Содержание

Введение

Если вы никогда прежде не использовали Shiny, добро пожаловать! *Shiny* представляет собой фреймворк языка программирования R, позволяющий с легкостью создавать функциональные интерактивные веб-приложения. С помощью Shiny вы можете перенести свою работу в R и представить ее результаты в браузере, чтобы все могли свободно ими пользоваться. Shiny способствует разработке сложных и эффективных веб-приложений с минимумом усилий.

В прошлом разработка веб-приложений на R давалась программистам весьма непросто, и на то было две основные причины:

- им нужно было в полной мере владеть современными веб-технологиями, включая языки HTML, CSS и JavaScript;
- в сложных интерактивных приложениях им приходилось внимательно отслеживать все связи между элементами, чтобы изменения входных значений влияли только на связанные с ними выходные.

Фреймворк Shiny значительно облегчает работу программистам при создании веб-приложений за счет:

- предоставления тщательно проработанного набора функций пользовательского интерфейса для автоматического генерирования кода HTML, CSS и JavaScript, необходимого для решения конкретных задач. Это означает, что вам не понадобится доскональное знание этих языков программирования и разметки, пока вам не станет тесно в рамках предоставляемых Shiny возможностей;
- применения нового стиля программирования, получившего название реактивное. С помощью этой концепции можно легко отслеживать и поддерживать зависимости между фрагментами кода. На практике это означает, что при изменении значения входного элемента Shiny автоматически определит, как с наименьшими усилиями обновить все связанные выходные элементы.

Разработчики используют Shiny для:

- создания дашбордов, помогающих в отслеживании высокоуровневых показателей с возможностью проводить детализированный анализ при необходимости;
- замены сотен страниц в формате PDF на одно интерактивное приложение, позволяющее пользователю переключаться между нужными ему результатами;
- донесения информации о сложных моделях до аудитории, не обладающей техническими знаниями, в виде информативных визуализаций и средств интерактивного анализа;
- автоматизации анализа данных в общих рабочих процессах с заменой процесса обмена электронными сообщениями на приложение Shiny, позволяющее пользователям загружать свои данные и выполнять

стандартный анализ. Таким образом можно сделать доступным продвинутый анализ в R пользователям, не обладающим навыками программирования;
- создания интерактивных демонстраций при обучении статистике и концепциям науки о данных, позволяющих студентам менять входные значения и наблюдать за изменениями в итоговом анализе.

Иными словами, с Shiny вы с легкостью можете делегировать некоторые свои суперспособности в R всякому, у кого есть доступ в интернет.

Для кого предназначена эта книга?

Прочитать данную книгу стоит двум основным группам аудитории:

- разработчикам R, заинтересованным в освоении фреймворка Shiny с целью перехода от базового анализа к полноценным интерактивным веб-приложениям. Чтобы взять от книги все, вам необходимо обладать определенным опытом анализа данных при помощи языка R и написания функций;
- разработчикам Shiny, желающим улучшить свои навыки владения этим инструментом для написания более быстрых и эффективных приложений. Вам будет особенно полезна эта книга, если ваши приложения постепенно начинают разрастаться и вам становится все сложнее контролировать происходящие в них процессы.

Что вы узнаете из этой книги?

Книга поделена на четыре части.

1. В первой части мы познакомимся с основами фреймворка Shiny, чтобы вы могли как можно быстрее написать свое первое приложение. Мы поговорим о структуре приложения, полезных компонентах пользовательского интерфейса и основах реактивного программирования.
2. Во второй части книги мы сделаем один шаг вперед и познакомимся со способами решения распространенных задач, включая взаимодействие с пользователем, загрузку и скачивание данных, создание пользовательского интерфейса при помощи кода, сокращение дублирующихся фрагментов кода и использование Shiny совместно с tidyverse.
3. Третья часть будет посвящена углубленной теории и практике реактивного программирования – базовой парадигмы, лежащей в основе Shiny. Если вы уже работаете с этим фреймворком, вы сможете извлечь максимум выгоды из данной главы, поскольку она закладывает теоретический фундамент, который поможет вам в разработке сложных интерактивных приложений, предназначенных для решения широкого спектра задач.
4. В заключительной части книги мы завершим исследование полезных техник, призванных повысить эффективность ваших приложений Shiny.

Вы узнаете, как выполнять процесс декомпозиции сложного приложения на функции и модули, использовать пакеты для лучшей организации вашего кода, тестировать свои работы на предмет наличия ошибок, а также измерять и улучшать производительность приложений.

