x 6
#>    .pred  mpg weight horsepower origin cylinders
#>    <dbl> <dbl>  <dbl>      <dbl> <chr>      <dbl>
#>  1  20.0   16  3433         150   USA          8
#>  2  16.7   15  3850         190   USA          8
#>  3  25.2   18  2774          97   USA          6
#>  4  30.3   27  2130          88   Asia         4
#>  5  28.0   24  2430          90   Europe       4
#>  6  21.0   19  3302          88   USA          6
#>  7  14.2   14  4154         153   USA          8
#>  8  14.7   14  4096         150   USA          8
#>  9  29.6   23  2220          86   USA          4
#> 10  29.2   24  2278          95   Asia         4
#> # ... и еще 88 строк
```

Теперь, когда мы применили модель к новым данным, давайте оценим ее производительность. Например, с помощью функции `rsq()` мы можем найти ее R-квадрат. Для кадра данных `mpg_results` нам надо указать с помощью аргумента `truth`, какой столбец содержит фактические значения Y, и с помощью столбца `estimate` — прогнозируемые значения.

```
rsq(data = mpg_results, truth = mpg, estimate = .pred)
#> # Тиббл: 1 x 3
#>   .metric .estimator  .estimate
#>   <chr>   <chr>          <dbl>
#> 1 rsq     standard       0.606
```

С коэффициентом детерминации (R-квадрат), равным 60,6 %, модель, полученная на основе обучающего набора данных, объясняет значительную изменчивость в тестовых данных.

Другой распространенный критерий оценки — *среднеквадратичная ошибка* (root mean square error, RMSE). В *главе 4* вы познакомились с такой концепцией, как *остатки* (разница между фактическим и прогнозируемым значениями); среднеквадратичная ошибка является стандартным отклонением остатков и, следовательно, оценкой разброса ошибок. Функция `rmse()` возвращает среднеквадратичное значение.

```
rmse(data = mpg_results, truth = mpg, estimate = .pred)
#> # Тиббл: 1 x 3
#>   .metric .estimator .estimate
#>   <chr>   <chr>          <dbl>
#> 1 rmse    standard        4.65
```

Поскольку это соотносится с масштабом зависимой переменной, не существует универсального способа оценки среднеквадратичной ошибки, но из двух конкури-

рующих моделей, использующих одни и те же данные, предпочтительнее та, у которой меньшая среднеквадратичная ошибка (RMSE).

Суперпакет `tidymodels` предоставляет множество методов для подбора и оценки моделей в R. Мы рассмотрели регрессионную модель, которая принимает непрерывную зависимую переменную, но, кроме этого, можно построить *модели классификации*, в которых зависимая переменная является категориальной. `tidymodels` — относительно новый пакет в R, поэтому ему посвящено не так много специальной литературы, но по мере его развития количество учебных материалов, несомненно, будет расти.

Заключение

Разумеется, вы могли бы более глубоко изучить и проверить зависимости в этом и других наборах данных, но предпринятые здесь шаги являются хорошим началом. Ранее вы умели проводить и интерпретировать такие действия в Excel, а теперь перешли к выполнению их в R.

Упражнения

Найдите время попробовать свои силы в анализе известных вам наборов данных посредством знакомых шагов, но теперь с использованием R. В конце *главы 4* вы практиковались в анализе данных из набора **ais**. Эти данные доступны в пакете R `DAAG`; попробуйте выполнить его установку и загрузку оттуда (он доступен как объект `ais`). Затем выполните следующие задания.

1. Визуализируйте распределение количества эритроцитов (**rcc**) в зависимости от пола (**sex**).
2. Ответьте на вопрос: существует ли значимое различие в количестве эритроцитов между двумя группами разного пола?
3. Создайте матрицу корреляции значимых переменных в этом наборе данных.
4. Визуализируйте зависимость между ростом (**ht**) и весом (**wt**).
5. Постройте регрессию **ht** по **wt**. Найдите уравнение подходящей линии регрессии. Имеется ли какая-то существенная зависимость? Какой процент дисперсии в **ht** объясняется **wt**?
6. Разделите регрессионную модель на подмножества для обучения и тестирования. Какова величина R-квадрата и среднеквадратичной ошибки (RMSE) вашей модели обучения?

ЧАСТЬ III

От Excel к Python

ГЛАВА 10

Первые шаги в Python для пользователей Excel

Язык программирования Python, созданный в 1991 году Гвидо ван Россумом, является бесплатным и имеет открытый исходный код, как и R. В то время ван Россум читал сценарии «Летающего цирка Монти Пайтона» (Monty Python's Flying Circus) и решил назвать язык в честь этой британской комедии. В отличие от R, предназначенного специально для анализа данных, Python был разработан как универсальный язык для выполнения таких задач, как взаимодействие с операционными системами, управление ошибками и т. д. Это привело к ряду важных последствий, отразившихся в особенностях «мышления» и работы Python с данными. Например, в *главе 7* вы видели, что R имеет встроенную табличную структуру данных. В Python такого нет; здесь для работы с данными следует больше полагаться на внешние пакеты.

Это отнюдь не является проблемой: Python, как и R, имеет тысячи пакетов, поддерживаемых обширным сообществом людей. Вы обнаружите, что Python используется для всего — от разработки мобильных приложений до встроенных устройств, и, конечно, для анализа данных. Армия его пользователей стремительно растет, и Python становится одним из наиболее популярных языков программирования не только для аналитики, но и для информационных технологий в целом.

Python был задуман как универсальный язык программирования, тогда как R разрабатывался специально для статистического анализа.

Загрузка Python

Фонд программного обеспечения Python (Python Software Foundation) поддерживает «официальный» исходный код Python. Поскольку исходный код является открытым, любой желающий может брать, добавлять и распространять код Python. Один из дистрибутивов Python — Anaconda, установка которого рекомендуется для работы с этой книгой. Он поддерживается одноименной коммерческой компанией и доступен в виде платных версий; мы будем использовать бесплатную отдельную версию (Individual Edition). Сейчас предлагается третья версия продукта, Python 3. Последнюю версию Python 3 можно загрузить с сайта Anaconda.

Python 2 и Python 3

Python 3, выпущенный в 2008 году, внес существенные изменения в язык и, что важно, не был обратно совместим с кодом Python 2. Это означает, что код, написанный на Python 2, мог не работать на Python 3 и наоборот. К моменту написания этой книги использование Python 2 было официально прекращено, хотя при изучении Python вы можете столкнуться с некоторыми ссылками и остатками кода, написанными на этой версии.

Помимо упрощенной установки языка Python, Anaconda поставляется с дополнениями, включая популярные пакеты, которые мы будем позже использовать в этой книге. Также этот дистрибутив содержит веб-приложение Jupyter Notebook, которое мы будем использовать при работе с Python.

Начало работы с Jupyter

Как уже упоминалось в *главе 6*, язык R был смоделирован по образцу языка S, предназначенного для разведочного анализа данных (Exploratory Data Analysis, EDA). Из-за итеративного характера EDA ожидаемый рабочий процесс языка — выполнение выбранных строк кода и исследование полученных выходных данных. Это дает возможность легко осуществлять анализ данных непосредственно из R-скрипта, файла с расширением **.r**. Мы использовали интегрированную среду разработки RStudio (RStudio IDE) для обеспечения дополнительной поддержки сессии программирования, например выделенных панелей для справочной документации и информации об объектах в нашей рабочей среде.

Язык Python, напротив, в некотором смысле ведет себя более как «низкоуровневые» языки программирования, в которых код сначала должен быть скомпилирован в машиночитаемый файл, а затем запущен на выполнение. Это может ощутимо усложнить выполнение поэтапного анализа данных из скрипта Python, файла с расширением **.py**. Этот болезненный момент использования Python для статистических и, в более широком смысле, научных вычислений привлек внимание физика и разработчика программного обеспечения Фернандо Переса (Fernando Pérez), запустившего в 2001 году вместе с коллегами проект IPython, предназначенный для создания более интерактивного интерпретатора Python (IPython — шутливое сокращение от «interactive Python»). Одним из результатов этой инициативы был новый тип файла, IPython Notebook, обозначаемого расширением **.ipynb**.

Этот проект быстро развивался, и в 2014 году IPython был включен в более широкий проект, Jupyter, независимую от языка программирования инициативу по разработке интерактивного программного обеспечения с открытым исходным кодом. Таким образом, IPython Notebook превратился в Jupyter Notebook, сохранив при этом расширение файла **.ipynb**. Блокноты Jupyter работают как интерактивные веб-приложения, дающие пользователям возможность комбинировать код с текстом, уравнениями и т. д. для создания мультимедийных интерактивных документов. Отчасти Jupyter получил свое название в честь записных книжек (блокнотов) Галилея, которые тот использовал для записей о своем открытии спутников планеты Юпитер. *Ядро* (kernel) используется в фоновом режиме для выполнения кода блокнота. Загрузив дистрибутив Anaconda, вы установили все компоненты, необходимые для

выполнения кода Python из блокнота Jupyter: теперь вам достаточно просто запустить сессию.

RStudio, Jupyter Notebooks и другие способы программирования

Возможно, вам не захочется расставаться с RStudio, чтобы познакомиться с еще одним интерфейсом. Но не забывайте, что код и приложение часто не связаны в системах с открытым исходным кодом; эти языки и платформы можно легко «смешивать». Например, R — один из множества языков с ядром для Jupyter. Название «Jupyter», кроме отсылки к Галилею, также является комбинацией названий трех основных поддерживаемых им языков: Julia, Python и R.

Вместе с тем можно выполнять скрипты Python из RStudio с помощью пакета R `reticulate`, который можно использовать более широко, чтобы запустить код Python из R. Это означает, что можно, к примеру, импортировать и обрабатывать данные в Python и затем использовать R для визуализации. К другим популярным программам для работы с кодом Python относятся PyCharm и Visual Studio Code. В RStudio также имеется свое приложение-блокнот — R Notebooks. Применяется та же концепция чередования кода и текста, что и в Jupyter, при поддержке нескольких языков, включая R и Python.

Как вы начинаете осознавать, существует великое множество инструментов, предназначенных для программирования в R и Python, значительно больше, чем можно охватить в этой книге. Наше внимание было сосредоточено на скриптах R из RStudio и Python из Jupyter Notebooks, т. к. они относительно понятны для начинающих и более распространены, чем другие конфигурации. Как только вы почувствуете себя уверенно в этих процессах, найдите в Интернете упомянутые здесь среды разработки (IDE). Продолжая учиться, вы узнаете еще больше способов взаимодействия с этими языками.

Шаги, необходимые для запуска блокнота Jupyter, различны для ОС Windows и Mac. На компьютере с Windows откройте меню **Пуск** (Start), затем найдите и запустите **Anaconda Prompt**. Это инструмент командной строки для работы с дистрибутивом Anaconda и еще один способ взаимодействия с кодом Python. Для получения более подробной информации о запуске Python из командной строки с учетом опыта работы в Excel ознакомьтесь с книгой Феликса Цумштейна «Python для Excel» (Felix Zumstein. Python for Excel). Из Anaconda prompt введите команду `jupyter notebook` в позиции курсора и нажмите <Enter>. Команда будет выглядеть подобным образом, но с другим путем к домашнему каталогу:

```
(base) C:\Users\User> jupyter notebook
```

На Mac откройте **Launchpad**, затем найдите и запустите **Terminal**. Это интерфейс командной строки, который поставляется с Mac и может использоваться для взаимодействия с Python. Из Terminal введите `jupyter notebook` в позиции курсора и нажмите <Enter>. Команда будет выглядеть подобным образом, но с другим путем к домашнему каталогу:

```
user@MacBook-Pro ~ % jupyter notebook
```

После выполнения этой команды в любой из систем произойдет следующее: во-первых, откроется дополнительное окно типа терминала. *Не закрывайте это окно.* Именно оно устанавливает соединение с ядром. Кроме того, интерфейс блокнота Jupyter автоматически откроется в веб-браузере по умолчанию. Если этого не произойдет, в терминальном окне будет отображаться ссылка, которую вы можете вставить в браузер. На рис. 10.1 показано, что́ вы должны увидеть в браузере.

Рис. 10.1. Целевая страница (лендинг) Jupyter

Jupyter запустится с интерфейсом, подобным проводнику файлов. Теперь вы можете найти папку, в которой будете сохранять ваши блокноты.

Чтобы открыть новый блокнот, перейдите в верхнюю правую часть окна браузера и выберите последовательно: **New** (Новый) | **Notebook** (Блокнот) | **Python 3**. Откроется новая вкладка с пустым блокнотом Jupyter. Так же, как и RStudio, Jupyter предоставляет значительно больше функций, чем мы можем рассмотреть здесь; для начала сосредоточимся на важнейших элементах. На рис. 10.2 показаны четыре основных компонента блокнота Jupyter. Рассмотрим каждый из них.

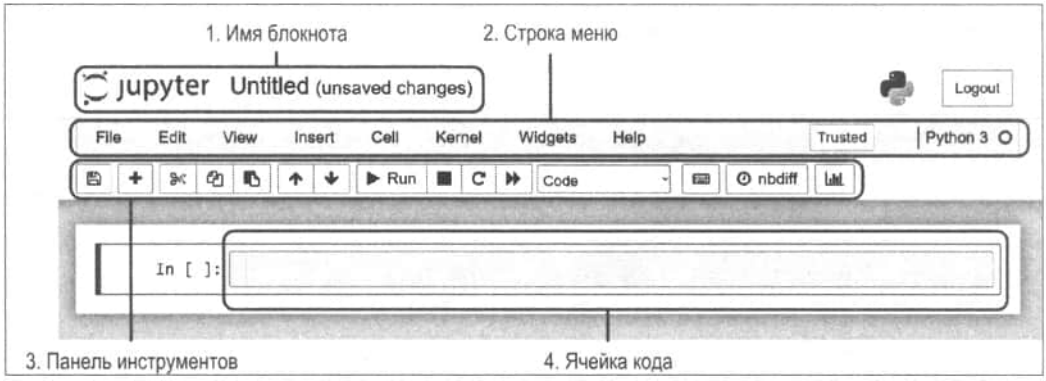


Рис. 10.2. Элементы интерфейса Jupyter

Имя блокнота: это имя файла с расширением **.ipynb**. Можно переименовать блокнот, щелкнув на имени и написав новое имя поверх текущего.

Строка меню: содержит различные операции для работы с блокнотом. Например, с помощью **File** вы можете открывать и закрывать блокноты. Их сохранение не вызывает никаких проблем, поскольку блокноты Jupyter автоматически сохраняются каждые 2 мин. Если вам понадобится конвертировать блокнот в скрипт **.py** Python или другой распространенный тип файла, сделать это можно, выбрав последовательно **File** (Файл) | **Download as** (Загрузить как). Также имеется меню **Help**, содержащее несколько руководств и ссылок на справочную документацию. В этом меню вы также найдете описание «горячих» клавиш Jupyter.

Ранее я упоминал, что *ядро* (kernel) служит для взаимодействия Jupyter с Python в фоновом режиме («под капотом»). Опция **Kernel** в строке меню содержит полезные операции для работы с ним. Иногда для того, чтобы ваш код на Python заработал, требуется лишь перезапустить ядро. Это можно сделать, выполнив последовательно: **Kernel** (Ядро) | **Restart** (Перезапустить).

Непосредственно под строкой меню расположена *панель инструментов*. На ней находятся полезные иконки для работы с вашим блокнотом, которые могут быть более удобными, чем навигация по меню: например, несколько иконок относятся к взаимодействию с ядром.

Вы можете также вставлять и перемещать ячейки в блокноте, где вы будете проводить бóльшую часть времени в Jupyter. Для начала давайте сделаем в панели инструментов еще одну вещь: найдите раскрывающееся меню, в котором в настоящее время установлено значение **Code**; измените его на **Markdown**.

Теперь перейдите в вашу первую *ячейку кода* и введите в ней фразу «Hello, Jupyter!» Снова перейдите в панель инструментов и нажмите иконку **Run** (Запустить). Произойдет следующее. Вы увидите, что ячейка `Hello, Jupyter!` отображается так, как она выглядела бы в документе текстового редактора. Далее вы увидите, что под предыдущей ячейкой находится новая ячейка кода, настроенная на ввод новой информации. Ваш блокнот будет выглядеть так, как показано на рис. 10.3.

Рис. 10.3. «Hello, Jupyter!»

Теперь вернитесь на панель инструментов и еще раз выберите в раскрывающемся меню **Markdown**. Как вы уже понимаете, блокноты Jupyter состоят из модульных ячеек разных типов. Мы сосредоточим внимание на двух наиболее популярных: **Markdown** и **Code**. *Markdown* — это язык разметки для форматирования простого текста посредством символов обычной клавиатуры.

Вставьте в пустую ячейку следующий текст:

```
# Большой заголовок 1
## Меньший заголовок 2
### Еще меньшие заголовки
#### И еще

* Использование одной звездочки выделяет курсивом *

** Использование двух звездочек выделяет жирным шрифтом **
```

```
- Используйте тире...
- чтобы составлять маркированные списки

Обратные апострофы выделяют код как текст `фиксированной ширины`
```

Теперь запустите ячейку: это можно сделать из панели инструментов или с помощью сочетания клавиш <Alt>+<Enter> для Windows или <Option>+<Return> для Mac. Вы получите результат, показанный на рис. 10.4.

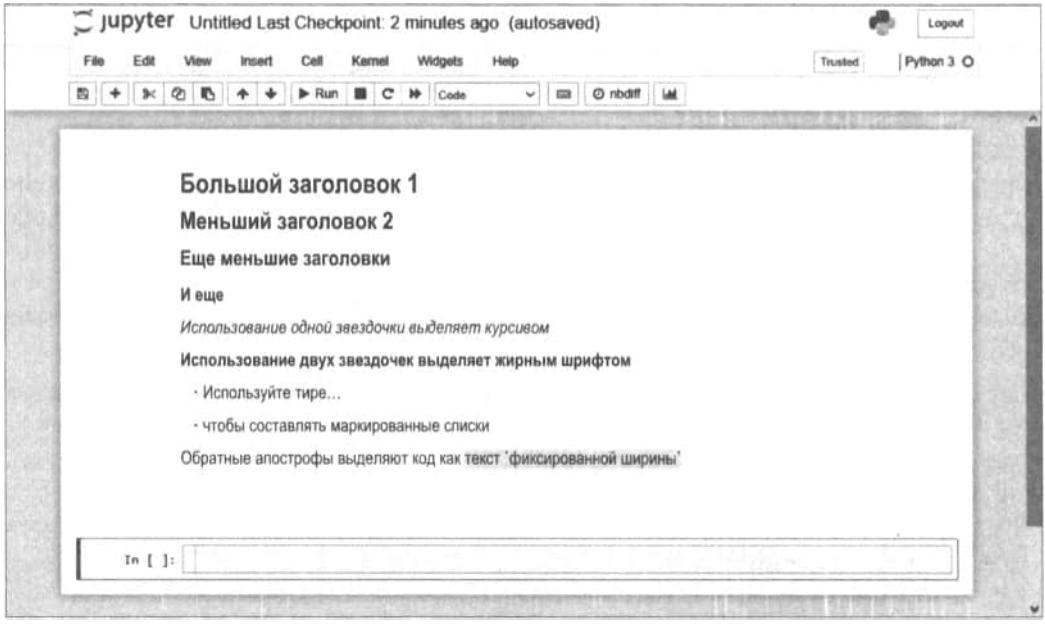


Рис. 10.4. Пример форматирования **Markdown** в Jupyter

Чтобы узнать больше о Markdown, перейдите в раздел **Help** (Справка) в строке меню. Его стоит изучить, чтобы создавать элегантные блокноты, которые могут включать в себя изображения, уравнения и многое другое. Но в этой книге мы сосредоточимся на блоке **Code**, поскольку именно туда ведет нас исполняемый Python. Сейчас вы должны находиться на третьей ячейке кода; оставьте ее в формате **Code**. Наконец, мы переходим к написанию кода на Python.

Python, так же как Excel и R, можно использовать в качестве удобного калькулятора. В табл. 10.1 указаны наиболее популярные арифметические операторы в Python.

Введите следующие арифметические действия, а затем запустите ячейки:

```
In [1]: 1 + 1
Out[1]: 2

In [2]: 2 * 4
Out[2]: 8
```

Таблица 10.1. *Основные арифметические операторы в Python*

Оператор	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
**	Возведение в степень
%%	Остаток от деления
//	Целочисленное деление

При выполнении блоков кода в Jupyter их вводам и выводам присваиваются нумерованные метки: `In []` (в) и `Out []` (из) соответственно.

Python также следует порядку операций; давайте попробуем запустить несколько примеров из одной ячейки:

```
In [3]: # Умножение перед сложением
        3 * 5 + 6
        2 / 2 - 7 # Деление перед вычитанием
Out[3]: -6.0
```

По умолчанию блокноты Jupyter возвращают только результат последнего запущенного кода в ячейке, поэтому разделим код на две части. Можно разбить ячейку под курсором как на Windows, так и на Mac с помощью сочетания клавиш <Ctrl>+<Shift>+<-> (минус):

```
In [4]: # Умножение перед сложением
        3 * 5 + 6

Out[4]: 21
In [5]: 2 / 2 - 7 # Деление перед вычитанием

Out[5]: -6.0
```

И, конечно же, в Python используются комментарии к коду. Как и в R, они начинаются со знака «решетка» (#), и точно так же их рекомендуется размещать на отдельных строках.

Так же как Excel и R, в Python имеется множество функций для чисел и символов:

```
In [6]: abs(-100)

Out[6]: 100

In [7]: len('Hello, world!')

Out[7]: 13
```

Как и R, но в отличие от Excel, Python чувствителен к регистру. Это означает, что работает *только* выражение `abs()`, не `ABS()` или `Abs()`:

```
In [8]: ABS(-100)

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-20-a0f3f8a69d46> in <module>
----> 1 print(ABS(-100))
      2 print(Abs(-100))

NameError: name 'ABS' is not defined
```

Python и отступы

Отступы в Python — это не просто пожелание: они могут быть *необходимы* для выполнения кода. Это связано с тем, что язык использует правильные отступы для компиляции и выполнения блоков кода или фрагментов Python, которые должны выполняться как единое целое. В этой книге вы не столкнетесь с этой проблемой, но, продолжив осваивать другие возможности Python, такие как написание функций или циклов, вы увидите, насколько важным является отступ.

Так же как и в R, вы можете использовать оператор `?` для получения информации о функциях, пакетах и т. д. Откроется окно, показанное на рис. 10.5, которое затем можно развернуть на весь экран или открыть в новом окне.

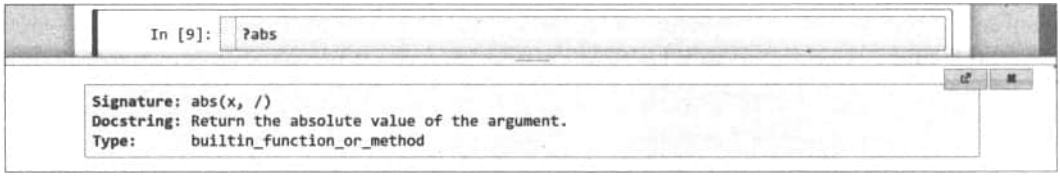


Рис. 10.5. Запуск документации в блокнотах Jupyter

Операторы сравнения в основном работают в Python так же, как в R и Excel; в Python результаты возвращаются как `True` (истинно) или `False` (ложно).

```
In [10]: # 3 больше 4?
         3 > 4

Out[10]: False
```

Так же как и в R, проверка равенства двух значений выполняется с помощью оператора `==`; а одиночный знак равенства `=` используется для присваивания объектов. В Python для присваивания объектов мы будем всегда использовать оператор `=`:

```
In [11]: # Присваивание объекта в Python
         my_first_object = abs(-100)
```

Возможно, вы заметили, что для ячейки 11 отсутствует компонент `Out []`. Причина заключается в том, что мы только *присвоили* объект; мы ничего не печатали. Давайте сделаем это сейчас:

```
In [12]: my_first_object

Out[12]: 100
```

Имена объектов в Python должны начинаться с буквы или символа подчеркивания, а остальная часть имени может содержать только буквы, цифры или символы подчеркивания. Кроме того, существует несколько запрещенных символов. Тем не менее для именования объектов в Python достаточно широкие возможности, но *возможность* назвать объект **scooby_doo** не значит, что вы должны именно так и сделать.

Python и PEP 8

Python Foundation использует документ «Предложения по улучшению Python» (Python Enhancement Proposals, PEPs) для объявления об изменениях и новых возможностях языка. Документ PEP 8 содержит руководство по стилю, являющееся универсальным стандартом для написания кода на Python. Среди его многочисленных правил и рекомендаций — соглашения об именовании объектов, добавлении комментариев и многое другое. Вы можете ознакомиться с полным руководством PEP 8 на сайте Python Foundation.

Так же как и в R, объекты в Python могут состоять из разных типов данных. В табл. 10.2 показаны некоторые основные типы данных в Python. Вы замечаете сходства и различия с R?

Таблица 10.2. *Основные типы объектов в Python*

Тип данных	Пример
String (строка)	`'Python', 'G. Mount', 'Hello, world!'`
Float (числа с плавающей точкой)	`6.2, 4.13, 3.1`
Integer (целые числа)	`3, -1, 12`
Boolean (булевы/логические значения)	`True, False`

Давайте присвоим несколько объектов. Определить, к какому типу они относятся, можно с помощью функции `type()`:

```
In [13]: my_string = 'Hello, world'
         type(my_string)

Out[13]: str

In [14]: # Двойные кавычки подходят и для строк
         my_other_string = "We're able to code Python!"
         type(my_other_string)

Out[14]: str

In [15]: my_float = 6.2
         type(my_float) Out[15]: float

Out[15]: float
```

```
In [16]: my_integer = 3
         type(my_integer)

Out[16]: int

In [17]: my_bool = True
   type(my_bool)

Out[17]: bool
```

Вы уже имели дело с объектами в R, поэтому вы, вероятно, не удивлены, что в Python их можно использовать как часть операций:

```
In [18]: # my_float равен 6.1?
         my_float == 6.1

Out[18]: False

In [19]: # Сколько символов в my_string?
         # (Та же функция, что и в Excel)
         len(my_string)

Out[19]: 12
```

С функциями в Python тесно связаны *методы*. Метод присоединяется к объекту с помощью точки и что-то делает с объектом. Например, чтобы сделать все буквы в строковом объекте заглавными, можно использовать метод `upper()`:

```
In [20]: my_string.upper()

Out[20]: 'HELLO, WORLD'
```

И функции, и методы используются для выполнения операций с объектами; в этой книге мы будем использовать и то и другое. Вы наверняка предполагаете, что Python, как и R, умеет хранить несколько значений в одном объекте. Но прежде чем перейти к этому, давайте рассмотрим, как работают *модули* в Python.

Модули в Python

Python был разработан как универсальный язык программирования, поэтому даже некоторые простейшие функции для работы с данными в стандартной поставке отсутствуют. Например, нам не удастся найти функцию для извлечения квадратного корня из числа:

```
In [21]: sqrt(25)

         ---------------------------------------------------------
         NameError                 Traceback (most recent call last)
         <ipython-input-18-1bf613b64533> in <module>
         ----> 1 sqrt(25)

         NameError: name 'sqrt' is not defined[1]
```

[1] Имя «sqrt» не определено. — *Ред.*

На самом деле эта функция в Python *имеется*. Но для получения доступа к ней необходимо открыть *модуль*, похожий на пакет в R. Несколько модулей поставляются как часть стандартной библиотеки Python; например, модуль `math` содержит множество математических функций, включая `sqrt()`. Можно вызвать этот модуль в нашей сессии с помощью инструкции `import`:

```
In [22]: import math
```

Инструкции — это указания, говорящие интерпретатору, что нужно делать. Мы, собственно, только что сказали Python: нужно импортировать модуль `math`. Теперь функция `sqrt()` должна быть доступна. Попробуем:

```
In [23]: sqrt(25)

---------------------------------------------------------
NameError                 Traceback (most recent call last)
<ipython-input-18-1bf613b64533> in <module>
----> 1 sqrt(25)

NameError: name 'sqrt' is not defined
```

Я совершенно честен с вами относительно функции `sqrt()`. Причина, по которой мы всё еще получаем ошибку, заключается в том, что необходимо четко указать Python, *откуда* берется эта функция. Это можно сделать, добавив префикс к имени модуля перед функцией следующим образом:

```
In [24]: math.sqrt(25)

Out[24]: 5.0
```

В стандартной библиотеке Python (Standard Library) огромное количество полезных модулей. К тому же существуют тысячи модулей сторонних производителей, объединенных в *пакеты* и представленных в каталоге/указателе пакетов Python (Python Package Index, PyPI). Существует стандартная система управления пакетами — **pip** (Package Installer for Python), которую можно использовать для установки как из каталога пакетов Python (PyPI), так и из внешних ресурсов.

Дистрибутив Anaconda значительно упростил работу с пакетами. Прежде всего, некоторые наиболее популярные пакеты Python поставляются предустановленными. Кроме того, Anaconda включает в себя компоненты, обеспечивающие совместимость всех пакетов на вашем компьютере. Поэтому предпочтительно устанавливать пакеты непосредственно с дистрибутива Anaconda, а не из pip. Установка пакета Python обычно выполняется из командной строки; вы с ней уже работали в Anaconda Prompt (Windows) или Terminal (Mac). Однако можно выполнить код командной строки в Jupyter, поставив в начале восклицательный знак. Давайте установим `plotly`, популярный пакет для визуализации данных, из Anaconda. Используем инструкцию `conda install`:

```
In [25]: !conda install plotly
```

Не все пакеты можно установить с дистрибутива Anaconda; в таком случае воспользуемся системой pip. Давайте выполним установку пакета `pyxlsb`, который может использоваться для считывания двоичных Excel-файлов `.xlsb` в язык Python:

```
In [26]: !pip install pyxlsb
```

Хотя загрузка пакетов непосредственно из Jupyter удобна, может быть неприятным сюрпризом попытка запуска вашего блокнота только для того, чтобы столкнуться с длительными или ненужными загрузками. Поэтому принято комментировать команды установки, что я и делаю в репозитории к данной книге.

Если вы используете Anaconda для запуска Python, сначала лучше выполнить установку с помощью `conda` и только затем — с pip, если пакет недоступен.

Обновление Python, Anaconda и пакетов Python

В табл. 10.3 перечислены еще некоторые полезные команды для управления средой Python. Также можно устанавливать и настраивать пакеты Anaconda с помощью такого метода управления интерфейсом, как **point-and-click** (наведи и щелкни), используя Anaconda Navigator, который устанавливается вместе с Anaconda Individual Edition. Запустите приложение на своем компьютере, затем перейдите в меню **Help** (Справка), чтобы подробнее ознакомиться с документацией.

Таблица 10.3. *Полезные команды для управления пакетами Python*

Команда	Описание
`conda update anaconda`	Обновление дистрибутива Anaconda
`conda update python`	Обновление версии Python
`conda update --all`	Обновление всех возможных пакетов, загруженных через `conda`
`pip list --outdated`	Список всех пакетов, загруженных через `pip`, которые могут быть обновлены

Заключение

В этой главе вы узнали, как работать с объектами и пакетами в Python, а также научились работать с блокнотами Jupyter.

Упражнения

Следующие упражнения обеспечивают дополнительную практику и помогут лучше разобраться в темах, изученных в этой главе.

1. В новом блокноте Jupyter выполните следующие действия.
   - Присвойте `a` сумму 1 и –4.
   - Присвойте `b` абсолютное значение `a`.

- Присвойте `d` разницу `b` минус 1.
- Ответьте на вопрос: будет ли `d` больше 2?

2. В стандартную библиотеку Python (Python Standard Library) включен модуль `random`, содержащий функцию `randint()`. Эта функция работает как функция `RANDBETWEEN()` в Excel; например, `randint(1, 6)` возвратит целочисленное значение между 1 и 6. Используйте эту функцию для нахождения среднего числа между 0 и 36.
3. Стандартная библиотека Python также включает в себя модуль под названием `this`. Что произойдет, когда вы импортируете этот модуль?
4. Загрузите пакет `xlutils` из Anaconda, затем воспользуйтесь оператором `?` для получения доступной документации по нему.

Я снова призываю вас как можно скорее начать использовать этот язык в вашей каждодневной работе, даже если поначалу он будет просто калькулятором. Также можно пробовать выполнять одни и те же задачи на R и Python, сравнивая и противопоставляя их. Если вы усвоили предыдущий материал и поняли, как связать R и Excel, то же самое будет работать и для связи Python с R.

ГЛАВА 11

Структуры данных в Python

В *главе 10* вы познакомились с простыми типами объектов в Python, такими как строки, целые числа и логические (булевы) значения. Теперь давайте рассмотрим группировку нескольких значений в так называемую *коллекцию*. Python по умолчанию поставляется с несколькими типами коллекций объектов. Мы начнем эту главу с типа *list* (список). Можно поместить значения в список, отделяя каждую запись запятой и размещая результаты в квадратных скобках:

```
In [1]: my_list = [4, 1, 5, 2]
        my_list

Out[1]: [4, 1, 5, 2]
```

Этот объект содержит все целые числа, но сам *не является* целочисленным типом данных: он является списком.

```
In [2]: type(my_list)
Out[2]: list
```

В список можно включить все различные виды данных... и даже другие списки:

```
In [3]: my_nested_list = [1, 2, 3, ['Boo!', True]]
        type(my_nested_list)

Out[3]: list
```

Другие типы коллекций в базовом Python

Python включает несколько других типов коллекций объектов (данных), кроме списка, прежде всего — *dictionary* (словарь), а также многие другие в модуле стандартной библиотеки `collections`. Типы коллекций различаются способами хранения значений и могут быть проиндексированы или изменены.

Как видите, списки достаточно универсальны для хранения данных. Но сейчас нас действительно интересует работа с чем-то, что могло бы функционировать как диапазон в Excel или вектор в R, а затем мы перейдем к табличным данным. Соответствует ли нашим требованиям простой список? Давайте определим это, умножив простой `my_list` на 2:

```
In [4]: my_list * 2

Out[4]: [4, 1, 5, 2, 4, 1, 5, 2]
```

Вероятно, это *не то*, что вы ищете: Python понял вас буквально и удвоил *список*, а не цифры *внутри списка*. Можно получить желаемый результат самостоятельно: если вам приходилось работать с циклами, вы можете выполнить здесь один из них, чтобы каждый элемент умножить на 2. Ничего страшного, если вы не работали с циклами ранее: оптимальный вариант — импортировать модуль, который упрощает вычисление в Python. Для этого используем модуль `numpy`, включенный в дистрибутив Anaconda.

Массивы *NumPy*

```
In [5]: import numpy
```

Как следует из названия[1], `numpy` является модулем для численных расчетов в Python, ставшим основой популярности Python как инструмента аналитики. Чтобы узнать больше о `numpy`, перейдите в раздел **Help** (Справка) в строке меню Jupyter и выберите **NumPy reference** (Справочное руководство NumPy). Сейчас мы сосредоточимся на *массиве* `numpy`. Это коллекция данных с элементами одного типа, которая может хранить данные с любым числом, или *n*, измерений. Мы сосредоточимся на *одномерном* массиве и преобразуем наш первый массив из списка с помощью функции `array()`:

```
In [6]: my_array = numpy.array([4, 1, 5, 2])
          my_array

Out[6]: array([4, 1, 5, 2])
```

На первый взгляд массив `numpy` очень похож на список; в конце концов, мы ведь просто создали его из списка. Но на самом деле это другой тип данных:

```
In [7]: type(my_list)

Out[7]: list

In [8]: type(my_array)

Out[8]: numpy.ndarray
```

Это `ndarray`, или *n*-мерный массив. Являясь другой структурой данных, он может вести себя по-другому с операциями. Например, если умножить массив `numpy`:

```
In [9]: my_list * 2

Out[9]: [4, 1, 5, 2, 4, 1, 5, 2]

In [10]: my_array * 2

Out[10]: array([ 8, 2, 10, 4])
```

[1] От *Numerical Python* — числовой (численный) Python. — *Ред.*

Во многом такое поведение должно напоминать вам диапазон в Excel или вектор в R. И действительно, как и векторы в R, массивы numpy будут *приводить* данные к одному типу:

```
In [11]: my_coerced_array = numpy.array([1, 2, 3, 'Boo!'])
         my_coerced_array

Out[11]: array(['1', '2', '3', 'Boo!'], dtype='<U11')
```

Типы данных в NumPy и Pandas

Вы увидите, что типы данных в numpy и более поздних версиях pandas работают немного иначе, чем в стандартном Python. Эти так называемые dtypes созданы для быстрого чтения и записи данных, а также для работы с низкоуровневыми языками программирования, такими как C или Fortran. Не стоит беспокоиться об используемых вами конкретных типах данных; сосредоточьтесь на общих типах, таких как числа с плавающей запятой, строки или булевы (логические) значения.

Как видите, numpy — это палочка-выручалочка при работе с данными в Python. Планируйте импортировать его *много раз*... что означает, много раз печатать его название. К счастью, облегчить ситуацию можно с помощью *псевдонима*. Используем ключевое слово as, чтобы присвоить массиву numpy его обычный псевдоним — np:

```
In [12]: import numpy as np
```

Это дает модулю временное, более простое для управления имя. Теперь каждый раз, когда необходимо вызвать код из numpy во время сессии Python, мы можем ссылаться на его псевдоним.

```
In [13]: import numpy as np
         # у numpy также есть функция sqrt():
         np.sqrt(my_array)

Out[13]: array([2.        , 1.        , 2.23606798, 1.41421356])
```

Помните, что псевдонимы являются *временными*, они работают в течение одной сессии Python. Псевдоним перестанет работать, если вы перезапустите ядро или создадите новый блокнот.

Индексирование и подмножества массивов *NumPy*

Давайте посвятим немного времени изучению способов извлечения отдельных элементов из массива numpy; проставим порядковый номер элемента в квадратные скобки непосредственно рядом с именем объекта:

```
In [14]: # Получить второй элемент ... так?
         my_array[2]

Out[14]: 5
```

Например, мы только что извлекли из массива второй элемент... *или нет*? Вернемся к массиву `my_array`; что *на самом деле* находится во второй позиции?

```
In [15]: my_array

Out[15]: array([4, 1, 5, 2])
```

Оказывается, во второй позиции расположена цифра 1, а цифра 5 — в *третьей*. Как объяснить это несоответствие? Причина в том, что Python считает не так, как это обычно делаем мы.

Чтобы разобраться в этой странной концепции, представьте следующую ситуацию: вы настолько увлеклись получением нового набора данных, что загрузили его несколько раз. Такая неосторожность приводит к получению серии файлов с такими именами:

- *dataset*.csv;
- *dataset* (1).csv;
- *dataset* (2).csv;
- *dataset* (3).csv.

Мы, люди, привыкли начинать счет с единицы. Но компьютеры зачастую начинают счет *с нуля*. Загрузка нескольких файлов — один из примеров: второй файл в действительности имеет имя `dataset (1)`, а не `dataset (2)`. Это называется *индексирование с нуля* и *всегда* делается в Python.

Индексирование с нуля и единицы

Компьютеры обычно начинают счет с нуля, но не всегда. Excel и R реализуют индексирование *с единицы*, когда считается, что первый элемент находится в позиции 1. Программисты могут придерживаться разных мнений относительно лучшего варианта, но вы должны уметь работать с тем и с другим.

Все это означает, что в Python индексирование с номером 1 возвращает значение во *второй* позиции, индексирование с номером 2 — в третьей и т. д.

```
In [16]: # *Теперь* давайте получим второй элемент
         my_array[1]

Out[16]: 1
```

Также можно выделить подмножество набора последовательных значений, что в Python называется *нарезкой*. Давайте попробуем найти элементы со второго по четвертый. Мы уже избавились от проблем, связанных индексированием с нуля; насколько это может быть сложно?

```
In [17]: # Давайте получим второй элемент... так?
         my_array[1:3]

Out[17]: array([1, 5])
```

Но погодите, это еще не всё. В дополнение к индексированию с нуля нарезка *исключает* конечный элемент. Это значит, что необходимо «добавить 1» к номеру второго элемента для получения желаемого диапазона:

```
In [18]: # *Теперь* получим элементы со второго по четвертый
         my_array[1:4]

Out[18]: array([1, 5, 2])
```

Вы можете делать с нарезкой в Python гораздо больше, например начать с конца объекта или выбрать все элементы с начала до определенной позиции. Сейчас важно помнить, что *Python использует индексирование с нуля.*

Двумерные массивы `numpy` могут служить в качестве табличных структур данных в Python, но все элементы должны быть одного и того же типа. При анализе данных в бизнес-контексте такое случается редко, поэтому мы перейдем к `pandas`, чтобы удовлетворить данное требование.

Кадры данных Pandas

Программная библиотека `pandas`, получившая свое название от «***pa**nel **da**ta of econometrics*» (панельные данные эконометрики), особенно полезна для обработки и анализа табличных данных. Так же как и массив `numpy`, она поставляется с дистрибутивом Anaconda. Ее типичный псевдоним — `pd`:

```
In [19]: import pandas as pd
```

Модуль `pandas` использует `numpy` в своей базе кода, и вы увидите некоторое сходство между ними. `pandas`, среди прочего, включает в себя одномерную структуру данных, называемую *серией* (series). Но его наиболее широко используемой структурой является двумерная структура *DataFrame*, кадр данных (звучит знакомо?). Кадр данных можно создавать из других типов данных, включая массивы `numpy`, с помощью функции `DataFrame`:

```
In [20]: record_1 = np.array(['Jack', 72, False])
         record_2 = np.array(['Jill', 65, True])
         record_3 = np.array(['Billy', 68, False])
         record_4 = np.array(['Susie', 69, False])
         record_5 = np.array(['Johnny', 66, False])

         roster = pd.DataFrame(data = [record_1,
             record_2, record_3, record_4, record_5],
               columns = ['name', 'height', 'injury'])

         roster

Out[20]:

              name height injury
         0    Jack     72  False
         1    Jill     65   True
```

```
2   Billy     68  False
3   Susie     69  False
4  Johnny     66  False
```

Обычно кадры данных содержат метки *имени* для каждого столбца. Также есть *индекс* по строкам, который (как вы уже догадались) по умолчанию начинается с 0. Наш набор данных слишком мал для изучения, поэтому давайте найдем что-нибудь еще. К сожалению, изначально Python не включает в себя кадры данных, но мы можем найти их с помощью пакета `seaborn`, который также устанавливается вместе с дистрибутивом Anaconda и часто имеет псевдоним `sns`. Функция `get_dataset_names()` возвратит список доступных для использования кадров данных:

```
In [21]: import seaborn as sns
         sns.get_dataset_names()

Out[21]:
        ['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes',
        'diamonds', 'dots', 'exercise', 'flights', 'fmri', 'gammas',
        'geyser', 'iris', 'mpg', 'penguins', 'planets', 'tips', 'titanic']
```

Название *iris* тоже звучит знакомо? Мы можем загрузить его в нашу сессию Python с помощью функции `load_data set()`, и напечатать первые пять строк, используя метод `head()`.

```
In [22]: iris = sns.load_dataset('iris')
         iris.head()
Out[22]:
            sepal_length  sepal_width  petal_length  petal_width  species
         0           5.1          3.5           1.4          0.2 setosa
         1           4.9          3.0           1.4          0.2 setosa
         2           4.7          3.2           1.3          0.2 setosa
         3           4.6          3.1           1.5          0.2 setosa
         4           5.0          3.6           1.4          0.2 setosa
```

Импорт данных в Python

Так же как и в R, данные обычно считываются из внешних файлов, а для этого необходимо работать с каталогами. В стандартную библиотеку Python входит модуль `os` для работы с путями к файлам и каталогами:

```
In [23]: import os
```

Для следующей части сохраните свой блокнот в главной папке репозитория к этой книге. По умолчанию Python устанавливает в качестве текущего рабочего каталога тот, в котором находится активный файл, поэтому не нужно беспокоиться об изменении каталога, как это делалось в R. По-прежнему можно проверить и изменить его с помощью функций `getcwd()` и `chdir()` из модуля `os` соответственно.

Python следует тем же общим правилам об относительных и абсолютных путях к файлам, что и R. Давайте посмотрим, сможем ли мы найти файл **test-file.csv** в ре-

позитории с помощью функции `isfile()`, которая находится в подмодуле `path` модуля `os`:

```
In [24]: os.path.isfile('test-file.csv')

Out[24]: True
```

Теперь попробуем найти этот файл, расположенный во вложенной папке **test-folder**:

```
In [25]: os.path.isfile('test-folder/test-file.csv')

Out[25]: True
```

Далее попробуйте поместить копию этого файла в папку, расположенную на один уровень выше вашей текущей папки. Вы найдете ее с помощью следующего кода:

```
In [26]: os.path.isfile('../test-file.csv')

Out[26]: True
```

Так же как и в R, вы, скорее всего, будете считывать данные из внешнего источника для работы ними в Python, и этот источник может быть практически каким угодно. В `pandas` входят функции для чтения данных, кроме всего прочего, из файлов **.xlsx** и **.csv** в кадры данных. Чтобы увидеть это, выполним чтение наших проверенных наборов данных **star.xlsx** и **districts.csv** из репозитория к книге. Для чтения рабочих книг в Excel используется функция `read_excel()`:

```
In [27]: star = pd.read_excel('datasets/star/star.xlsx')
         star.head()
Out[27]:
   tmathssk  treadssk            classk  totexpk   sex freelunk   race
0       473       447       small.class        7  girl       no  white
1       536       450       small.class       21  girl       no  black
2       463       439  regular.with.aide        0   boy      yes  black
3       559       448           regular       16   boy       no  white
4       489       447       small.class        5   boy      yes  white
   schidkn
0       63
1       20
2       19
3       69
4       79
```

Аналогичным образом можно использовать `pandas` для чтения файлов **.csv** с помощью функции `read_csv()`:

```
In [28]: districts = pd.read_csv('datasets/star/districts.csv')
         districts.head()
Out[28]:
            schidkn      school_name       county
         0        1          Rosalia  New Liberty
         1        2  Montgomeryville       Topton
```

```
2         3            Davy     Wahpeton
3         4         Steelton    Palestine
4         6       Tolchester      Sattley
```

Если вам потребуется, например, читать другие типы файлов Excel или определенные диапазоны и листы, обратитесь к документации `pandas`.

Исследование кадра данных

Продолжим исследовать кадр данных **star**. Метод `info()` позволит получить некоторые важные сведения, такие как размеры и типы столбцов:

```
In [29]: star.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5748 entries, 0 to 5747
Data columns (total 8 columns):
#   Столбец      Ненулевой  Счет  Dtype
---  -------      --------------  -----
0   tmathssk     5748  non-null   int64
1   treadssk     5748  non-null   int64
2   classk       5748  non-null   object
3   totexpk      5748  non-null   int64
4   sex          5748  non-null   object
5   freelunk     5748  non-null   object
6   race         5748  non-null   object
7   schidkn      5748  non-null   int64
dtypes: int64(4), object(4)
memory usage: 359.4+ KB
```

Мы можем получить описательную статистику с помощью метода `describe()`:

```
In [30]: star.describe()

Out[30]:
              tmathssk     treadssk      totexpk      schidkn
       count  5748.000000  5748.000000  5748.000000  5748.000000
       mean    485.648051   436.742345     9.307411    39.836639
       std      47.771531    31.772857     5.767700    22.957552
       min     320.000000   315.000000     0.000000     1.000000
       25%     454.000000   414.000000     5.000000    20.000000
       50%     484.000000   433.000000     9.000000    39.000000
       75%     513.000000   453.000000    13.000000    60.000000
       max     626.000000   627.000000    27.000000    80.000000
```

По умолчанию `pandas` содержит только описательную статистику числовых переменных. Переопределить это можно с помощью аргумента `include = 'all'`.

