Учебник по языку R для начинающих

Борис Демешев

2016-08-25

Оглавление

О книге **5**

1 Первые шаги **7**

1.1 Установка софта . . . 7
1.2 Весёлый калькулятор . . . 8
1.3 Первый скрипт . . . 8
1.4 Установка и подключение пакетов . . . 9
1.5 Чтение и запись данных . . . 13
1.6 Интернет-источники данных . . . 14
1.7 Стиль кода . . . 15
1.8 Две записи функций . . . 16
1.9 Манипуляции с данными . . . 17
1.10 Графики . . . 17
1.11 У меня ошибка! . . . 17
1.12 Ресурсы по R . . . 17

2 Друзья R **19**

2.1 Рабочий процесс . . . 19
2.2 Контроль версий . . . 19
2.3 Латех . . . 19
2.4 Маркдаун . . . 19
2.5 Воспроизводимые исследования . . . 19
2.6 Написание своего пакета . . . 19
2.7 Вычисления в облаке . . . 20
2.8 Презентации . . . 20
2.9 Про эту книжку . . . 20

3 Статистика и не только **23**

3.1 Генерирование случайных величин . . . 23
3.2 Базовые статистические тесты . . . 24
3.3 Множественная регрессия . . . 24
3.4 Квантильная регрессия . . . 25
3.5 Инструментальные переменные . . . 25
3.6 Гетероскедастичность . . . 25
3.7 Работа с качественными переменными . . . 25
3.8 Логит и пробит с визуализацией . . . 25
3.9 Метод главных компонент . . . 25
3.10 Мультиколлинеарность . . . 25
3.11 LASSO . . . 25
3.12 Метод максимального правдоподобия . . . 26
3.13 Метод опорных векторов . . . 26
3.14 Случайный лес . . . 26
3.15 Экспоненциальное сглаживание . . . 26
3.16 ARMA модели . . . 26
3.17 GARCH . . . 26
3.18 VAR и BVAR . . . 26
3.19 Панельные данные . . . 26
3.20 Байесовский подход: первые шаги . . . 26
3.21 Байесовский подход: STAN . . . 27
3.22 Карты . . . 27
3.23 Дифференциальные уравнения . . . 27
3.24 Задачи оптимизации . . . 27

О книге

Сейчас живут три кита анализа данных и научных вычислений — Julia, Python и R. Эта книга про одного из китов!

```
library("knitr")
library("tikzDevice")

activateTikz <- function() {

  # tikz plots options
  options(tikzDefaultEngine = "xetex")

  # cash font metrics for speed:
  options(tikzMetricsDictionary = "./tikz_metrics")

  add_xelatex <- c("\\defaultfontfeatures{Ligatures=TeX,Scale=MatchLowercase}",
                   "\\setmainfont{Linux Libertine O}",
                   "\\setmonofont{Linux Libertine O}",
                   "\\setsansfont{Linux Libertine O}",
                   "\\newfontfamily{\\cyrillicfonttt}{Linux Libertine O}",
                   "\\newfontfamily{\\cyrillicfont}{Linux Libertine O}",
                   "\\newfontfamily{\\cyrillicfontsf}{Linux Libertine O}")

  options(tikzXelatexPackages = c(getOption("tikzXelatexPackages"),
                                  add_xelatex))

  # does remove warnings:
  # it is important to remove fontenc package wich is loaded by default
  options(tikzUnicodeMetricPackages = c("\\usetikzlibrary{calc}",
                                        "\\usepackage{fontspec, xunicode}", add_xelatex))


  opts_chunk$set(dev = "tikz", dev.args = list(pointsize = 11))
}


colFmt <- function(x, color) {
  outputFormat <- opts_knit$get("rmarkdown.pandoc.to")
```

```
  if (outputFormat == "latex") {
    result <- paste0("\\textcolor{", color, "}{", x, "}")
  } else if (outputFormat %in% c("html", "epub")) {
    result <- paste0("<font color='", color, "'>", x, "</font>")
  } else {
    result <- x
  }
  return(result)
}

outputFormat <- opts_knit$get("rmarkdown.pandoc.to")


if (outputFormat == "latex") {
  activateTikz()
}
```

Данная версия книги скомпилирована для latex.

Глава 1

Первые шаги

ываыв а ываыв

1.1. Установка софта

Казалось бы, чего проще — поставить программу?! Однако не всегда всё идёт гладко.

Самая распространённая проблема, с которой мне доводилось бороться на разных компьютерах, — это русские буквы и пробелы в названиях файлов и папок под Windows.