Чего вы не узнаете из этой книги?

Данная книга целиком и полностью посвящена созданию эффективных приложений с помощью Shiny и принципам лежащего в ее основе реактивного программирования. Я сделаю все возможное, чтобы показать вам лучшие приемы в работе с данными, программировании на языке R и инженерии программного обеспечения, но вам придется почитать и другие источники, чтобы в полной мере овладеть всеми этими навыками. Если вам нравится, как я пишу, вы можете посмотреть и другие мои книги на эту тему: *R for Data Science* (https://r4ds.had.co.nz), *Advanced R* (https://adv-r.hadley.nz) и *R Packages* (https://r-pkgs.org).

Также есть несколько тем из области Shiny, которых я не буду касаться в данной книге:

- при написании приложений мы будем рассматривать только встроенный *набор инструментов пользовательского интерфейса* (user interface toolkit). Может, он и не самый привлекательный, но для обучения очень даже подойдет. Если вам нужно больше или просто надоел базовый интерфейс, на просторах интернета есть множество пакетов, позволяющих до неузнаваемости изменить внешний вид ваших приложений. Подробнее читайте в разделе книги, посвященном фреймворку *Bootstrap*;
- тема развертывания приложений Shiny выходит за рамки данной книги, поскольку здесь многое зависит от сторонних факторов, не имеющих отношения к R, – больше культурного и организационного свойства, но никак не технического. Если вы делаете первые шаги в вопросах развертывания приложений Shiny, я бы посоветовал посмотреть выступление Джо Ченга по адресу https://www.rstudio.com/resources/rstudioconf-2019/shiny-in-production-principles-practices-and-tools/. Из него вы почерпнете все базовые вещи, связанные с развертыванием приложений, включая распространенные проблемы и способы их решения. После этого я бы рекомендовал посетить страницу https://www.rstudio.com/products/connect, посвященную продукту RStudio Connect, предназначенному для развертывания приложений в рамках организации, а также соответствующий раздел на сайте Shiny по адресу https://shiny.rstudio.com/articles/#deployment.

Требования

Перед тем как продолжить, убедитесь в том, что у вас на компьютере стоит все необходимое программное обеспечение для работы:

R

Если у вас еще не установлен R на компьютере, возможно, вы читаете не ту книгу. Мы на протяжении книги будет предполагать, что у вас есть базовые знания о языке R. Если вы хотите узнать, как использовать R, я бы рекомендовал для чтения книгу *R for Data Science* (https://r4ds.had.co.nz), которая позволит вам сделать первые шаги в этой новой для вас среде.

RStudio

RStudio представляет собой бесплатную *интегрированную среду разработки* (integrated development environment – IDE) для R. И хотя вы можете создавать и использовать приложения на Shiny в любом окружении R, включая R GUI и ESS, именно RStudio обладает некоторыми полезными особенностями, облегчающими написание, отладку и развертывание приложений Shiny. Мы рекомендуем вам загрузить RStudio Desktop по адресу https://www.rstudio.com/products/rstudio/download/ и дать ему шанс. В то же время это совершенно не обязательно для разработки приложений и чтения книги.

Пакеты R

В этой книге мы будем использовать множество пакетов R. Вы можете установить их все сразу, запустив следующий код:

```
install.packages(c(
    "gapminder", "ggforce", "gh", "globals", "openintro", "profvis",
    "RSQLite", "shiny", "shinycssloaders", "shinyFeedback",
    "shinythemes", "testthat", "thematic", "tidyverse", "vroom",
    "waiter", "xml2", "zeallot"
))
```

Если вы уже загружали пакет Shiny ранее, убедитесь, что у вас установлена его версия не ниже 1.6.0.

Принятые в книге обозначения

При написании книги мы использовали следующие обозначения:

- *курсив* – обозначает новые термины, ссылки, электронные адреса, имена и расширения файлов;
- моноширинный шрифт – используется в листингах и при обозначении программных элементов, таких как имена переменных или функций, баз данных, типов данных, переменных окружения, выражений и ключевых слов.

Использование примеров кода

Сопроводительные материалы (фрагменты кода, упражнений и т. д.) доступны для загрузки по адресу https://mastering-shiny.org/. На все листинги в книге распространяется лицензия MIT (MIT License): https://www.mit.edu/~amini/LICENSE.md.