```
In [31]: star.describe(include = 'all')
```

```
Out[31]:
          tmathssk     treadssk             classk      totexpk   sex  \
count   5748.000000  5748.000000               5748  5748.000000  5748
unique          NaN          NaN                  3          NaN     2
top             NaN          NaN  regular.with.aide          NaN   boy
freq            NaN          NaN               2015          NaN  2954
mean     485.648051   436.742345                NaN     9.307411   NaN
std       47.771531    31.772857                NaN     5.767700   NaN
min      320.000000   315.000000                NaN     0.000000   NaN
25%      454.000000   414.000000                NaN     5.000000   NaN
50%      484.000000   433.000000                NaN     9.000000   NaN
75%      513.000000   453.000000                NaN    13.000000   NaN
max      626.000000   627.000000                NaN    27.000000   NaN

       freelunk   race      schidkn
count      5748   5748  5748.000000
unique        2      3          NaN
top          no  white          NaN
freq       2973   3869          NaN
mean        NaN    NaN    39.836639
std         NaN    NaN    22.957552
min         NaN    NaN     1.000000
25%         NaN    NaN    20.000000
50%         NaN    NaN    39.000000
75%         NaN    NaN    60.000000
max         NaN    NaN    80.000000
```

`NaN` (Not a Number) — это специальное значение в `pandas` для обозначения отсутствующих или недоступных данных, таких как стандартное отклонение категориальной переменной.

Индексирование и подмножества кадров данных

Давайте вернемся к небольшому кадру данных **roster**; будем обращаться к различным элементам по их позиции в строке и столбце. Для индексирования кадра данных можно использовать метод `iloc`, или `integer location` (целочисленное расположение). Запись в квадратных скобках должна быть вам знакома, но на этот раз нам нужно индексировать и по строке, и по столбцу (напомню, и то и другое начинаются с нуля). Давайте испытаем это на кадре данных **roster**, созданном нами ранее.

```
In [32]: # Первая строка, первый столбец кадра данных
         roster.iloc[0, 0]

Out[32]: 'Jack'
```

Здесь также можно использовать нарезку, чтобы захватить несколько строк или столбцов:

```
In [33]: # Со второй по четвертую строки, с первого по третий столбцы

Out[33]:
     name height injury
 1   Jill     65   True
 2  Billy     68  False
 3  Susie     69  False
```

Чтобы проиндексировать весь столбец по имени, используем соответствующий метод `loc`. Оставим пустой фрагмент в первой позиции индекса, чтобы захватить все строки, а затем укажем имя интересующего нас столбца:

```
In [34]: # Выбрать все строки в столбце name
         roster.loc[:, 'name']

Out[34]:
        0     Jack
        1     Jill
        2    Billy
        3    Susie
        4   Johnny
        Name: name, dtype: object
```

Запись кадров данных

Модуль `pandas` также включает функции для записи кадров данных в файлы формата **.csv** и рабочие книги **.xlsx** с помощью методов `write_csv()` и `write_xlsx()` соответственно:

```
In [35]: roster.to_csv('output/roster-output-python.csv')
         roster.to_excel('output/roster-output-python.xlsx')
```

Заключение

За короткий срок вы прошли большой путь от одноэлементных объектов к спискам, массивам `numpy` и, наконец, к кадрам данных `pandas`. Надеюсь, вам удалось увидеть развитие этих структур данных и связь между ними, оценив при этом дополнительные преимущества представленных пакетов. Следующие главы, посвященные Python, в значительной степени ориентированы на `pandas`, который, как вы уже поняли, опирается на `numpy`, а также на основные правила Python, такие как индексирование с нуля.

Упражнения

В этой главе вы узнали, как работать с несколькими структурами данных и типами коллекций в Python. Следующие упражнения позволят дополнительно попрактиковаться и лучше понять изученные темы.

1. «Нарежьте» следующий массив таким образом, чтобы получить элементы с третьего по пятый.

```
practice_array = ['I', 'am', 'having', 'fun', 'with', 'Python']
```

2. Загрузите кадр данных **tips** из пакета `seaborn`.
   - Распечатайте некоторую информацию об этом кадре данных, например количество наблюдений и тип каждого столбца.
   - Распечатайте описательную статистику этого кадра данных.
3. Репозиторий к книге содержит файл **ais.xlsx** в подпапке **ais** папки **datasets**. Считайте его в Python как кадр данных.
   - Распечатайте первые пять строк этого кадра данных.
   - Запишите только столбец **sport** этого кадра данных в Excel в виде файла **sport.xlsx**.

ГЛАВА 12

Обработка и визуализация данных в Python

В *главе 8* вы узнали, как обрабатывать и визуализировать данные с помощью набора пакетов `tidyverse`. Здесь мы попрактикуемся с аналогичными методами на том же наборе данных **star**, но на этот раз в Python. В частности, мы будем использовать модули `pandas` и `seaborn` для обработки и визуализации данных соответственно. Это не будет полным руководством по использованию этих модулей или Python для анализа данных, но, напротив, только побудит вас к самостоятельным исследованиям.

Я буду, насколько это возможно, повторять шаги и выполнять те же операции, которые мы выполняли в *главе 8*. Поскольку они вам уже знакомы, я буду уделять внимание не столько способам обработки и визуализации данных, сколько тому, как выполнять эти операции в Python. Итак, давайте загрузим необходимые модули и начнем работать с набором данных **star**. Третий модуль, `matplotlib`, вам незнаком и будет использоваться для дополнения нашей работы в `seaborn`. Этот модуль поставляется уже установленным в Anaconda. Мы будем использовать вложенный модуль `pyplot` с псевдонимом `plt`.

```
In [1]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt

        star = pd.read_excel('datasets/star/star.xlsx')
        star.head()
Out[1]:
   tmathssk  treadssk            classk  totexpk   sex freelunk   race  \
   0       473       447       small.class       7  girl     no  white
   1       536       450       small.class      21  girl     no  black
   2       463       439  regular.with.aide       0   boy    yes  black
   3       559       448           regular      16   boy     no  white
   4       489       447       small.class       5   boy    yes  white

     schidkn

   0       63
   1       20
   2       19
   3       69
   4       79
```

Постолбцовые операции

В *главе 11* вы узнали, что модуль `pandas` попытается преобразовать одномерные структуры данных в серии (series). Этот, казалось бы, тривиальный момент будет весьма важен при выборе столбцов. Рассмотрим пример: предположим, мы хотели *всего лишь* сохранить столбец **tmathssk** из нашего кадра данных. Можно сделать это при помощи уже знакомой записи с одиночными скобками, но технически это даст в результате серии, а не кадр данных:

```
In [2]: math_scores = star['tmathssk']
        type(math_scores)

Out[2]: pandas.core.series.Series
```

Вероятно, лучше получить кадр данных, если мы не уверены, хотим ли, чтобы **math_scores** оставался одномерной структурой. Для этого можно использовать два набора скобок вместо одного:

```
In [3]: math_scores = star[['tmathssk']]
        type(math_scores)

Out[3]: pandas.core.frame.DataFrame
```

Следуя этому шаблону, мы можем оставить в наборе **star** только нужные нам столбцы. Для подтверждения я буду использовать атрибут `columns`.

```
In [4]: star = star[['tmathssk','treadssk','classk','totexpk','schidkn']]
        star.columns

Out[4]: Index(['tmathssk', 'treadssk', 'classk',
               'totexpk', 'schidkn'], dtype='object')
```

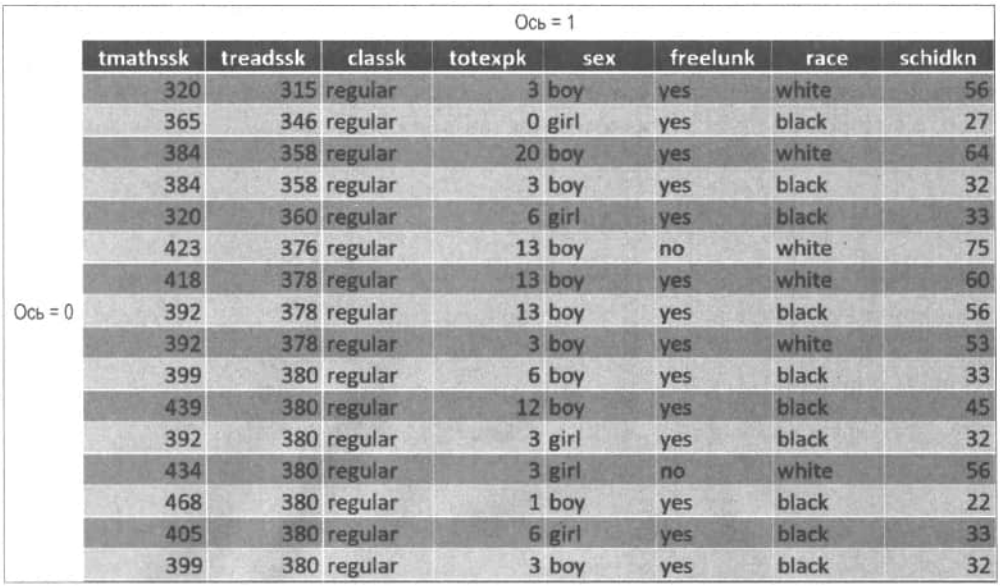
Объектно-ориентированное программирование в Python

До сих пор мы встречались с методами и функциями в Python — это то, что объекты могут выполнять. Вместе с тем атрибуты представляют некоторое *состояние* самого объекта. Они прикрепляются к имени объекта с помощью точки; в отличие от методов, скобки не используются. Атрибуты, функции и методы являются элементами *объектно-ориентированного программирования* (ООП) — парадигмы, предназначенной для структурирования работы в простые и многократно используемые фрагменты кода. Чтобы узнать больше о том, как ООП работает в Python, обратитесь к книге Алекса Мартелли и соавт. «Основы Python в двух словах (Alex Martelli et al. Python in a Nutshell).

Удалить определенные столбцы можно, используя метод `drop()`, с помощью которого можно удалять столбцы или строки, поэтому необходимо указать, какие именно, с помощью аргумента `axis`. В `pandas` строки отображаются по оси 0, а столбцы — по оси 1, как показано на рис. 12.1.

Для удаления столбца **schidkn** используйте следующий код:

```
In [5]: star = star.drop('schidkn', axis=1)
        star.columns
```



	tmathssk	treadssk	classk	totexpk	sex	freelunk	race	schidkn
	320	315	regular	3	boy	yes	white	56
	365	346	regular	0	girl	yes	black	27
	384	358	regular	20	boy	yes	white	64
	384	358	regular	3	boy	yes	black	32
	320	360	regular	6	girl	yes	black	33
	423	376	regular	13	boy	no	white	75
	418	378	regular	13	boy	yes	white	60
	392	378	regular	13	boy	yes	black	56
	392	378	regular	3	boy	yes	white	53
	399	380	regular	6	boy	yes	black	33
	439	380	regular	12	boy	yes	black	45
	392	380	regular	3	girl	yes	black	32
	434	380	regular	3	girl	no	white	56
	468	380	regular	1	boy	yes	black	22
	405	380	regular	6	girl	yes	black	33
	399	380	regular	3	boy	yes	black	32

Рис. 12.1. Оси кадра данных в **pandas**

```
Out[5]: Index(['tmathssk', 'treadssk', 'classk', 'totexpk'],
          dtype='object')
```

Теперь узнаем, как получить новые столбцы кадра данных. Это можно сделать с помощью записи с использованием скобок — на этот раз я хочу получить результат в виде серий (series), поскольку каждый столбец кадра данных в действительности является серией (так же как каждый столбец в кадре данных R представляет собой вектор). Теперь я подсчитаю средние баллы по математике и чтению:

```
In [6]: star['new_column'] = star['tmathssk'] + star['treadssk']
        star.head()
```

```
Out[6]:
   tmathssk  treadssk             classk  totexpk  new_column
0       473       447        small.class        7         920
1       536       450        small.class       21         986
2       463       439  regular.with.aide        0         902
3       559       448            regular       16        1007
4       489       447        small.class        5         936
```

И снова: **new_column** не является описательным именем переменной. Давайте исправим это с помощью функции `rename()`. Используем аргумент `columns` и передадим в него данные в формате, который вам, скорее всего, незнаком:

```
In [7]: star = star.rename(columns = {'new_column':'ttl_score'})
        star.columns
```

```
Out[7]: Index(['tmathssk', 'treadssk', 'classk', 'totexpk', 'ttl_score'],
          dtype='object')
```

Фигурные скобки, использовавшиеся в последнем примере, являются *словарем* (dictionary) Python. Словари — это коллекции пар *ключ — значение*, в которых ключ и значение каждого элемента разделены двоеточием. Это базовая структура данных Python, с которой следует познакомиться поближе при дальнейшем изучения языка.

Построчные операции

Теперь перейдем к обычным построчным операциям. Начнем с сортировки, которую можно выполнить в `pandas` с помощью метода `sort_values()` . Передадим аргументу `by` список столбцов, по которым хотим выполнить сортировку, в их соответствующем порядке:

```
In [8]: star.sort_values(by=['classk', 'tmathssk']).head()

Out[8]:
     tmathssk  treadssk   classk  totexpk  ttl_score
309       320       360  regular        6
680
1470      320       315  regular        3        635
2326      339       388  regular        6        727
2820      354       398  regular        6        752
4925      354       391  regular        8        745
```

По умолчанию все столбцы сортируются по возрастанию. Чтобы это изменить, можно включить еще один аргумент, `ascending`, который будет содержать список флажков `True`/`False`. Давайте отсортируем набор данных **star** по размеру класса (**classk**) по возрастанию, а также по баллам по математике (`treadssk`) по убыванию. Поскольку мы не назначаем результат сортировки набору **star**, она не будет постоянной для этого набора данных.