Заповедь 1. Если ты используешь Windows, то никогда при серьёзной работе не используй русские буквы и пробелы в названиях файлов и папок. Папку с котиками можно назвать мои котики :)

Заповедь 1 легко нарушить даже не осознавая этого. Если имя пользователя Windows написано русскими буквами, например, Машенька, то все документы этого пользователя будут находиться в папке C:/Users/Машенька/Documents/.

Поэтому для серьёзной работы под Windows нужно создать нового пользователя с английским именем, например, Mashenka и залогиниться из-под него. Переименование старого пользователя не помогает.

1.1.1. R

...

1.1.2. Rstudio

...

1.1.3. LaTeX

...

1.1.4. git-клиент

...

1.1.5. Текстовый редактор

Самое важное: Word — это не текстовый редактор! Текстовый редактор — это программа с помощью которой редактируют файлы с текстовым содержимым, а Word сохраняет файлы в специальном формате, где кроме текста сохраняется ещё куча дополнительной информации. Расширение у текстового файла может быть довольно произвольным, .txt, .md, .R, .tex и так далее.

Текстовых редакторов много. Я советую кросс-платформенный открытый и бесплатный Atom.

1.1.6. STAN

...

1.1.7. jupyter

...

1.1.8. gretl

Тебе страшно? Тебя пугает даже список того, что нужно установить? Ты боишься R, а регрессию надо построить через 5 минут? Тогда разумное спасение — это gretl. Для gretl не обязательно учиться программировать: статистические тесты, графики и эконометрические модели доступны через меню. Кроме того, gretl даёт возможность пользователю взаимодействовать с R, что спасает в тех случаях, когда возможностей gretl не хватает.

Естественно, gretl кросс-платформенный открытый и бесплатный, gretl.

1.2. Весёлый калькулятор

...

1.3. Первый скрипт

....

Если текст программы содержит русские или другие неанглийские буквы, например, в комментариях, то при сохранении файла Rstudio предложит выбрать кодировку.

картинка

Кодировка определяет какой конкретно числовой код будет сопоставлен в записанном файле каждой букве. Например, букве ё в кодировке UTF-8 сопоставлен десятичный[1] код 1105.

Для русского языка есть несколько распространённых кодировок: UTF-8 и CP1251. Linux и Macos используют по умолчанию кодировку UTF-8, а вот Windows[2] сохраняет простые текстовые файлы в кодировке CP1251.

Если русскоязычный файл записать в одной кодировке, а пытаться открыть с помощью другой, то мы увидим на экране "кракозябры". Поэтому хорошо, когда все используют одну кодировку. Кодировка UTF-8 более универсальна, чем CP1251. Например, с помощью кодировки UTF-8 в одном тексте можно использовать и русские буквы и французские акценты и китайские иероглифы.

> Мы всегда будем использовать кодировку UTF-8.

1.4. Установка и подключение пакетов

Одна из сильных сторон R — это открытость: каждая домохозяйка может написать свой пакет для R и выложить его в публичное пользование. Пакеты расширяют возможности R. Для R написано более 10 тысяч пакетов. Среди них есть и откровенный мусор, и бриллианты, например, ggplot2, настолько ценные, что их копируют в другие языки программирования.

Скорее всего нужный тебе пакет можно найти:

1. В официальном хранилище пакетов R, CRAN.

Здесь пакеты прошли минимальное тестирование. Это отнюдь не гарантия качества пакета, но всё же серьезный давно функционирующий пакет наверняка будет выложен на CRAN.

2. В системе репозиториев github.com.

Здесь, как правило, разработчики публикуют более свежие версии пакетов, ещё не выложенные на CRAN, или молодые пакеты в процессе разработки.

3. В хранилище пакетов для биологов bioconductor.

Это своя отдельная экосистема пакетов R со специальным инсталлятором, блэкджеком и поэтэссами.

Есть и другие хранилища пакетов, например, R-forge и ..., но они гораздо менее популярны.

Сначала надо определиться, какой пакет тебе нужен. Можно погуглить, можно воспользоваться официальным классификатором пакетов R ....

Чтобы начать использовать какой-нибудь пакет R нужно сделать две вещи. Во-первых, установить его. Установка означает, что пакет будет скачан из Интернета и сохранён в специальной папке на

[1] Если шестнадцатиричный, то 0451. Ради интереса можно посмотреть сопоставление букв и их кодов в UTF-8, например, на unicode-table.com

[2] На самом деле всё немного хитрее и сама Windows технически использует UTF-16, а вот многие приложения под ней — CP1251.