Если у вас есть технические вопросы относительно фрагментов кода, можете написать нам по адресу bookquestions@oreilly.com.

Вы вправе использовать материалы из книги для решения своих задач. В основном код, представленный в данной книге, можно использовать в собственных программах и документации. Вы не обязаны обращаться к нам за разрешением, если речь идет не о копировании большого фрагмента текста. Например, использование нескольких фрагментов из книги при написании своей программы не потребует специального разрешения. В то же время продажа или распространение материалов, принадлежащих O'Reilly, предполагает наличие разрешения. Ответы на вопросы с использованием цитирования из этой книги не требуют разрешения. А включение большой части текста из этой книги в свою документацию – требует.

Мы будем признательны, если вы будете ссылаться на книгу при цитировании, хотя и не требуем от вас этого. Обычно такие ссылки включают в себя название книги, имя автора и название издательства, а также ISBN. Например, «Mastering Shiny by Hadley Wickham (O'Reilly). Copyright 2021 Hadley Wickham, 978-1-492-04738-4».

Для получения разрешений можно обратиться по адресу permissions@oreilly.com.

Благодарности

Данная книга была написана в открытую, а по ее окончании главы свободно выкладывались в Twitter. Таким образом, книгу можно назвать полноценным совместным опытом, а в процессе ее написания очень многие люди вычитывали черновики, исправляли опечатки, предлагали улучшения и вносили свой вклад в содержимое. Без этой неоценимой помощи книга никогда не стала бы такой, какой вы ее видите, и я очень признателен всем, кто внес свою лепту в ее написание.

Мы хотели бы поблагодарить поименно всех 83 человек, предложивших свои улучшения на GitHub (в алфавитном порядке): Adam Pearce (@1wheel), Adi Sarid (@adisarid), Alexandros Melemenidis (@alex-m-ffm), Anton Klåvus (@antonvsdata), Betsy Rosalen (@betsyrosalen), Michael Beigelmacher (@brooklynbagel), Bryan Smith (@BSCowboy), c1au6io_hh (@c1au6i0), @canovasjm, Chris Beeley (@ChrisBeeley), @chsafouane, Chuliang Xiao (@ChuliangXiao), Conor Neilson (@condwanaland), @d-edison, Dean Attali (@daattali), Daniel-David521 (@Danieldavid521), David Granjon (@DivadNojnarg), Eduardo Vásquez (@edovtp), Emil Hvitfeldt (@EmilHvitfeldt), Emilio (@emilopezcano), Emily Riederer (@emilyriederer), Eric Simms (@esimms999), Federico Marini (@federicomarini), Frederik Kok Hansen (@fkoh111), Frans van Dunné (@FvD), Giorgio Comai (@giocomai), Hedley (@heds1), Henning (@henningsway), Hlynur (@hlynurhallgrims), @hsm207, @jacobxk, James Pooley (@jamespooley), Joe Cheng (@jcheng5), Julien Colomb (@jcolomb), Juan C. Rodriguez (@jcrodriguez1989), Jennifer (Jenny) Bryan (@jennybc), Jim Hester (@jimhester), Joachim Gassen (@joachim-gassen), Jon Calder (@jonmcalder), Jonathan Carroll (@jonocarroll), Julian Stanley (@julianstanley), @jyuu, @kaanpekel, Karandeep Singh (@kdpsingh), Robert Kirk DeLisle (@KirkDCO), Elaine (@loomalaine), Malcolm Barrett (@malcolmbarrett), Marly Gotti (@marlycormar), Matthew Wilson (@MattW-Geospatial), Matthew T. Warkentin (@mattwarkentin), Mauro Lepore (@maurolepore), Maximilian Rohde (@maxdrohde), Matthew Berginski (@mbergins), Michael Dewar (@michael-dewar), Mine Cetinkaya-Rundel (@mine-cetinkaya-rundel), Maria Paula Caldas (@mpaulacaldas), nthobservation (@nthobservation), Pietro Monticone (@pitmonticone), psychometrician (@psychometrician), Ram Thapa (@raamthapa), Janko Thyson (@rappster), Rebecca Janis (@rbjanis), Tom Palmer (@remlapmot), Russ Hyde (@russHyde), Barret Schloerke (@schloerke), Scott (@scottyd22), Matthew Sedaghatfar (@sedaghatfar), Shixiang Wang (@ShixiangWang), Praer (Suthira Owlarn) (@sowla), Sébastien Rochette (@statnmap), @stevensbr, André Calero Valdez (@Sumidu), Tanner Stauss (@tmstauss), Tony Fujs (@tonyfujs), Stefan Moog (@trekonom), Jeff Allen (@trestletech), Trey Gilliland (@treygilliland), Albrecht (@Tungurahua), Valeri Voev (@ValeriVoev), Vickus (@Vickusr), William Doane (@WilDoane), 黄湘云 (@XiangyunHuang) и gXcloud (@xwydq).