```
In [9]: # Сортировать по размеру класса по возрастанию
        # и по баллам по математике по убыванию
star.sort_values(by=['classk', 'tmathssk'], ascending=[True, False]).head()

Out[9]:
      tmathssk  treadssk   classk  totexpk  ttl_score
724        626       474  regular       15       1100
1466       626       554  regular       11       1180
1634       626       580  regular       15       1206
2476       626       538  regular       20       1164
2495       626       522  regular        7       1148
```

Чтобы отфильтровать кадр данных, сначала используем условную логику для создания серий (Series) флажков `True`/`False`, указывающих, соответствует ли каждая строка определенным критериям. Затем сохраним в кадре данных только строки, в которых записи в сериях помечены как `True`. Например, сохраним только строки, в которых `classk` равен `small.class`.

```
In [10]: small_class = star['classk'] == 'small.class'
         small_class.head()

Out[10]:
        0    True
        1    True
        2   False
        3   False
        4    True
        Name: classk, dtype: bool
```

Теперь можно выполнить фильтрацию по результирующим сериям с помощью скобок. Для подтверждения количества строк и столбцов в новом кадре данных используем атрибут `shape`:

```
In [11]: star_filtered = star[small_class]
         star_filtered.shape

Out[11]: (1733, 5)
```

`star_filtered` будет содержать меньше строк, чем набор **star**, но такое же количество столбцов:

```
In [12]: star.shape

Out[12]: (5748, 5)
```

Далее попробуем найти записи, в которых `treadssk` больше или равен 500:

```
In [13]: star_filtered = star[star['treadssk'] >= 500]
         star_filtered.shape

Out[13]: (233, 5)
```

Также можно выполнять фильтрацию по нескольким условиям, используя операторы `and/or`. Как и в R, в Python знаки `&` и `|` обозначают «и» и «или» соответственно. Давайте передадим оба предыдущих критерия в один оператор, поместив каждый из них в круглые скобки и связав их знаком `&`:

```
In [14]: # Найти все записи с баллом по чтению не менее 500 и в маленьком классе
         star_filtered = star[(star['treadssk'] >= 500) &
                              (star['classk'] == 'small.class')]
         star_filtered.shape

Out[14]: (84, 5)
```

Агрегирование и объединение данных

Для группировки наблюдений в кадре данных будем использовать метод `groupby()`. Распечатав `star_grouped`, вы увидите, что это объект `DataFrameGroupBy`:

```
In [15]: star_grouped = star.groupby('classk')
         star_grouped
```

```
Out[15]: <pandas.core.groupby.generic.DataFrameGroupBy
           object at 0x000001EFD8DFF388>
```

Теперь можно выбрать другие поля для агрегирования в этом сгруппированном кадре данных. В табл. 12.1 перечислены основные методы агрегирования.

Таблица 12.1. *Полезные методы агрегирования в pandas*

Метод	Тип агрегирования
`sum()`	Сумма
`count()`	Количество значений
`mean()`	Среднее значение
`max()`	Максимальное значение
`min()`	Минимальное значение
`std()`	Стандартное отклонение

Далее приведен средний балл по математике для каждого размера класса:

```
In [16]: star_grouped[['tmathssk']].mean()

Out[16]:
                      tmathssk
   classk
   regular            483.261000
   regular.with.aide  483.009926
   small.class        491.470283
```

Теперь найдем наивысший общий балл за каждый год педагогического стажа. Поскольку в результате будет возвращено довольно большое количество строк, я включу метод `head()`, чтобы получить только несколько из них. Такая практика добавления нескольких методов к одной и той же команде называется *цепочкой методов*:

```
In [17]: star.groupby('totexpk')[['ttl_score']].max().head()

Out[17]:
              ttl_score
       totexpk
       0           1171
       1           1133
       2           1091
       3           1203
       4           1229
```

В *главе 8* рассматривались сходства и различия между функцией Excel `VLOOKUP()` и левым внешним соединением. Я считаю свежую копию набора **star**, как и **districts**. Для соединения этих наборов данных используем `pandas`, а чтобы «отыскать» данные из **school-districts** в **star** — метод `merge()`. Установив для аргумента `how` (как)

значение `left` (левый), мы зададим левое внешнее соединение — тип соединения, наиболее похожий на функцию `VLOOKUP()`:

```
In [18]: star = pd.read_excel('datasets/star/star.xlsx')
         districts = pd.read_csv('datasets/star/districts.csv')
         star.merge(districts, how='left').head()

Out[18]:
   tmathssk  treadssk            classk  totexpk   sex freelunk   race  \
0       473       447       small.class        7  girl       no  white
1       536       450       small.class       21  girl       no  black
2       463       439  regular.with.aide        0   boy      yes  black
3       559       448           regular       16   boy       no  white
4       489       447       small.class        5   boy      yes  white

   schidkn    school_name          county
0       63     Ridgeville     New Liberty
1       20  South Heights         Selmont
2       19      Bunnlevel         Sattley
3       69          Hokah      Gallipolis
4       79   Lake Mathews  Sugar Mountain
```

Python, как и R, довольно интуитивен в вопросе соединения данных: он по умолчанию «знал», что надо объединять по **schidkn** и извлек и **school_name**, и **county**.

Преобразование данных

Рассмотрим способы увеличения ширины и длины набора данных в Python, опять же, с использованием `pandas`. Для начала можно использовать функцию `melt()` для объединения **tmathssk** и **treadssk** в один столбец. Для этого я укажу кадр данных с помощью аргумента `frame`; переменную, используемую в качестве уникального идентификатора, — с помощью `id_vars`; и переменные, которые следует объединить в один столбец, — с помощью `value_vars`. Также я укажу, как нужно назвать результирующее значение и пометить переменные с помощью `value_name` и `var_name` соответственно:

```
In [19]: star_pivot = pd.melt(frame=star, id_vars = 'schidkn',
             value_vars=['tmathssk', 'treadssk'], value_name='score',
             var_name='test_type')
         star_pivot.head()

Out[19]:
           schidkn test_type  score
         0       63  tmathssk    473
         1       20  tmathssk    536
         2       19  tmathssk    463
         3       69  tmathssk    559
         4       79  tmathssk    489
```

Что вы думаете о переименовании **tmathssk** и **treadssk** в **math** (математика) и **reading** (чтение) соответственно? Воспользуемся словарем Python, чтобы настроить объект `mapping`, который служит чем-то вроде «справочной таблицы» для перекодировки значений. Передадим данные в метод `map()`, который перекодирует `test_type`. Также я применю метод `unique()` для подтверждения того, что только **math** и **reading** найдены теперь в **test_type**:

```
In [20]: # Переименовать записи в `test_type`
         mapping = {'tmathssk':'math','treadssk':'reading'}
         star_pivot['test_type'] = star_pivot['test_type'].map(mapping)

         # Найти уникальные значения в test_type
         star_pivot['test_type'].unique()

Out[20]: array(['math', 'reading'], dtype=object)
```

Чтобы расширить обратно **star_pivot** до отдельных столбцов **math** и **reading**, используем метод `pivot_table()`. Сначала с помощью аргумента index укажем, по какой переменной выполнять индексирование, затем — какие переменные содержат метки и значения с помощью аргументов `columns` и `values` соответственно.

В `pandas` можно настроить уникальные индексные столбцы; по умолчанию `pivot_table()` установит любые переменные, которые вы включите в аргумент `index`. Изменить это можно с помощью метода `reset_index()`. Чтобы узнать больше об индексировании в `pandas`, а также об огромном количестве других способов манипулирования данными и методов анализа, которые мы не можем рассмотреть здесь, обратитесь к книге Уэса Маккини «Python и анализ данных»[1] (Wes McKinney. Python for Data Analysis).

```
In [21]: star_pivot.pivot_table(index='schidkn',
          columns='test_type', values='score').reset_index()

Out[21]:
         test_type  schidkn        math     reading
         0                1  492.272727  443.848485
         1                2  450.576923  407.153846
         2                3  491.452632  441.000000
         3                4  467.689655  421.620690
         4                5  460.084746  427.593220
         ..             ...         ...         ...
         74              75  504.329268  440.036585
         75              76  490.260417  431.666667
         76              78  468.457627  417.983051
         77              79  490.500000  434.451613
         78              80  490.037037  442.537037

         [79 rows x 3 columns]
```

[1] Издана в России. — *Ред.*

Визуализация данных

Кратко рассмотрим визуализацию данных в Python, в частности с использованием пакета `seaborn`. Поскольку пакет `seaborn` отлично подходит для статистического анализа и работы с кадрами данных `pandas`, в данном случае это удачный выбор. Так же как `pandas` надстроен над `numpy`, пакет `seaborn` использует функции другого популярного пакета Python для построения графиков — `matplotlib`.

Пакет `seaborn` включает в себя множество функций для построения графиков различных типов. Мы изменим аргументы в этих функциях, чтобы указать, какой набор данных отображать графически, какие переменные располагаются по осям x и y, какие цвета использовать и т. д. Начнем с визуализации количества наблюдений для каждого уровня **classk** с помощью функции `countplot()`.

Наш набор данных — **star**, что мы и укажем с помощью аргумента `data`. Чтобы расположить уровни **classk** по оси x, используем аргумент `x`. В результате получим «график подсчета», или «каунтплот» (countplot), показанный на рис. 12.2.

```
In [22]: sns.countplot(x='classk', data=star)
```

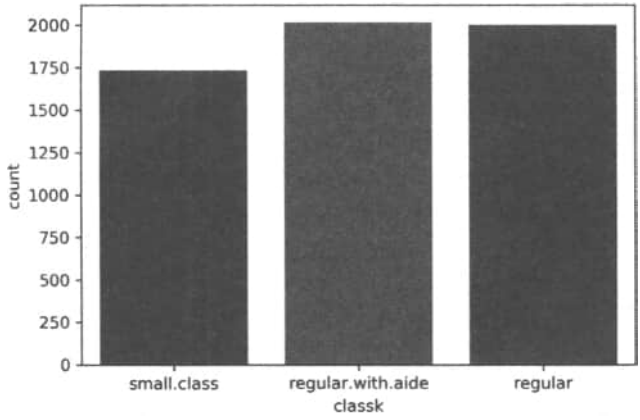


Рис. 12.2. График подсчета

Далее, для получения гистограммы **treadssk** используем функцию `displot()`. Напомню, мы укажем `x` и `data`. Результат показан на рис. 12.3.

```
In [23]: sns.displot(x='treadssk', data=star)
```

Функции пакета `seaborn` содержат множество дополнительных аргументов для настройки внешнего вида графика (гистограммы). В частности, можно изменить количество столбиков гистограммы, например, до 25, и цвет — например, на розовый (pink). Результат показан на рис. 12.4.

```
In [24]: sns.displot(x='treadssk', data=star, bins=25, color='pink')
```

Для создания блочной диаграммы, или «боксплота», предназначена функция `boxplot()`; пример на рис. 12.5.

```
In [25]: sns.boxplot(x='treadssk', data=star)
```

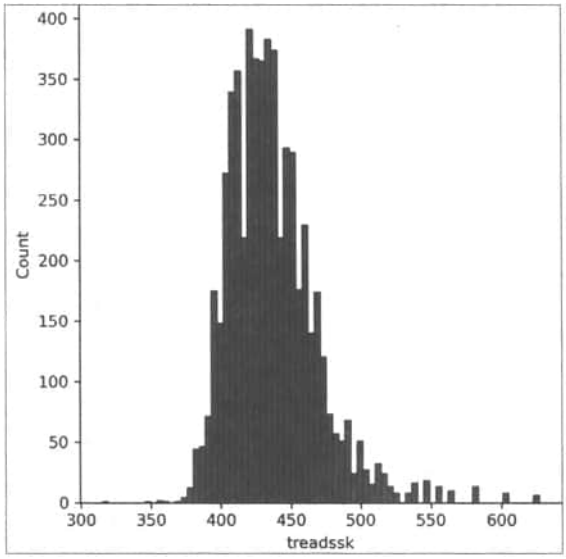


Рис. 12.3. Гистограмма

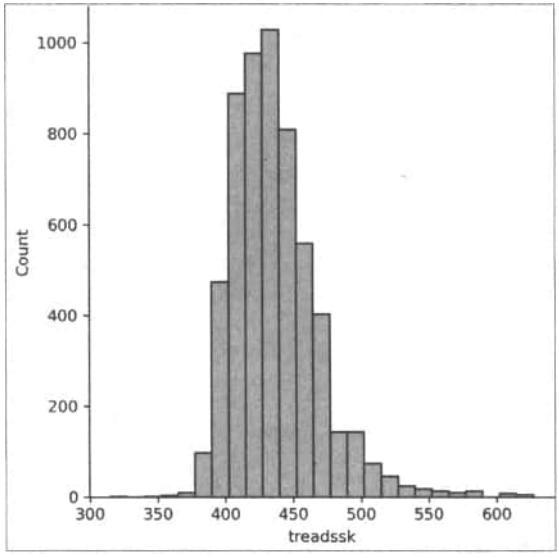


Рис. 12.4. Гистограмма с пользовательскими настройками

В любом из этих случаев можно «перевернуть» график, включив интересующую нас переменную в аргумент `y`. Давайте сделаем это с блочной диаграммой, в результате чего получим то, что показано на рис. 12.6 как выходные данные.

```
In [26]: sns.boxplot(y='treadssk', data=star)
```

Чтобы построить блочную диаграмму для каждого уровня размера класса, добавим дополнительный аргумент в диаграмму **classk** по оси x, в результате чего получим блочную диаграмму по группам, как на рис. 12.7.