жёстком диске. Установка пакета выполняется один раз. Каждый раз при использовании пакета устанавливать его не нужно. Переустанавливать пакет имеет смысл только если вышла новая его версия. Во-вторых, нужно подключить пакет. Подключение пакета выполняется каждый раз перед его использованием.

С репозитория CRAN пакет ставится командой R:

```
install.packages("имя пакета")
```

В Rstudio установить пакет с репозитория CRAN можно через меню: Tools - Install packages. Далее нужно набрать название пакета, можно указать сразу несколько названий через пробел, и нажать Install.

> Главное при установке пакета — не бояться сообщений красным цветом!

Любые **сообщения** (messages) R выводит красным цветом и по неопытности их можно принять за ошибку, что скорее всего не так. Ошибка всегда сопровождается словом Error. Если слова Error нет, то всё идёт по плану!

Почему R использует красный цвет? Потому, что установка пакета — это потенциально опасное действие, как и установка любой программы. Пакеты на официальном репозитории CRAN проходят определённую проверку, но если ты используешь R для многомиллионных сделок каждый день, то неплохо бы точно знать, что ты ставишь :)

Установить пакет с github.com немногим сложнее. Здесь надо знать не только название пакета, но и название репозитория, где он хранится. Часто название репозитория — это фамилия автора пакета. Официальной классификации всех пакетов R на github нет, поэтому чтобы понять, какой нужен, остаётся только гуглить.

Кроме того, для установки пакетов с github.com потребуется установить с официального репозитория

Джентельменский набор пакетов R зависит от сферы деятельности, но практически всем сталкивающимся с анализом данных пригодятся:

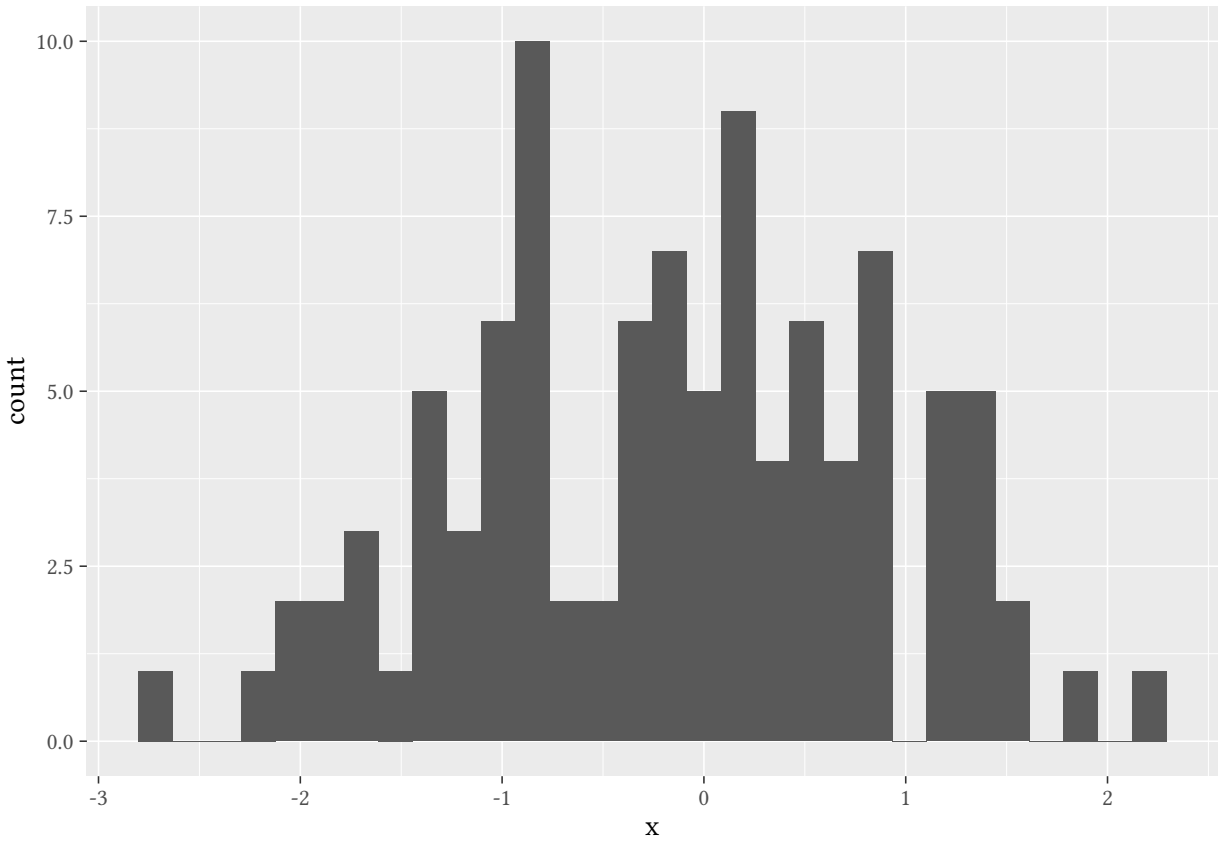
...