Как была написана эта книга

Книга была написана в RStudio с использованием пакета bookdown (http://bookdown.org).

При написании книги использовался R версии 4.0.3 (2020-10-10) и следующие версии пакетов:

Пакет	Версия	Источник
gapminder	0.3.0	standard (@0.3.0)
Ggforce	0.3.2	standard (@0.3.2)
Gh	1.2.0	standard (@1.2.0)
Globals	0.14.0	standard (@0.14.0)
Openintro	2.0.0	standard (@2.0.0)
Profvis	0.3.7.9000	GitHub (rstudio/profvis@ca1b272)
RSQLite	2.2.3	standard (@2.2.3)
Shiny	1.6.0	standard (@1.6.0)
shinycssloaders	1.0.0	standard (@1.0.0)
shinyFeedback	0.3.0	standard (@0.3.0)
shinythemes	1.2.0	standard (@1.2.0)
Testthat	3.0.2.9000	Hub (r-lib/testthat@4793514)
Thematic	0.1.1	GitHub (rstudio/thematic@d78d24a)
Tidyverse	1.3.0	standard (@1.3.0)
Vroom	1.3.2	standard (@1.3.2)
Waiter	0.2.0	standard (@0.2.0)
xml2	1.3.2	standard (@1.3.2)
Zeallot	0.1.0	standard (@0.1.0)

Часть I

ПРИСТУПАЕМ К РАБОТЕ

Прочитав первые четыре главы книги, вы сможете написать свое первое приложение с использованием фреймворка Shiny. Первая глава будет посвящена базовым принципам приложения Shiny и его структуре. Во второй и третьей главах вы погрузитесь в детальный разбор двух основных составляющих любого приложения Shiny: клиентского интерфейса, или *фронтенда* (того, что пользователь видит в браузере), и серверной части, или *бэкенда* (кода, заставляющего приложение работать). А завершим мы первую часть книги примером из практики, который поможет закрепить все полученные ранее знания.

Глава 1

Ваше первое приложение Shiny

Введение

В этой главе вы создадите свое первое приложение Shiny. Начнем мы с минимальной шаблонной заготовки для полноценного приложения, после чего научимся запускать и останавливать его. Далее вы узнаете, что из себя представляют два ключевых компонента любого приложения Shiny, а именно *UI* (сокращенно от *user interface* – интерфейс пользователя), определяющий внешний вид приложения, и *функция server*, в которой реализовано поведение приложения. Фреймворк Shiny использует принципы *реактивного программирования* (reactive programming) для автоматического обновления элементов вывода при изменении значений элементов ввода, так что завершим эту главу мы знакомством с третьим важнейшим компонентом приложений Shiny – *реактивными выражениями* (reactive expressions).

Если вы еще не установили пакет Shiny, сейчас самое время сделать это, запустив следующую строку кода:

```
install.packages("shiny")
```

Если пакет Shiny у вас уже установлен, проверьте, что его версия – 1.5.0 или выше, при помощи функции `packageVersion("shiny")`.

Затем загрузите пакет Shiny в вашу текущую сессию:

```
library(shiny)
```

Создание директории и файла приложения

Создать *приложение Shiny* (Shiny app) можно несколькими способами. Простейший из них – создать новую директорию для приложения и сохранить в ней файл с именем *app.R*. В этом файле будет содержаться вся необходимая информация как по внешнему виду приложения, так и по его поведению.

Итак, создайте папку и сохраните в ней файл *app.R* со следующим содержимым:

```
library(shiny)
ui <- fluidPage(
    "Hello, world!"
)
server <- function(input, output, session) {
}
shinyApp(ui, server)
```

Не поверите, но это уже полноценное, хоть и весьма тривиальное, приложение Shiny! Этот простой скрипт делает ровно четыре вещи.