```
In [27]: sns.boxplot(x='classk', y='treadssk', data=star)
```

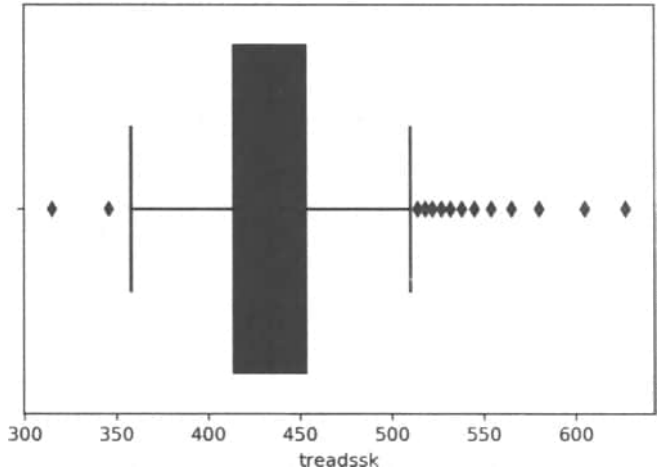


Рис. 12.5. Блочная диаграмма

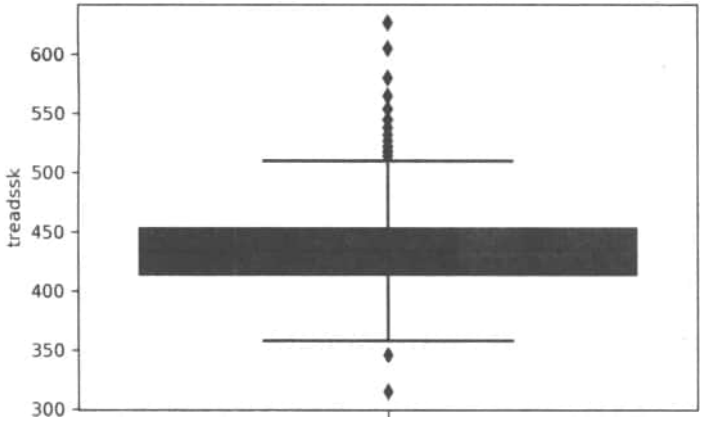


Рис. 12.6. «Перевернутая» блочная диаграмма

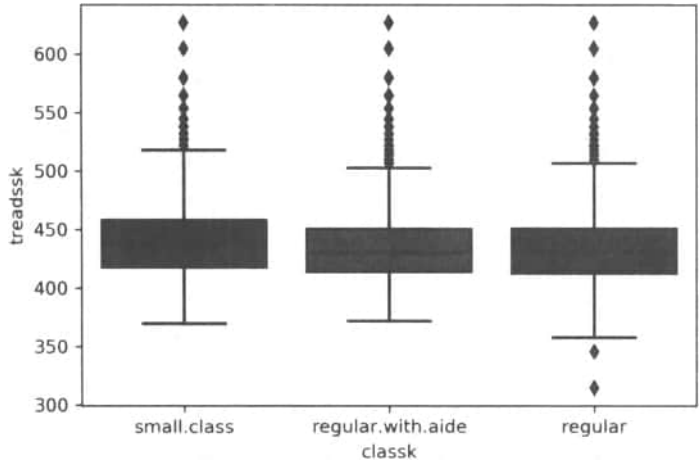


Рис. 12.7. Блочная диаграмма по группам

Теперь давайте используем функцию `scatterplot()`, чтобы построить график взаимосвязи **tmathssk** по оси x и **treadssk** по оси y. Результат показан на рис. 12.8.

```
In [28]: sns.scatterplot(x='tmathssk', y='treadssk', data=star)
```

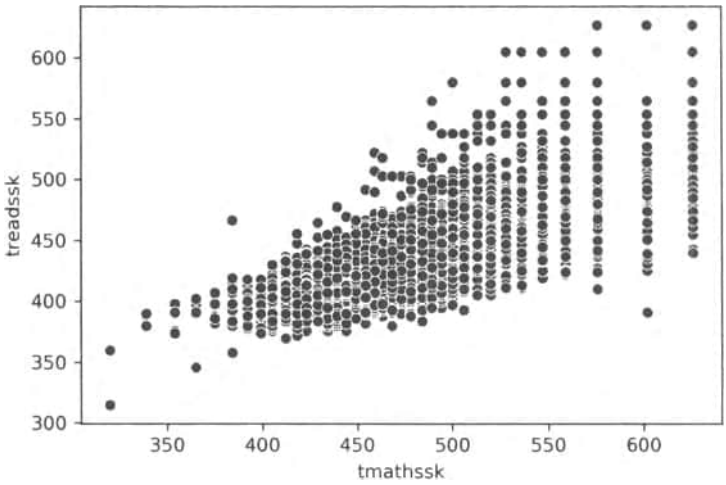


Рис. 12.8. Диаграмма рассеяния

Предположим, мы хотим поделиться этой диаграммой с внешней аудиторией, которая может не знать, что такое **treadssk** и **tmathssk**. Можно добавить к нашей диаграмме дополнительные метки, воспользовавшись нужными функциями в `matplotlib.pyplot`. Запустим ту же функцию `scatterplot()`, как и раньше, но на этот раз также вызовем функции из `pyplot` для добавления пользовательских меток по осям x и y, а также создания заголовка диаграммы. Результат представлен на рис. 12.9.

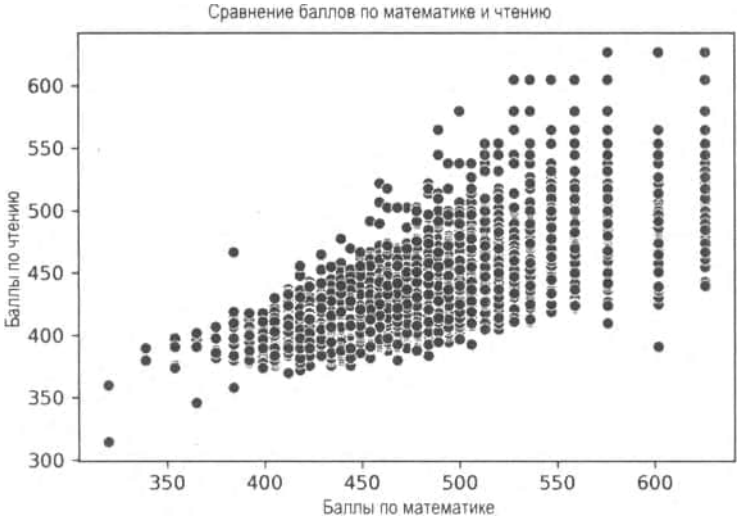


Рис. 12.9. Диаграмма рассеяния с настроенными метками осей и заголовком

```
In [29]: sns.scatterplot(x='tmathssk', y='treadssk', data=star)
         plt.xlabel('Math score')
         plt.ylabel('Reading score')
         plt.title('Math score versus reading score')
```

Пакет `seaborn` содержит значительно больше возможностей для построения внешне привлекательных визуализаций данных. Для получения подробной информации ознакомьтесь с официальной документацией.

Заключение

Пакеты `pandas` и `seaborn` обладают значительно бóльшими возможностями, чем мы здесь рассмотрели, но и этого достаточно, чтобы вы могли приступить к выполнению реальной задачи — изучению и тестированию взаимосвязей данных, чему посвящена *глава 13*.

Упражнения

В репозитории к книге имеются два файла, расположенные в подпапке **census** (численность населения) папки **datasets**: **census.csv** и **census-divisions.csv**. Считайте их в Python и выполните следующие задания.

1. Отсортируйте данные по регионам (по возрастанию), по подразделениям (по возрастанию) и по численности населения (по убыванию) (для выполнения этого вам потребуется объединить наборы данных). Запишите результаты в рабочий лист Excel.
2. Исключите поле почтового кода из объединенного набора данных.
3. Создайте новый столбец **density** (плотность населения), содержащий расчет численности населения, деленной на площадь суши.
4. Визуализируйте зависимость между площадью суши и численностью населения для всех наблюдений в 2015 году.
5. Найдите общую численность населения для каждого региона в 2015 году.
6. Создайте таблицу, содержащую названия штатов и данные о численности населения за каждый год в промежутке с 2010 по 2015, расположенные в отдельном столбце.

ГЛАВА 13

Кульминация: Python для анализа данных

В конце *главы 8* вы расширили свои знания в R, чтобы исследовать и протестировать зависимости в наборе данных **mpg**. Здесь мы займемся тем же, но с использованием Python. Поскольку аналогичная работа уже проделана нами в Excel и R, я буду меньше внимания уделять теоретическому обоснованию нашего анализа и больше — практической стороне и выполнению задач в Python.

Для начала вызовем все необходимые модули. Некоторые из них для вас новые: мы импортируем подмодуль `stats` из библиотеки (модуля) `scipy`. С этой целью используем ключевое слово `from`, чтобы сообщить Python, какой модуль искать, затем — обычное ключевое слово `import`, чтобы выбрать подмодуль. Как очевидно из названия, мы используем подмодуль `stats` (статистика) из библиотеки (модуля) `scipy` для статистического анализа. Затем с помощью нового пакета под названием `sklearn`, или `scikit-learn`, проверим правильность разделения нашей модели на тестирующую и обучающую части. Этот пакет стал основным ресурсом для машинного обучения и устанавливается вместе с Anaconda.

```
In [1]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from scipy import stats
        from sklearn import linear_model
        from sklearn import model_selection
        from sklearn import metrics
```

С помощью аргумента `usecols` функции `read_csv()` мы можем указать, какие столбцы считать в кадр данных:

```
In [2]: mpg = pd.read_csv('datasets/mpg/mpg.csv',usecols=
            ['mpg','weight','horsepower','origin','cylinders'])
        mpg.head()
```

```
Out[2]:
    mpg  cylinders  horsepower  weight origin
 0  18.0          8         130    3504   USA
 1  15.0          8         165    3693   USA
 2  18.0          8         150    3436   USA
 3  16.0          8         150    3433   USA
 4  17.0          8         140    3449   USA
```

Разведочный анализ данных

Начнем с описательной статистики:

```
In[3]: mpg.describe()

Out[3]:
              mpg   cylinders  horsepower       weight

count  392.000000  392.000000  392.000000   392.000000
mean    23.445918    5.471939  104.469388  2977.584184
std      7.805007    1.705783   38.491160   849.402560
min      9.000000    3.000000   46.000000  1613.000000
25%     17.000000    4.000000   75.000000  2225.250000
50%     22.750000    4.000000   93.500000  2803.500000
75%     29.000000    8.000000  126.000000  3614.750000
max     46.600000    8.000000  230.000000  5140.000000
```

Поскольку переменная **origin** (регион происхождения) — категориальная, по умолчанию она не отображается как часть функции `describe()`. Исследуем эту переменную с помощью таблицы частот. Это можно сделать в `pandas` с помощью функции `crosstab()`. Прежде всего укажем, какие данные поместить в индекс: **origin**. Таким образом, мы получим итоговое количество для каждого уровня, установив аргумент `columns` в значение `count` (рассчитать):

```
In [4]: pd.crosstab(index=mpg['origin'], columns='count')

Out[4]:
col_0   count
origin
Asia       79
Europe     68
USA       245
```

Чтобы создать двумерную таблицу частот, мы можем приравнять `columns` к другой категориальной переменной, такой как `cylinders`:

```
In [5]: pd.crosstab(index=mpg['origin'], columns=mpg['cylinders'])

Out[5]:
cylinders  3   4  5   6    8
origin
Asia       4  69  0   6    0
Europe     0  61  3   4    0
USA        0  69  0  73  103
```

Далее извлечем описательную статистику из **mpg** (пробег) по каждому уровню **origin** (регион происхождения). Я сделаю это посредством соединения двух методов и последующего разложения результатов:

```
In[6]: mpg.groupby('origin').describe()['mpg']

Out[6]:
        count       mean       std   min    25%   50%     75%   max
origin
Asia     79.0  30.450633  6.090048  18.0  25.70  31.6  34.050  46.6
Europe   68.0  27.602941  6.580182  16.2  23.75  26.0  30.125  44.3
USA     245.0  20.033469  6.440384   9.0  15.00  18.5  24.000  39.0
```

Мы можем визуализировать общее распределение **mpg**, как показано на рис. 13.1:

```
In[7]: sns.displot(data=mpg, x='mpg')
```

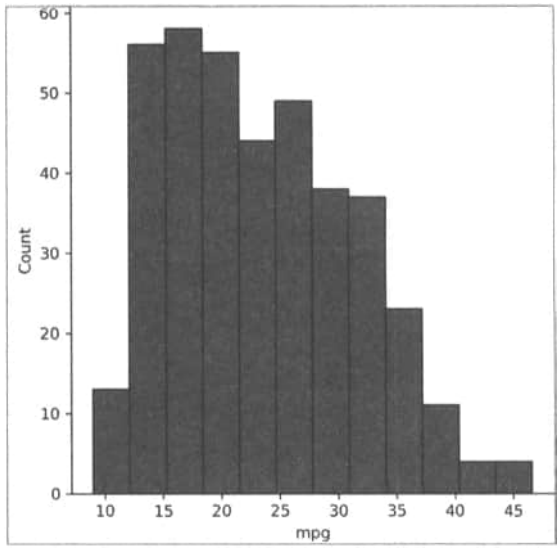


Рис. 13.1. Гистограмма **mpg**

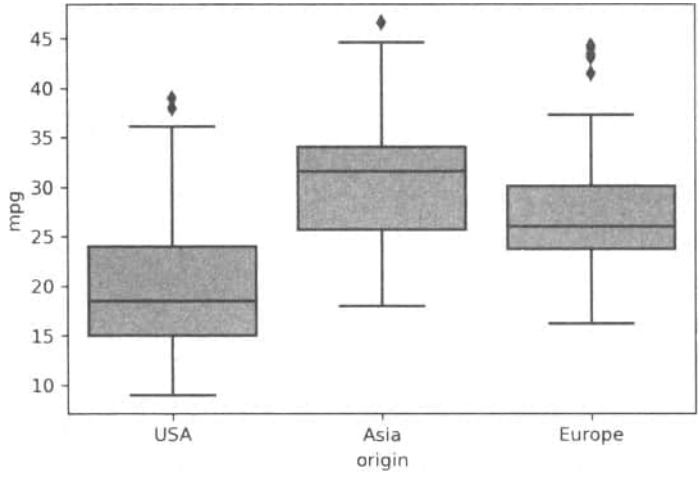


Рис. 13.2. Блочная диаграмма распределения **mpg** по **origin**

Теперь давайте построим блочную диаграмму («боксплот»), как показано на рис. 13.2, сравнивая распределение **mpg** (пробег) по каждому уровню **origin** (регион происхождения):

```
In[8]: sns.boxplot(x='origin', y='mpg', data=mpg, color='pink')
```

Мы можем установить аргумент `col` функции `displot()` в значение `origin` для создания фасетных гистограмм, таких как на рис. 13.3:

```
In[9]: sns.displot(data=mpg, x="mpg", col="origin")
```

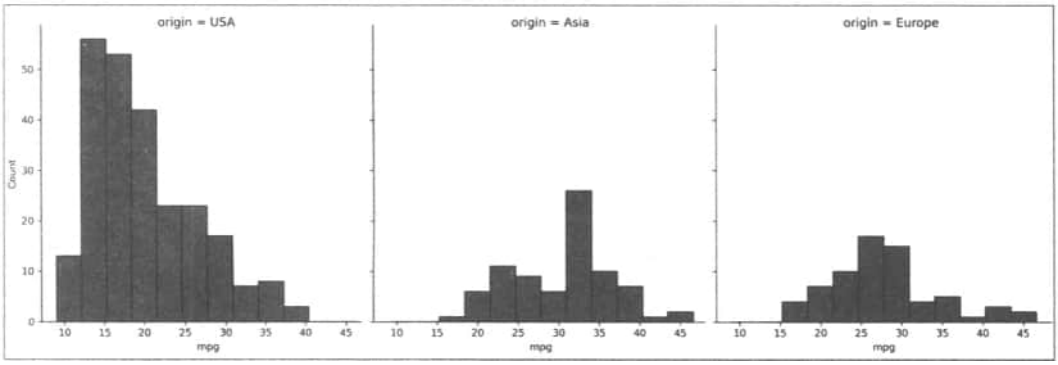


Рис. 13.3. Фасетная гистограмма **mpg** по **origin**

Проверка гипотез

Давайте еще раз протестируем разницу в пробеге между американскими и европейскими автомобилями. Для упрощения анализа разобьем наблюдения в каждой группе на кадры данных.