Очень часто пакеты R ошибочно называют библиотеками. Библиотека — это папка на жёстком диске компьютера, где хранятся пакеты.

Если пакет установлен, то можно воспользоваться его командами. Если из пакета нужна всего одна команда и один раз, то быстрее указать и нужный пакет, и нужную команду. Например, вызовем команду qplot из пакета ggplot2 и построим гистограмму для случайной выборки из 100 нормальных стандартных случайных величин:

```
x <- rnorm(100) # генерируем случайную выборку из 100 нормальных N(0;1) случайных величин
ggplot2::qplot(x)
```

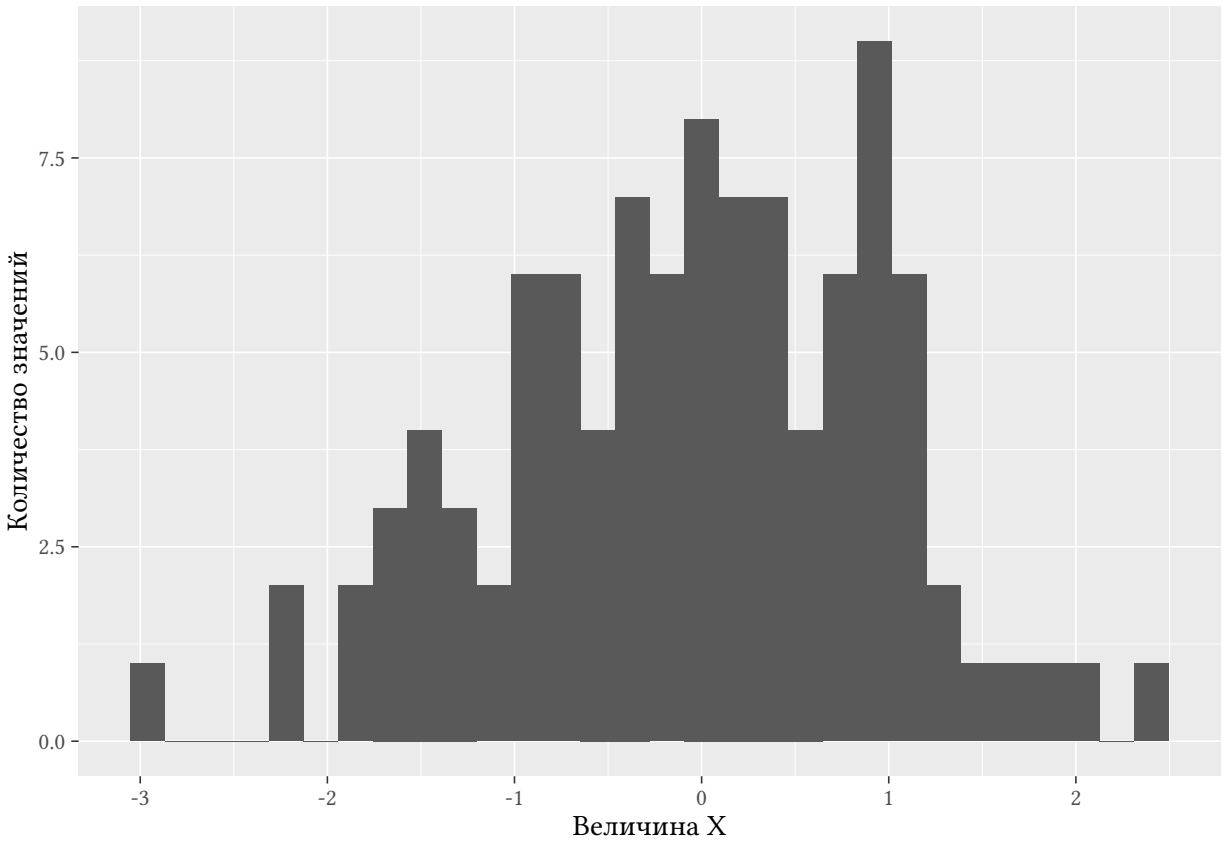
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