1. Загружает пакет Shiny при помощи инструкции `library(shiny)`.
2. Определяет пользовательский интерфейс – страницу HTML, с которой будет взаимодействовать пользователь. В данном случае это страница со словами «Hello, world!».
3. Формирует поведение приложения путем определения функции *`server`*. У нас эта функция пустая, так что наше приложение не будет делать ровным счетом ничего, но очень скоро мы это исправим.
4. Вызывает функцию *`shinyApp(ui, server)`* для сборки и запуска приложения Shiny с пользовательским интерфейсом и серверным поведением.

Примечание. В RStudio предусмотрено два удобных способа создания приложения Shiny:

- вы можете создать новую папку и файл *app.R* с шаблонным приложением, выбрав в меню **File** пункт **New Project**, затем в открывшемся диалоговом окне щелкнув на пункт **New Directory** и выбрав вариант **Shiny Web Application**;
- если у вас уже есть файл *app.R*, вы можете воспользоваться возможностью быстрой вставки заранее заготовленного фрагмента кода, называемого *сниппетом* (snippet), – для этого введите текст `shinyapp` и нажмите сочетание клавиш **Shift**+**Tab**.

Запуск и остановка

Запустить приложение также можно несколькими способами:

- нажать на кнопку **Run App** на панели инструментов, как показано на рис. 1.1;
- использовать комбинацию клавиш **Cmd**/**Ctrl**+**Shift**+**Enter**;
- если вы не используете RStudio, вы можете загрузить (`source()`)[1] весь документ или вызвать функцию `shiny::runApp()` с указанием пути к папке, в которой располагается файл *app.R*.

[1] Дополнительные обрамляющие скобки здесь очень важны. Функция `shinyApp()` создает приложение только при печати, а скобки позволяют подгрузить последний результат из файла, который в противном случае вернулся бы невидимым.

Рис. 1.1 ❖ Кнопка **Run App** располагается в правой части панели инструментов

Выберите один из перечисленных способов запуска приложения, и вы увидите его на экране в виде, показанном на рис. 1.2. Наши поздравления! Вот вы и написали свое первое приложение Shiny!

Рис. 1.2 ❖ Самое простое приложение Shiny

Перед тем как закрыть приложение, вернитесь в RStudio и взгляните на консоль. Вы обнаружите что-то типа этого:

```
#> Listening on http://127.0.0.1:3827
```

Здесь вы видите адрес, где находится ваше приложение, – 127.0.0.1, что означает «этот компьютер» и номер порта, присвоенный случайным образом, – в данном случае это 3827. Вы можете ввести этот адрес в любой совместимый[1] браузер, чтобы открыть еще одну копию приложения.

Заметьте, что при запущенном приложении оболочка R переходит в состояние занятости: командная строка не видна, а на панели инструментов в консоли показывается иконка с символом остановки. При запуске приложения Shiny происходит блокировка консоли R, так что до остановки приложения вы не сможете вводить в нее новые команды.

Остановить выполнение приложения и получить доступ к консоли можно одним из следующих способов:

---

[1] Shiny стремится поддерживать все современные браузеры (https://www.rstudio.com/about/platform-support). Обратите внимание, что Internet Explorer версии ниже IE11 не поддерживается для запуска Shiny напрямую из сессии R. При этом приложения Shiny, развернутые на сервере Shiny или на сайте *ShinyApps.io*, могут работать в IE10 (более ранние версии IE больше не поддерживаются).

- нажать на иконку с символом остановки на панели инструментов в консоли;
- перейти в консоль и нажать на клавишу **Esc** (или сочетание клавиш **Ctrl**+**C**, если вы не используете RStudio);
- закрыть окно приложения Shiny.

Традиционный для приложений Shiny процесс разработки состоит из написания кода, запуска приложения, его проверки, написания нового кода, повторного запуска и так по кругу. Если вы используете RStudio, вам не нужно будет каждый раз останавливать и запускать приложение, чтобы увидеть изменения. Вместо этого вы можете нажать на кнопку **Reload app** на панели инструментов или сочетание клавиш **Cmd**/**Ctrl**+**Shift**+**Enter**. Другие этапы разработки приложения мы обсудим в главе 5.

Добавление элементов пользовательского интерфейса

Теперь давайте добавим пару элементов управления в наше приложение, чтобы оно не выглядело таким пустым. К примеру, выведем информацию о всех доступных наборах данных из пакета *datasets*.