```
In[10]: usa_cars = mpg[mpg['origin']=='USA']
        europe_cars = mpg[mpg['origin']=='Europe']
```

t-тест для независимых выборок

Для выполнения t-теста используем функцию `ttest_ind()` из библиотеки `scipy.stats`. Эта функция ожидает два массива `numpy` в качестве аргументов; также подойдут серии (Series) из `pandas`:

```
In[11]: stats.ttest_ind(usa_cars['mpg'], europe_cars['mpg'])

Out[11]: Ttest_indResult(statistic=-8.534455914399228,
             pvalue=6.306531719750568e-16)
```

К сожалению, вывод здесь довольно скудный: хотя он включает p-значение, в нем отсутствует доверительный интервал. Чтобы запустить t-тест с бóльшим количеством выходных данных, воспользуйтесь модулем `researchpy`.

Перейдем к анализу непрерывных переменных. Начнем с матрицы корреляции. Мы можем использовать метод `corr()` из пакета `pandas`, включив только релевантные переменные:

```
In[12]: mpg[['mpg','horsepower','weight']].corr()

Out[12]:
                mpg  horsepower    weight
mpg        1.000000   -0.778427 -0.832244
horsepower -0.778427    1.000000  0.864538
weight     -0.832244    0.864538  1.000000
```

Визуализируем зависимость между переменными **weight** (вес) и **mpg** (пробег) диаграммой рассеяния, как показано на рис. 13.4:

```
In[13]: sns.scatterplot(x='weight', y='mpg', data=mpg)
        plt.title('Relationship between weight and mileage')
```

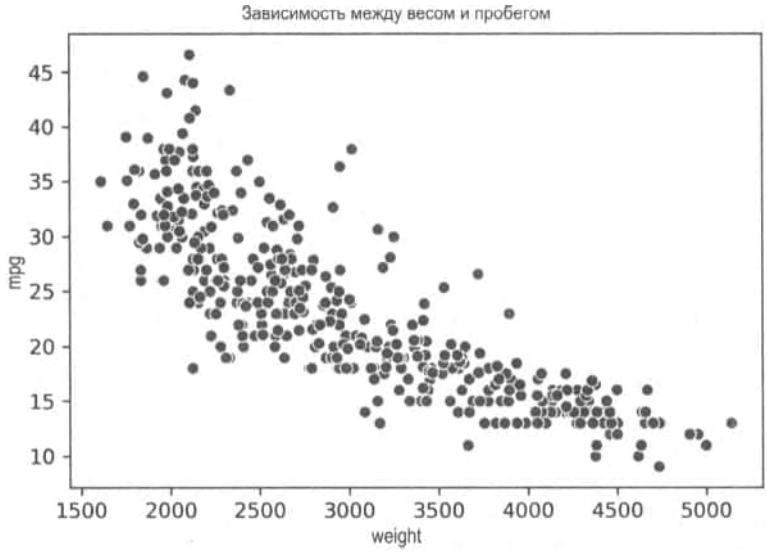


Рис. 13.4. Диаграмма рассеяния **mpg** по **weight**

Также можно построить диаграммы рассеяния по всем парам набора данных с помощью функции `pairplot()` из пакета `seaborn`. Гистограммы каждой переменной расположены по диагонали (рис. 13.5).

```
In[14]: sns.pairplot(mpg[['mpg','horsepower','weight']])
```

Линейная регрессия

Пришло время для линейной регрессии. Для этого мы используем функцию `linregress()` из библиотеки `scipy`, которая также ожидает два массива `numpy`, или серии (Series) из `pandas`. Зададим независимую и зависимую переменные с помощью аргументов `x` и `y` соответственно:

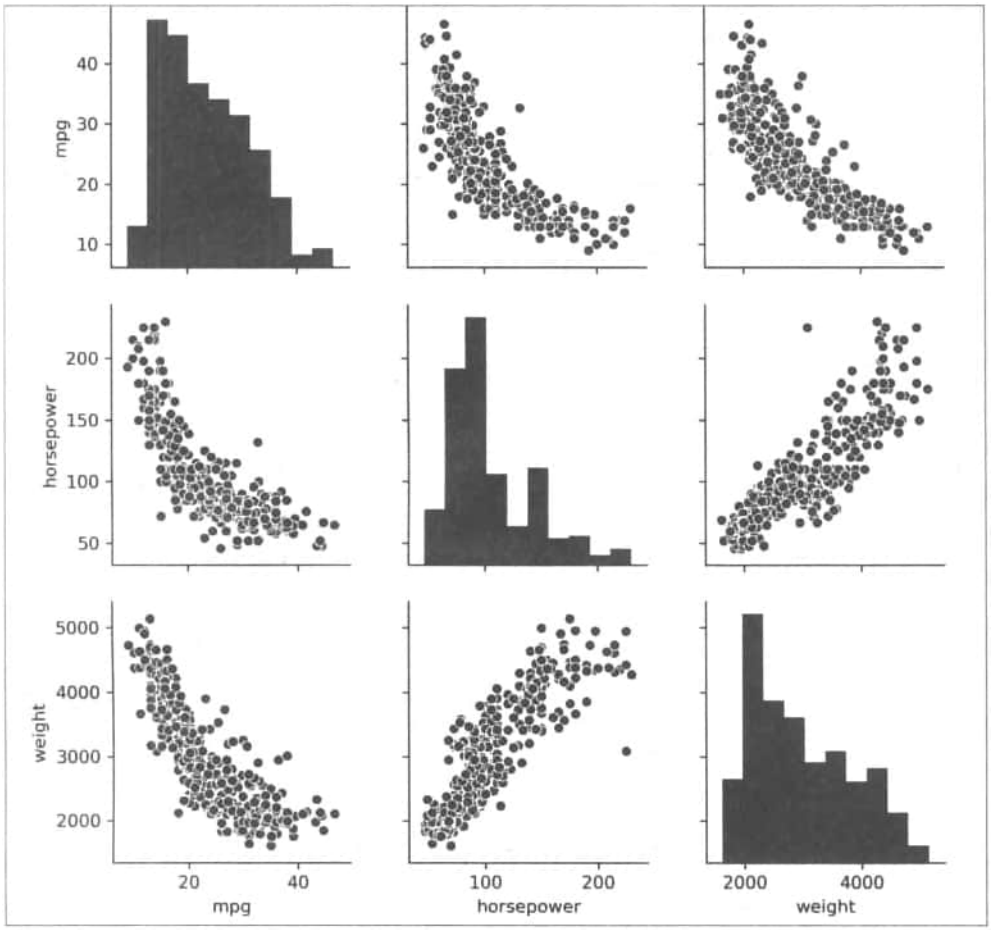


Рис. 13.5. Парный график

```
In[15]: # Линейная регрессия веса по пробегу
        stats.linregress(x=mpg['weight'], y=mpg['mpg'])

Out[15]: LinregressResult(slope=-0.007647342535779578,
   intercept=46.21652454901758, rvalue=-0.8322442148315754,
   pvalue=6.015296051435726e-102, stderr=0.0002579632782734318)
```

Вы опять увидите, что некоторые выходные данные, к которым вы, возможно, привыкли, здесь отсутствуют. *Будьте осторожны*: включенное `rvalue` (r-значение) является *коэффициентом корреляции*, а не коэффициентом детерминации (R-квадрат). Чтобы получить больше выходных данных регрессии, воспользуйтесь модулем `statsmodels`.

И последнее, хотя и не менее важное: наложим линию регрессии на диаграмму рассеяния. Библиотека `seaborn` включает в себя отдельную функцию для выполнения именно этой задачи — `regplot()`. Как обычно, зададим независимые и зависимые переменные, а также укажем, где взять данные. Результат представлен на рис. 13.6.

```
In[16]: # Наложить линию регрессии на диаграмму рассеяния
        sns.regplot(x="weight", y="mpg", data=mpg)
        plt.xlabel('Weight (lbs)')
        plt.ylabel('Mileage (mpg)')
        plt.title('Relationship between weight and mileage')
```

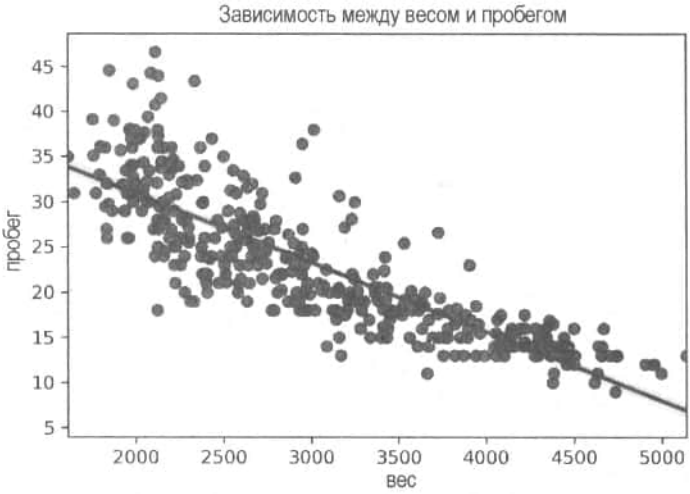


Рис. 13.6. Диаграмма рассеяния с линией регрессии

Разделение и проверка данных для обучения и тестирования

В конце *главы 9* вы узнали, как разделять данные для обучения и тестирования при построении линейной регрессионной модели в R.

Сейчас мы будем использовать функцию `train_test_split()` для разделения нашего набора данных на *четыре* кадра данных: не только по обучению и тестированию, но и по независимым и зависимым переменным. Сначала передадим кадр данных, содержащий независимую переменную, затем — содержащий зависимую переменную. Используя аргумент `random_state`, запустим генератор случайных чисел так, чтобы результаты оставались согласованными для следующего примера:

```
In[17]: X_train, X_test, y_train, y_test =
        model_selection.train_test_split(mpg[['weight']], mpg[['mpg']],
        random_state=1234)
```

По умолчанию данные разделены на части для обучения и тестирования в соотношении 75/25:

```
In[18]: y_train.shape

Out[18]: (294, 1)
```

[1] Зависимость между весом и пробегом. — *Ред.*

```
In[19]: y_test.shape

Out[19]: (98, 1)
```

Теперь совместим модель с данными для обучения. Сначала зададим линейную модель с помощью функции `LinearRegression()`, затем обучим эту модель, используя функцию `regr.fit()`. Чтобы получить предсказанные значения для тестового набора данных, можно использовать функцию `predict()`. В результате мы получим массив `numpy`, а не кадр данных `pandas`, поэтому метод `head()` не может использоваться для печати нескольких первых строк. Тем не менее мы можем «нарезать» его:

```
In[20]: # Создать объект линейной регрессии
        regr = linear_model.LinearRegression()

        # Обучить модель с помощью наборов для обучения
        regr.fit(X_train, y_train)

        # Сделать прогнозы, используя набор для обучения
        y_pred = regr.predict(X_test)

        # Распечатать 5 первых наблюдений
        y_pred[:5]

Out[20]: array([[14.86634263], [23.48793632],
        [26.2781699 ],
        [27.69989655],
        [29.05319785]])
```

Атрибут `coef_` возвращает коэффициент нашей модели обучения:

```
In[21]: regr.coef_

Out[21]: array([[-0.00760282]])
```

Чтобы получить больше информации о модели, например p-значение или R-квадрат, попробуйте выполнить совмещение модели с помощью пакета `statsmodels`.

Сейчас мы оценим производительность нашей модели на данных для обучения, на этот раз используя подмодуль `metrics` из пакета `sklearn`. Передадим фактические и прогнозные значения в функции `r2_score()` и `mean_squared_error()`, в результате чего получим R-квадрат и среднеквадратичную ошибку (RMSE) соответственно:

```
In[22]: metrics.r2_score(y_test, y_pred)

Out[22]: 0.6811923996681357

In[23]: metrics.mean_squared_error(y_test, y_pred)