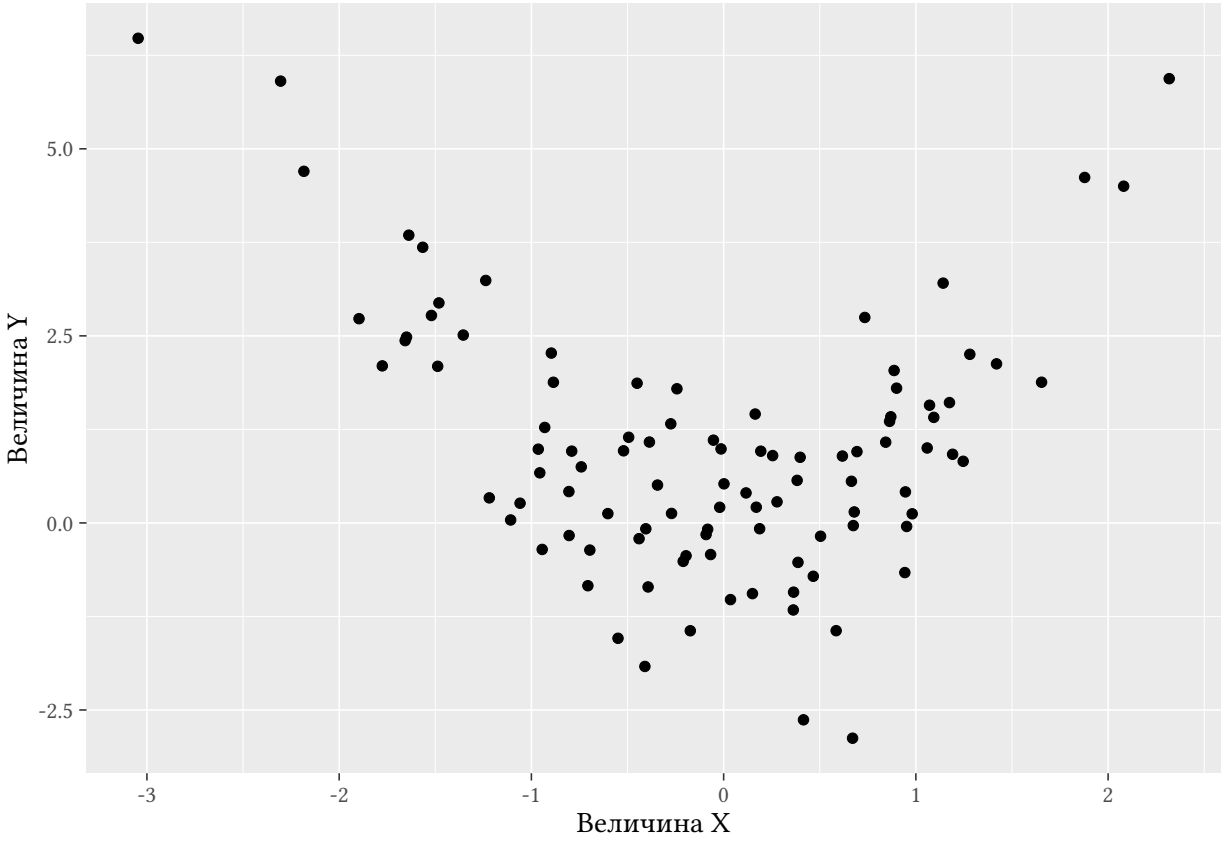
Если же ты хочешь использовать команды некоторого пакета много раз, то проще **подключить** пакет командой library(). При этом не надо будет каждый раз набирать название пакета и двойное двоеточие. Можно просто использовать команды из этого пакеты:

```
library(”ggplot2”) # подключаем пакет ‘ggplot2‘

x <- rnorm(100) # генерируем случайную выборку из 100 нормальных N(0;1) случайных величин
y <- x^2 + rnorm(100)

# строим гистограмму величины x:
qplot(x) + xlab(”Величина X”) + ylab(”Количество значений”)
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```