Замените код в переменной `ui` на приведенный ниже:

```
ui <- fluidPage(
    selectInput("dataset", label = "Dataset", choices = ls("package:datasets")),
    verbatimTextOutput("summary"),
    tableOutput("table")
)
```

Здесь мы видим следующие функции:

- *`fluidPage()` – функция разметки* (layout function), отвечающая за визуальную структуру приложения. Далее в этой книге мы еще поговорим о ней более подробно;
- *`selectInput()`* – элемент ввода, представляющий собой список, с помощью которого пользователь может сделать свой выбор. В данном случае мы присвоили списку метку *Dataset* и заполнили его наименованиями встроенных в R наборов данных. Подробнее об элементах ввода вы узнаете далее в этой главе;
- *`verbatimTextOutput()`* и *`tableOutput()`* – элементы вывода, предназначенные для отображения обработанной Shiny информации (как именно это происходит, мы покажем очень скоро). При этом элемент `verbatimTextOutput()` служит для вывода текста, а `tableOutput()` – для отображения табличных данных. Больше об элементах вывода вы узнаете в процессе чтения этой главы.

Функции разметки и элементы ввода и вывода применяются для совершенно разных целей, но все они, по сути, представляют собой лишь причудливые способы генерирования HTML-кода, и если вы вызовете любую

из этих функций за пределами приложения Shiny, то увидите код HTML на консоли – ничего более. Вскоре вы узнаете, как именно работают эти функции и элементы «под капотом».

Давайте запустим приложение снова. Теперь оно выглядит так, как показано на рис. 1.3, – в виде страницы с выпадающим списком. При этом мы видим только элемент ввода, но не видим определенные нами элементы вывода, поскольку еще не сообщили Shiny, как между собой должны быть связаны ввод и вывод.

Рис. 1.3 ❖ Приложение Shiny с интерфейсом пользователя

Добавление поведения

Теперь давайте оживим наши элементы вывода, определив их поведение в функции `server`.

Интерактивность приложений Shiny обеспечивается при помощи реактивного программирования. В главе 3 мы более подробно поговорим о реактивном программировании, а сейчас вам достаточно будет знать, что эта концепция позволяет Shiny определиться со способом выполнения расчетов, не запуская сами действия. Это как разница между тем, чтобы дать человеку рецепт и потребовать от него приготовить вам бутерброд.

Мы расскажем Shiny, как заполнять наши элементы вывода, предложив для этого свои рецепты или инструкции. Замените пустую функцию `server` на код, представленный ниже:

```
server <- function(input, output, session) {
    output$summary <- renderPrint({
        dataset <- get(input$dataset, "package:datasets")
        summary(dataset)
    })

    output$table <- renderTable({
        dataset <- get(input$dataset, "package:datasets")
        dataset
    })
}
```

В левой части от оператора присваивания (`<-`) располагаются идентификаторы наших выходных элементов вида `output$ID` – именно с их помощью мы сообщаем Shiny инструкцию для заполнения элементов с указанными идентификаторами. Справа от оператора находятся специальные *функции отображения* (render function) с заключенным в них кодом. Каждая функция с шаблонным именем `render{Тип}` призвана генерировать вывод определенного типа (например, текст, таблицу или график) и часто соотносится с шаблонными

функциями вида `{тип}Output`. В нашем случае функция `renderPrint()` работает совместно с `verbatimTextOutput()` для отображения статистической сводки в виде *текста с фиксированной шириной* (verbatim), а функция `renderTable()` действует в паре с функцией `tableOutput()` для вывода табличных данных.

Снова запустите приложение и посмотрите, что произойдет при изменении выбора в выпадающем списке. На рис. 1.4 показано, как будет выглядеть приложение после запуска.