Out[23]: 21.63348076436662
```

Заключение

Мое обычное замечание о том, что мы только *коснулись* возможностей анализа этого или любого другого набора данных, относится и к этой главе. Но я надеюсь, вы чувствуете, что достигли некоторых успехов при работе с данными в Python.

Упражнения

Еще раз взгляните на набор данных **ais**, на этот раз с Python. Считайте рабочую книгу Excel из репозитория на GitHub и выполните следующие задания. Вы уже должны чувствовать себя достаточно уверенно в проведении такого анализа.

1. Визуализируйте распределение количества эритроцитов (**rcc**) в зависимости от пола (**sex**).
2. Ответьте на вопрос: существует ли значимое различие в количестве эритроцитов между двумя группами разного пола?
3. Создайте матрицу корреляции значимых переменных в этом наборе данных.
4. Визуализируйте зависимость между ростом (**ht**) и весом (**wt**).
5. Постройте регрессию **ht** по **wt**. Найдите уравнение подходящей линии регрессии. Имеется ли какая-то существенная зависимость?
6. Разделите регрессионную модель на подмножества для обучения и тестирования. Какова величина R-квадрата и среднеквадратичной ошибки ((RMSE) вашей модели обучения?

ГЛАВА 14

Заключение и дальнейшие шаги

В предисловии к книге я поставил следующую цель обучения.

> К концу книги вы должны научиться *выполнять разведочный анализ данных и проверку гипотез, используя язык программирования*.

Искренне надеюсь, что вы чувствуете себя достигшими этой цели, а также уверенными в дальнейшем успешном продвижении к другим уровня анализа. Чтобы завершить данный этап нашего аналитического путешествия, я хотел бы поделиться с вами некоторыми темами, которые помогут расширить полученные знания.

Дополнительные элементы стека анализа данных

В *главе 5* перечислены четыре основные категории прикладных программ, используемых в анализе данных: электронные таблицы, языки программирования, базы данных и инструменты бизнес-аналитики. Поскольку мы сосредоточились на статистических элементах аналитики, то уделяли особое внимание первым двум элементам стека анализа данных. Перечитайте эту главу, чтобы вспомнить, как остальные элементы стека связаны с уже изученными и что́ нужно о них знать.

План исследований и бизнес-эксперименты

В *главе 3* мы говорили о том, что достоверный анализ данных возможен только из достоверно собранных *исходных данных*: как говорится, «мусор на входе, мусор на выходе» («garbage in, garbage out», GIGO). В этой книге мы предполагали, что данные были тщательно собраны и подходили для анализа, а также содержали репрезентативную выборку. Мы работали с хорошо известными наборами данных, зачастую взятыми из авторитетных исследований, так что это надежные источники.

Но иногда вы не можете быть настолько уверены в имеющихся данных; вы можете нести ответственность *и* за их сбор, *и* за анализ. Поэтому следует узнать больше о *плане* и *методах* исследований. Эта область может оказаться довольно сложной и академичной, но она нашла практическое применение в сфере бизнес-экспериментов. Обратитесь к книге Стефана Х. Томке «Экспериментирование работает:

удивительная сила бизнес-экспериментов» (Stefan H. Thomke. Experimentation Works: The Surprising Power of Business Experiments) для получения информации о том, как и почему следует применять исследовательские методы в бизнесе.

Дополнительные статистические методы

Как упоминалось в *главе 4*, мы только коснулись типов доступных статистических критериев, хотя многие из них основаны на принципах проверки гипотез, рассмотренных в *главе 3*.

Чтобы получить общие сведения о прочих статистических методиках, обратитесь к книге Сары Босло «Статистика в двух словах» (Sarah Boslaugh. Statistics in a Nutshell). Затем перейдите к книге Питера Брюса и соавт. «Практическая статистика для специалистов Data Science (Peter Bruce et al. Practical Statistics for Data Scientists), чтобы научиться применять полученные знания, используя R и Python. Как видно из названия, последняя книга находится на грани между статистикой и наукой о данных.

Наука о данных и машинное обучение

В *главе 5* вкратце рассмотрены различия между статистикой, аналитикой и наукой о данных. Но несмотря на различия в методах, в этих областях больше сходств, чем различий.

Если вы действительно интересуетесь наукой о данных и машинным обучением, сосредоточьтесь на изучении R и Python, добавив к этому некоторые элементы SQL и баз данных. Чтобы понять, как язык R используется в науке о данных, обратитесь к книге Хэдли Уикхема и Гаррета Гролемунда «R для науки о данных» (Hadley Wickham and Garrett Grolemund. R for Data Science). Изучить Python вам поможет книга Орельена Жерона «Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow» (Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow).

Контроль версий

В *главе 5* также говорится о важности воспроизводимости. Давайте посмотрим на главное направление борьбы с этой проблемой. Вероятно, вы уже сталкивались с набором файлов типа приведенных ниже:

- *proposal*.txt;
- *proposal-v2*.txt;
- *proposal-Feb23*.txt;
- *proposal-final*.txt;
- *proposal-FINAL-final*.txt.

Возможно, один пользователь создал файл **proposal-v2.txt**, а другой — **proposal-Feb23.txt**. Кроме того, существует разница между файлами **proposal-final.txt** и **proposal-FINAL-final.txt**, и с этим тоже надо справляться. Довольно трудно разобраться, какая копия является «главной» и как преобразовать и перенести все изменения в эту копию, сохраняя информацию о том, кто и что изменил.

На помощь может прийти *система контроля версий*, предназначенная для отслеживания проектов по времени, например дополнений и изменений, внесенных разными пользователями. Контроль версий значительно меняет правила игры для совместной работы и отслеживания изменений, но требует внимательного изучения.

Git — доминирующая система контроля версий, довольно популярная среди аналитиков, разработчиков программного обеспечения и других технических специалистов. В частности, они часто используют облачный хостинг GitHub для управления проектами Git. Чтобы получить представление о Git и GitHub, обратитесь к книге Джона Лолигера и Мэтью Маккалоу «Контроль версий с Git» (Version Control with Git). Чтобы узнать, как соединить Git и GitHub с R и RStudio, ознакомьтесь с онлайн-ресурсом Дженни Брайан и др. «Счастливый Git и GitHub для пользователя R» (Jenny Bryan et al. Happy Git and GitHub for the useR). В настоящее время Git и другие системы контроля версий относительно нетривиальны в процессе аналитики данных, но их популярность растет, отчасти в связи с возросшей потребностью в воспроизводимости.

Этика

От записи и сбора до анализа и моделирования данные окружены этическими проблемами. В *главе 3* вы узнали о статистической предвзятости (особенно в контексте машинного обучения): модель может начать ущемлять интересы групп людей несправедливым или незаконным способом. Если собираются данные об отдельных лицах, необходимо учитывать конфиденциальность и согласие этих лиц.

Этика не всегда имела приоритет в аналитике и науке о данных. К счастью, ситуация начинает меняться, и эта тенденция может продолжаться только при всесторонней поддержке общества. Краткое руководство по внедрению этических стандартов в работу с данными можно найти в книге Майка Лоукидеса и соавт. «Этика и наука о данных» (Mike Loukides et al. Ethics and Data Science).

Двигайтесь вперед и выбирайте то, что нравится

Меня часто спрашивают, на каких из инструментов следует сосредоточиться с учетом требований работодателя и роста их популярности. Мой ответ: найдите время, чтобы понять свои предпочтения и позвольте этим интересам формировать ваш план обучения, вместо того чтобы пытаться подстроиться под «следующий большой бум» в аналитических инструментах. *Все* эти навыки ценны. Умение согласовывать и сочетать инструменты аналитики важнее любого из них по отдельности,

что требует доступа к широкому спектру инструментов. Но вы не можете стать экспертом во всем. Лучшая стратегия обучения должна напоминать букву «Т»: знакомство с широким спектром различных инструментов работы с данными и относительно глубокое владение некоторыми из них.

Напутствие

Найдите минутку, чтобы взглянуть на всё, чего вы достигли с помощью этой книги, — вы должны испытывать гордость. Но не стоит медлить, ведь нужно еще так много изучить, и потребуется не так уж много времени, чтобы понять: в этой книге вы только коснулись верхушки айсберга. Вот вам упражнение для конца главы и конца книги: не сидите сложа руки, продолжайте учиться и развиваться в аналитике.

Предметный указатель

A

ANOVA (дисперсионный анализ) 83
arrange(), функция 135
array(), функция 181

B

bind_cols(), функция 162
boxplot(), функция 199

C

chdir() 185
cor(), функция 158
corr(), метод 208
count(), функция 153
countplot() 199
CRAN (Комплексная R-архивная сеть) 103
crosstab(), функция 205
CRUD, операции выполняемые SQL 97

D

data.frame(), функция 120
DataFrame, функция 184
desc(), функция 136
describe(), метод 187
◊ функция 153
displot() 207, функция 199
drop(), метод 192

F

facet_wrap(), функция 155
filter(), функция 136

G

geom_boxplot(), функция 146
geom_histogram(), функция 146
geom_point(), функция 149
geom_smooth(), функция 160
get_dataset_names(), функция 185
getcwd() 185
ggplot(), функция 145
glance(), функция 162
group_by(), функция 138
groupby(), метод 195

H

head(), метод 185

I

iloc, метод 188
info(), метод 187
initial_split(), функция 161
isfile() 186

J

Jupyter Notebook 168

L

left_join(), функция 140
linear_reg(), функция 162
LinearRegression(), функция 211
linregress(), функция 208
list, коллекция объектов 180
lm(), функция 159
load_data set() 185
loc, метод 189

M

map(), метод 198
Markdown 171
melt(), функция 197
merge(), метод 196
mutate(), функция 134

N

NA 140
NaN 188
ndarray, n-мерный массив 181
NORM.DIST(), функция 49
numpy
◊ массив 181
◊ модуль 181

P

pairplot(), функция 208
pairs(), функция 158
pandas 184
PEP, предложения по улучшению Python 175
pip, система управления пакетами Python 177
pipe, оператор 141
pivot_table(), метод 198
pivot_wider(), функция 144
Power Pivot 95
Power Query 95
Power View 95
predict(), функция 162, 211
PyPI, каталог пакетов Python 177
Python 167
p-значение (p-value) 64, 65

R

read_csv(), функция 186
read_excel(), функция 186
recode(), функция 144
regplot(), функция 209
regr.fit(), функция 211
rename(), функция 134
reset_index(), метод 198
RMSE, среднеквадратичная ошибка 163
rmse(), функция 163
rsq(), функция 163
R-squared (R-квадрат), коэффициент детерминации 86
RStudio 103
R-квадрат (R-squared), коэффициент детерминации 86

S

scatterplot() 202
select(), функция 132
sort_values(), метод 194
SQL, язык структурированных запросов 97
sum(), функция 138
summarize(), функция 138

T

t.test(), функция 157
testing(), функция 161
tidy(), функция 162
tidyverse 123
train_test_split(), функция 210
training(), функция 161
ttest_ind(), функция 207
type(), функция 175
t-распределение (распределение Стьюдента) 66
t-статистика (t-критерий) 84

U

unique(), метод 198
upper(), метод 176

V

VBA (Visual Basic for Applications), язык программирования Excel 94

W

write_csv(), метод 189
write_xlsx(), метод 189

А

Альфа (число) 61
Аналитика данных 90

Б

Безусловная (маргинальная) вероятность 43
Бизнес-аналитика 91
Блокноты Jupyter 168

В

Векторы в R 118
Визуализация данных в R 145
Выборочное пространство 42
Выбросы 40

Д

Двоичные переменные 23
Двумерная таблица частот 28
Двумерный анализ 75
Диаграмма рассеяния 75
Дискретная переменная 24
Дискретное распределение вероятностей 44
Дисперсионный анализ (ANOVA) 83
Доверительный интервал 65

З

Закон больших чисел, ЗБЧ 54

И

Импорт данных в R 122
Индексирование
◊ в Python 183
◊ вектора в R 119
Интегрированная среда разработки (IDE) 103
Инференциальная статистика 56

К

Кадр (рамка, фрейм) данных 120
Категориальные (качественные) переменные 22
Квартили 39
Количественные переменные 24
Коллекции Python 180
Комментарии в R 108
Коэффициент
◊ детерминации, R-квадрат 86
◊ корреляции 76
▫ Пирсона 75

Л

Линейная зависимость 75
Линейная модель 80
Линейная регрессия 80, 81
Литерал 111
Ложная зависимость 88

М

Мастер импорта наборов данных в RStudio 125
Математическое ожидание 54
Матрица корреляции 77
Машинное обучение 91
Медиана 31
Межквартильный размах 39
Меры центральной тенденции 30
Метод наименьших квадратов (ordinary least squares, OLS) 86
Методы в Python 176
Множественная (многофакторная) линейная регрессия 87
Мода 31
Модули в Python 177

Н

Наука о данных 91
Непрерывное распределение вероятностей 47
Непрерывные переменные 24
Номинальные переменные 23
Нормальное распределение вероятностей 47
Нулевая корреляция 75

О

Объектно-ориентированное программирование 192
Одномерная линейная регрессия 87
Одномерная таблица частот 28
Одномерный анализ 75
Операторы
◊ R 107
◊ в Python 172
◊ сравнения в R 109

Описательная статистика 30
Остатки (residual) 83, 85
Отрицательная линейная зависимость, 75
Отступы в Python 174

П

Пакеты в R 112
Подмножества
◊ в Python 183
◊ векторов 119
Положительная линейная зависимость 75
Порядковые переменные 23
Предел погрешности 68
Преобразование данных в R 143
Присваивание объектов в R 109
Проверка гипотез 56
Проекты RStudio 124
Псевдоним в Python 182

Р

Рабочий каталог в R 123
Разведочный анализ данных (exploratory data analysis, EDA) 19
Разделение на обучение и тестирование 161
Распределение вероятностей 44
Реляционные базы данных 96
Репрезентативная выборка 57

С

Сводные таблицы (PivotTables) 27
Словари в Python 194
Совместная вероятность 43
Среднее значение 30
Среднеквадратичная ошибка (RMSE) 163
Стандартная ошибка 68
Стандартное нормальное распределение 66
Стандартное отклонение 34
Статистика 90
Статистическая значимость теста 61
Статистическая предвзятость 57
Статистический критерий 66
Стек анализа данных 90
Структура объекта в R 118
СУБД, система управления реляционными базами данных 96
Сумма квадратов остатков 86

Т

Теоретическая вероятность 45
Тиббл (tibble) 125
Типы
◊ (режимы) данных в R 111
◊ данных в Python 175
◊ переменных 22
Точечная оценка 67, 85

У

Условная вероятность 43

Ф

Факторы в R 122
Функции пакета dplyr 132
Функция массы вероятности (probability mass function, PMF) 49

Ц

Центральная предельная теорема (ЦПТ) 51, 53
Цепочка методов 196

Э

Экспериментальная вероятность 45
Эмпирическое правило 49

Об авторе

Джордж Маунт — основатель и генеральный директор Stringfest Analytics, консалтинговой фирмы, специализирующейся на обучении и повышении квалификации в области аналитики данных. Ранее он работал с ведущими учебными платформами и компаниями в этой сфере. Джордж регулярно выступает с докладами на темы обучения анализу данных и ведет блог, расположенный по адресу:

https://stringfestanalytics.com.

Джордж Маунт получил степень бакалавра по экономике в Хиллсдейлском колледже и степень магистра в сфере финансов и информационных систем в Университете Кейс Вестерн Резерв (Case Western Reserve University). Он проживает в Кливленде, штат Огайо.

Об изображении на обложке

Птица на обложке книги — это североамериканская ореховка (Nucifraga columbiana) из семейства врановых. Эту птицу, также известную как ворона Кларка, можно встретить на западе США и в западной части Канады.

Тело североамериканской ореховки серое, а перья крыльев и хвоста — черно-белые. Ее длинный конусообразный клюв, ноги и лапы также черные, а средняя длина тела достигает 11,3 дюйма (28,8 см). Североамериканская ореховка использует свой длинный кинжалоподобный клюв для шелушения кедровых шишек и извлекает из них крупные семена, которые затем закапывает в лесные тайники, чтобы прокормиться зимой. Хотя эти птицы запоминают основные места своих тайников, те семена, которые они не находят, играют важную роль в появлении новых кедровых лесов. За один сезон североамериканская ореховка может запасти до 30 000 семян.

Остальная часть рациона североамериканской ореховки состоит из других семян, ягод, улиток, падалицы, а также яиц и молодняка других птиц. Отчасти благодаря запасам семян эти птицы начинают размножаться в конце зимы, устраивая гнезда на горизонтальных сучьях хвойных деревьев. Оба родителя заботятся о своих птенцах, которые обычно покидают гнездо на 18–21-й день после вылупления.

Природоохранный статус североамериканской ореховки — «наименее опасный», хотя есть свидетельства того, что изменение климата может повлиять на ареал и популяцию этой птицы в будущем. Многие из животных на обложках O'Reilly находятся под угрозой исчезновения; все они важны для мира. Иллюстрация на обложке этой книги выполнена Карен Монтгомери на основе черно-белой гравюры из «Иллюстрированной естественной истории» Джона Георга Вуда.

O'REILLY®

Погружение в аналитику данных

Аналитика данных может показаться сложной сферой, но если вы опытный пользователь Excel, у вас есть уникальное преимущество. С помощью этого практического руководства пользователи Excel среднего уровня получат прочное понимание аналитики и стека данных. Прочитав эту книгу, вы сможете проводить исследовательский анализ данных и проверку гипотез с помощью языков программирования Python и R.

Исследование и проверка взаимосвязей — основа аналитики. Используя описанные инструменты и механизмы, вы освоите более продвинутые методы анализа данных. Джордж Маунт подробно объясняет ключевые статистические концепции с помощью электронных таблиц, а затем помогает применить полученные знания об обработке данных для написания программ на языках R и Python.

Эта практическая книга поможет вам:

- **Изучить основы аналитики в Excel.** Используйте Excel для проверки взаимосвязи между переменными, научитесь применять его возможности в статистике и аналитике.
- **Перейти от Excel к R.** Перенесите данные в R, язык программирования с открытым исходным кодом, специально разработанный для выполнения статистического анализа.
- **Перейти от Excel к Python.** Узнайте, как перенести информацию из Excel в Python, научитесь выполнять разведочный анализ, проверку гипотез, а также полный анализ данных средствами этого языка.

«Джордж подробно рассказывает, что нужно сделать, чтобы перейти от Excel к науке о данных и аналитике».

— **Джордан Голдмайер**, обладатель сертификата Microsoft Excel MVP

«Эта книга — уникальное пособие, которое можно использовать и как справочник, и как учебник по бизнесу и аналитике данных».

— **Айден Джонсон**, специалист по анализу данных и преподаватель

Джордж Маунт — основатель и генеральный директор Stringfest Analytics, консалтинговой фирмы, специализирующейся на обучении и повышении квалификации в области аналитики данных, ранее работал с ведущими учебными платформами и компаниями в этой сфере. Джордж регулярно ведет блоги и выступает с докладами на темы обучения анализу данных.

ISBN 978-5-9775-6866-1

9 785977 568661

191036, Санкт-Петербург,
Гончарная ул., 20
Тел.: (812) 717-10-50,
339-54-17, 339-54-28
E-mail: mail@bhv.ru
Internet: www.bhv.ru

bhv®