```
# строим диагрраму рассеяния величин x и y:
qplot(x, y) + xlab("Величина X") + ylab("Величина Y")
```



При подключении пакета, как и при его установке, не стоит пугаться сообщений красным шрифтом. Только явное слово Error говорит об ошибке. Кроме того, часто можно столкнуться с **предупреждением** (warning) о том, что пакет был создан для более новой версии R.

Warning message: package 'xxx' was built under R version 3.3.1

Это не страшно. Это означает лишь, что у разработчика пакета xxx установлена более свежая версия R, чем у тебя. Обновлять R на своём компьютере при каждом его мелком обновлении, пожалуй, неразумно, но раз в полгода стоит.

> Правила хорошего тона советуют подключать все нужные пакеты в начале скрипта.

1.5. Чтение и запись данных

Прежде всего неплохо бы знать, где лежит на жёстком диске файл с нужными данными. Напомню, что названия файлов и папок не должны содержать русским букв и пробелов!

У R есть понятие **рабочей папки** (working folder). В рабочей папке R ищет все требуемые файлы. Одно из простых решений — указать в качестве рабочей папки ту папку, где лежит нужный файл и далее прочитать его.

Допустим, нужный нам файл лежит в папке C:/project_A/data/. Тогда для установки рабочей папки достаточно выполнить команду:

```
setwd("C:/project_A/data/")
```

Вместо этой команды можно воспользоваться меню Rstudio: Session - Set working directory - Choose directory. Далее выбрать нужную папку и нажать Open.

После этого можно прочитать нужный нам файл. Начнём с пакета rio позволяющего импортировать данные практически любого типа. На самом деле авторы пакета rio просто объединили усилия многих разработчиков в единую команду. И получилось хорошо :)

Хочешь загрузить данные в формате .csv? Пожалуйста!

```
data <- rio::import("имя_файла.csv")
```

Хочешь загрузить данные в формате .xlsx? Пожалуйста!

```
data <- rio::import("имя_файла.xlsx")
```

Однако не всегда всё идёт гладко, поэтому остановимся подробнее на разных форматах данных.

1.6. Интернет-источники данных

Зачастую данные не обязательно даже сохранять. В R есть пакеты, дающие доступ к некоторым источникам данных в Интернете:

1. quandl
2. quantmod
3. WDI

   СССР — родина слонов!

Пакеты, дающие доступ к данным по России:

1. sophisthse sophist.hse.ru
2. cbr Центральный Банк России
3. datamos datamos.ru
4. finam.ru.

А эти источники ещё ждут желающих написать пакет для R:

1. gks.ru
2. open data gov ???

1.7. Стиль кода

R одинаково выполнит и команды

```
x<-6-7
y<--6+9
x -    y
```

и команды

```
x <- 6 - 7
y <- -6 + 9
x - y
```

Однако второй вариант гораздо приятнее для чтения. С тем, кто пишет код как в первом примере, Английская королева рядом не сядет! Чтобы иметь возможность войти в палату Лордов и Общин, тебе следует писать стильный код!

Если ты работаешь в команде, то руководствуйся тем стилем кода, который в ней принят. А для новичков я советую использовать стиль кода, которого придерживается Hadley Wickham, автор очень популярных пакетов R ggplot2 и dplyr:

1. После запятой всегда пиши пробел. Перед запятой — никогда:

```
paste0("Hi ", "guys!")
```

2. Знак присваивания <-, знаки арифметических действий (+, -, *), логические проверки (>, <, == и прочие) с двух сторон окружай одинарными пробелами.

```
x <- (3.5 + 7) * (2.8 - 9)
```

3. Открывающую фигурную скобку оставляй на старой строке, а закрывающую — ставь на новую:

```
if (x == y) {
  message("Variables x and y are equal.")
}
```

В Rstudio можно включить автоматическую проверку стиля кода в Tools - Global options - Code - Diagnostics. Настоящие сэры и истинные леди в разделе Diagnostics могут проставить все галочки.

1.8. Две записи функций

Мы все привыкли к тому, что домохозяйки пишут рецепт в естественном порядке, а математики функции — в обратном. Сравни:

> Возьмите пепел перьев чёрного петуха
>
> Добавьте печень дракона
>
> Варите на медленном огне 2 дня

и

$$\cos(\sin(|x|))$$

У домохозяек порядок изложения совпадает с порядком действий. У математиков сначала написано про косинус, но считается он в самом конце.

Похоже Лёнька Эйлер и Алёшка Клеро

фото

введя обозначение $f(x)$ отделили математиков от домохозяек и, вероятно, пустили математику по ложному пути. Было бы гораздо удобнее, если бы в математике функции также записывали в естественном порядке! Но обозначение $f(x)$ мы впитали с молоком матери, уже вряд ли что исправишь.

R позволяет использовать обе традиции обозначени.

Традиция Эйлера-Клеро:

```
cos(sin(abs(10)))
```

```
## [1] 0.8556344
```

Для того, чтобы иметь возможность писать операции в естественном порядке, подключаем пакет dplyr:

```
library("dplyr")
```

И теперь в традиции лучших кулинарных рецептов можно написать

```
10 %>% abs() %>% sin() %>% cos()
```

```
## [1] 0.8556344
```

Оператор %>% называется трубочкой (pipe). (? канал) По первому впечатлению кажется, что эти трубочки долго писать. Но стоит к ним привыкнуть и ощущаешь, что они безумно удобны для сложных операций!

1.9. Манипуляции с данными

(здесь про типы данных)

1.10. Графики

...

1.11. У меня ошибка!

> Шеф! Всё пропало! Гипс снимают, клиент уезжает!

поговори с уточкой, посиди у озера

1.12. Ресурсы по R

Ты хочешь скачать научную статью или книжку бесплатно?

- Научные книги можно найти на gen.lib.rus.ec.
- Научные статьи можно скачать на sci-hub.cc. Есть даже бот @scihubot для Telegram, которые вышлет в ответ на запрос полный текст статьи.

Глава 2

Друзья R

ывава ыва ыва test

2.1. Рабочий процесс

...

2.2. Контроль версий

...

2.3. Латех

...

2.4. Маркдаун

...

2.5. Воспроизводимые исследования

...

2.6. Написание своего пакета

...

2.7. Вычисления в облаке

...

2.8. Презентации

...

2.9. Про эту книжку

Разбить на большее количество глав!!!

Что-то убрать? чтобы была ещё одна книжка? :)

Общие принципы:

1. Неформальный стиль, на “ты”
2. Больше картинок. Лицензия?
3. Больше гипер-ссылок.
4. Буква ё обязательна.

Пакет bookdown с помощью которого написана эта книжка ещё немного сыроват. В процессе работы я обнаружил, что:

1. Порой помогает удаление промежуточных файлов и компиляция заново.
2. После неанглоязычного названия главы обязательно должна идти метка {#label}.
3. Каждый .Rmd файл содержит только одну главу. Глава обозначается заголовком первого уровня #.
4. Сослаться на главу или подраздел можно с помощью \@ref(label).
5. Сослаться на источник литературы можно с помощью [@reference]
6. Автодобавление пакетов

глючит на ”M”uller

6. Для создания MOBI книг:

```
brew update
brew cask install calibre
brew cask install kindlegen
```

В предисловии

```
bookdown::kindlegen:
  epub: _book/r_manual.epub
```

ругается

или

```
bookdown::calibre:
  input: _book/r_manual.epub
  output: _book/r_manual.mobi
```

ругается

Если надо изобразить yaml в чанке кода, пока пишу, что он bash

7. заставляем травис работать

Создаём новый токен на github: кликнуть по иконке пользователя, далее settings - token - generate new token.

```
sudo gem install travis
travis encrypt GITHUB_TOKEN=[token from githum]
```

Именно переменная GITHUB_TOKEN должна использоваться для того, чтобы обращаться к гитхабу, т.е.

Добавляем получившуюся закодированную строку в .travis.yml. Забавно, что похоже используется рандомный ключ для шифрования и поэтому зашифрованная строка каждый раз выходит разной.

Можно не шифровать её, а добавить в travis-ci.org/user/repo/settings

Для быстрой компиляции добавляем в .travis.yml указание, что мы будем использовать готовые бинарные пакеты R:

```
r_binary_packages:
  - ggplot2
  - dplyr
  - rio
```

Черт его знает? всё равно компилирует?

8. Красный шрифт

по мотивам http://stackoverflow.com/questions/29067541

```
colFmt <- function(x, color) {
  outputFormat <- opts_knit$get("rmarkdown.pandoc.to")
  if (outputFormat == "latex") {
    result <- paste0("\\textcolor{", color, "}{", x, "}")
  } else if (outputFormat %in% c("html", "epub")) {
    result <- paste0("<font color='", color, "'>", x, "</font>")
  } else {
    result <- x
  }
  return(result)
}
```

Then you can use it inline like this: MY RED TEXT

9. Файл _output.yaml это просто output: кусок из yaml-части файла index.Rmd. Поэтому можно внести _output.yaml обратно в index.Rmd, чтобы все настройки были в одном месте.

Формально _output.yaml действует на все .Rmd документы в папке, но что там будет кроме учебника в целом. Разве что отдельные главы компилировать :)

Глава 3

Статистика и не только

ываываыв ываываыв

3.1. Генерирование случайных величин

Для решения задач по теории вероятностей или исследования свойств статистических алгоритмов может потребоваться сгененировать случайную выборку из заданного закона распределения.

Генерируем 10 равномерных на отрезке $[4; 10.5]$ случайных величин:

```
x <- runif(10, min = 4, max = 10.5)
```

Генерируем 10 нормальных $N(2; 9)$ случайных величин с математическим ожиданием 2 и дисперсией $9 = 3^2$:

```
x <- rnorm(10, mean = 2, sd = 3)
```

Например, с помощью симуляций легко оценить математическое ожидание $E(1/X)$, где $X \sim N(2; 9)$. Для этого мы вспомним Закон Больших Чисел. Он говорит, что арифметическое среднее по большой выборке стремится по вероятности и почти наверное к математическому ожиданию. Поэтому мы просто сгенерируем большую выборку в миллион наблюдений:

```
n_obs <- 10^6
x <- rnorm(n_obs, mean = 2, sd = 3)
mean(1/x)
```

```
## [1] 1.073921
```

Также легко оценить многие вероятности. Например, оценим вероятность $P(X_1 + X_2 + X_3^2 > 5)$, где величины X_i независимы и одинаково распределены $X_i \sim U[0; 2]$:

```
n_obs <- 10^6
x_1 <- runif(n_obs, min = 0, max = 2)
x_2 <- runif(n_obs, min = 0, max = 2)
x_3 <- runif(n_obs, min = 0, max = 2)
success <- x_1 + x_2 + x_3^2 > 5
sum(success) / n_obs
```

```
## [1] 0.148711
```

Здесь вектор success будет содержать значение TRUE там, где условие x_1 + x_2 + x_3^2 > 5 выполнено, и FALSE там, где условие не выполнено. При сложении командой sum() каждое TRUE будет посчитано как единица, а каждое FALSE как ноль. Поэтому sum(success) даст количество раз, когда условие x_1 + x_2 + x_3^2 > 5 выполнено.

... тут ещё несколько распределений

... помимо генерации случайных чисел нарисуем функции плотности и распределения

Если выполнить команду rnorm(10, mean = 2, sd = 3) на двух разных компьютерах или два раза на одном и том же, то результат будет разный. Не зря же они случайные :) Однако генерирование случайных величин никак не противоречит идее абсолютно точной воспроизводимости исследований. Для того, чтобы получились одинаковые результаты, необходимо синхронизировать генераторы случайных чисел на этих двух компьютерах. Делается это путём задания **зерна генератора** (seed). Зерно также называют стартовым значением. В качестве зерна подойдёт любое целое число.

И в результате запуска кода

```
set.seed(42)
rnorm(1, mean = 2, sd = 3)
```

```
## [1] 6.112875
```

все компьютеры выведут число 6.112875.

Если код содержит генерерирование случайных чисел, то для воспроизводимости результатов необходимо задавать зерно генератора

3.2. Базовые статистические тесты

...

3.3. Множественная регрессия

...

3.4. Квантильная регрессия

Незаслуженно забытой оказывается квантильная регрессия. Коэнкер (ссылка) утверждает, что развитие эконометрики началось именно с квантильной регрессии. Для оценок квантильной регрессии не существует формул в явном виде, поэтому она проиграла классической регрессии среднего с формулой $\hat{\beta} = (X'X)^{-1}X'y$. Сейчас компьютер позволяет начихать на отсутствие явных формул :)

...

3.5. Инструментальные переменные

...

3.6. Гетероскедастичность

...

3.7. Работа с качественными переменными

...

3.8. Логит и пробит с визуализацией

...

3.9. Метод главных компонент

...

3.10. Мультиколлинеарность

...

3.11. LASSO

...

3.12. Метод максимального правдоподобия

...

3.13. Метод опорных векторов

...

3.14. Случайный лес

...

3.15. Экспоненциальное сглаживание

...

3.16. ARMA модели

...

3.17. GARCH

...

3.18. VAR и BVAR

...

Я не верю в пользу от структурных BVAR, поэтому их здесь нет :)

3.19. Панельные данные

...

3.20. Байесовский подход: первые шаги

...

3.21. Байесовский подход: STAN

...

3.22. Карты

Где карта, Билли?

3.23. Дифференциальные уравнения

...

3.24. Задачи оптимизации

...