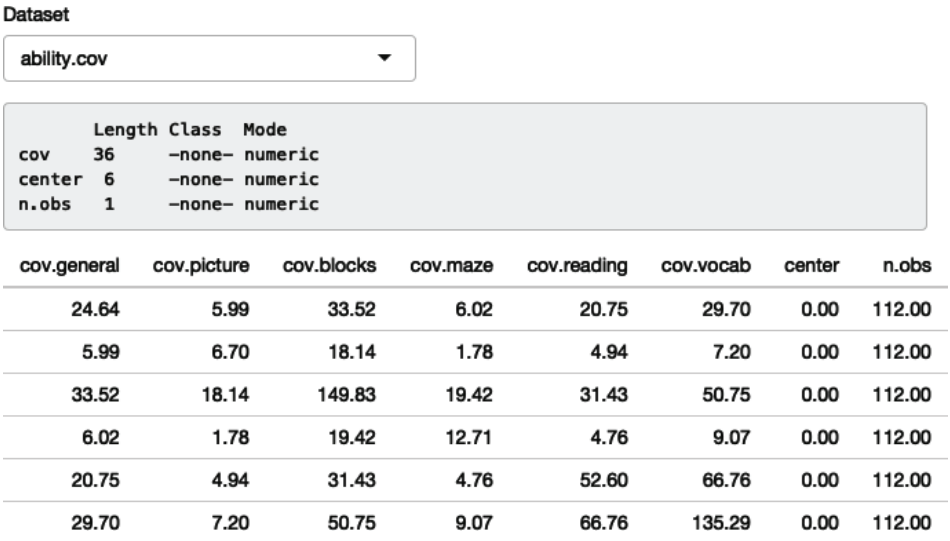

Рис. 1.4 ❖ Соединив воедино элементы ввода и вывода, мы получили полноценное интерактивное приложение

Обратите внимание, что выходной текст и таблица обновляются сразу после выбора набора данных из списка. Такая зависимость создается неявно, поскольку мы ссылаемся на элемент ввода `input$dataset` непосредственно в функциях вывода. В переменной `input$dataset` находится текущее значение элемента ввода с идентификатором `dataset`, и элементы вывода будут обновляться сразу после изменения этого значения. В этом состоит суть *реактивности* (reactivity): элементы вывода будут автоматически *реагировать* (react), то есть пересчитываться, при изменении значений соответствующих элементов ввода.

Снижение дублирования кода при помощи реактивных выражений

Вы наверняка заметили, что даже в таком простом примере, как этот, мы вынуждены были продублировать часть кода – в частности, представленную ниже строку мы написали дважды:

```
dataset <- get(input$dataset, "package:datasets")
```

Но вы, будучи программистом, прекрасно знаете все минусы дублирования кода: от дополнительных временных затрат на его выполнение до сложности с поддержкой и отладкой. В нашем случае это не критично, но мне хотелось продемонстрировать эту проблему на предельно простом примере.

В традиционном программировании на языке R мы обычно используем две техники для снижения количества повторений в коде: значения сохраняем в переменные, а вычисления – в функции. К сожалению, применительно к Shiny ни одна из этих техник не позволит добиться желаемого результата, и причины этого будут подробно описаны при обсуждении концепции реактивного программирования. Таким образом, здесь нам просто не обойтись без *реактивных выражений* (reactive expression).

Реактивные выражения создаются путем заключения фрагмента кода в блок *`reactive`*`({...})` и присвоения его переменной. Запуск реактивного выражения осуществляется путем обращения к этой переменной как к функции. И хотя внешне это действительно выглядит как обычный вызов функции, в случае с реактивным выражением есть один важный нюанс – фактически оно вычисляется только при первом обращении, после чего результат вычисления кешируется до тех пор, пока не потребуется его обновить.

Ниже показано, как можно переписать нашу функцию `server()`, чтобы в ней использовались реактивные выражения. Внешне приложение будет вести себя так же, как и прежде, но его эффективность повысится за счет того, что извлекать набор данных мы будем один раз, а не два:

```
server <- function(input, output, session) {
    # Создаем реактивное выражение
    dataset <- reactive({
        get(input$dataset, "package:datasets")
    })

    output$summary <- renderPrint({
        # Используем реактивное выражение, обращаясь к нему как к функции
        summary(dataset())
    })

    output$table <- renderTable({
        dataset()
    })
}
```

Позже мы подробнее поговорим о реактивных выражениях, но даже полученного вами на данном этапе поверхностного знания об элементах ввода и вывода, а также о реактивных выражениях будет достаточно, чтобы писать полезные приложения Shiny!

Заключение

В данной главе вы создали простое приложение Shiny. Конечно, оно не потрясает воображение и не несет большой пользы, но вы увидели, как легко

можно сконструировать веб-приложение с использованием базовых знаний в R. В следующих двух главах вы еще больше узнаете об интерфейсе пользователя и реактивном программировании – двух основных строительных блоках Shiny. Теперь пришло время взглянуть на удобные шпаргалки, которые помогут вам лучше запомнить основы структуры приложений Shiny.

Упражнения

Упражнение 1

Создайте приложение, приветствующее пользователя по имени. Вы пока еще не знаете всех функций, которые понадобятся для выполнения этого задания, так что мы включили несколько строк кода в случайном порядке. Подумайте, какие из них вам понадобятся для приложения, и вставьте их в свой код в правильном порядке:

```
tableOutput("mortgage")
output$greeting <- renderText({
```

```
    paste0("Hello ", input$name)
})
numericInput("age", "How old are you?", value = NA)
textInput("name", "What's your name?")
textOutput("greeting")
output$histogram <- renderPlot({
    hist(rnorm(1000))
}, res = 96)
```

Упражнение 2

Представьте, что ваш коллега пишет приложение, в котором пользователь может ввести значение (*x*) между 1 и 50 и посмотреть, чему будет равно произведение введенного значения и цифры 5. Вот его первая попытка:

```
library(shiny)

ui <- fluidPage(
    sliderInput("x", label = "Если x равно", min = 1, max = 50, value = 30),
    "то x умножить на 5 будет",
    textOutput("product")
)

server <- function(input, output, session) {
    output$product <- renderText({
        x * 5
    })
}

shinyApp(ui, server)
```

К сожалению, при запуске приложение выдало ошибку, показанную на рис. 1.5.

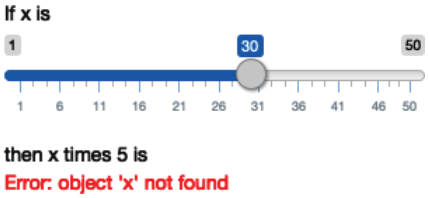

Рис. 1.5 ❖ Объект x не найден

Помогите коллеге решить проблему!

Упражнение 3

Расширьте приложение из предыдущего упражнения таким образом, чтобы пользователь мог вводить оба множителя: *x* и *y*. Итоговый результат должен выглядеть так, как показано на рис. 1.6.

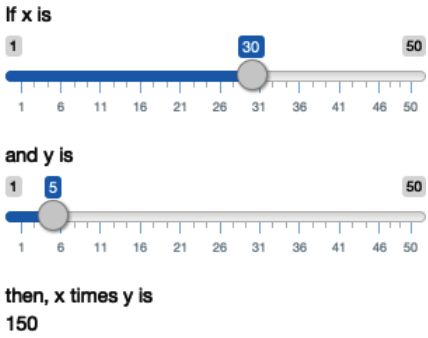

Рис. 1.6 ❖ Приложение, рассчитывающее произведение введенных чисел

Упражнение 4

Взгляните на код приложения, ставшего развитием предыдущего примера. Что здесь добавилось? Как можно снизить количество дублирующегося кода за счет применения реактивных выражений?

```
library(shiny)

ui <- fluidPage(
    sliderInput("x", "Если x равно", min = 1, max = 50, value = 30),
    sliderInput("y", "и y равно", min = 1, max = 50, value = 5),
    "то (x * y) будет", textOutput("product"),
    "(x * y) + 5 будет", textOutput("product_plus5"),
    "а (x * y) + 10 будет", textOutput("product_plus10")
)

server <- function(input, output, session) {
    output$product <- renderText({
        product <- input$x * input$y
        product
    })
    output$product_plus5 <- renderText({
        product <- input$x * input$y
        product + 5
    })
    output$product_plus10 <- renderText({
        product <- input$x * input$y
        product + 10
    })
}

shinyApp(ui, server)
```

Упражнение 5

Приведенное ниже приложение очень похоже на то, что вы видели ранее в этой главе: вы выбираете набор данных из пакета (на этот раз мы использовали пакет ggplot2) и на выходе получаете сводку по нему и график. Заметьте, что здесь мы использовали реактивные выражения для уменьшения дублирования кода. Но в коде есть три ошибки. Вы сможете найти и исправить их?

```
library(shiny)
library(ggplot2)

datasets <- c("economics", "faithfuld", "seals")

ui <- fluidPage(
    selectInput("dataset", "Dataset", choices = datasets),
    verbatimTextOutput("summary"),
    tableOutput("plot")
)

server <- function(input, output, session) {
    dataset <- reactive({
        get(input$dataset, "package:ggplot2")
    })

    output$summary <- renderPrint({
        summary(dataset())
    })

    output$plot <- renderPlot({
        plot(dataset)
    }, res = 96)
}

shinyApp(ui, server)
```