O'REILLY®

R for Data Science

IMPORT, TIDY, TRANSFORM, VISUALIZE, AND MODEL DATA

Hadley Wickham &
Garrett Grolemund

O'REILLY®

R for Data Science

Learn how to use R to turn raw data into insight, knowledge, and understanding. This book introduces you to R, RStudio, and the tidyverse, a collection of R packages designed to work together to make data science fast, fluent, and fun. Suitable for readers with no previous programming experience, *R for Data Science* is designed to get you doing data science as quickly as possible.

Authors Hadley Wickham and Garrett Grolemund guide you through the steps of importing, wrangling, exploring, and modeling your data and communicating the results. You'll get a complete, big-picture understanding of the data science cycle, along with basic tools you need to manage the details.

You'll learn how to:

- **Wrangle**—transform your datasets into a form convenient for analysis
- **Program**—learn powerful R tools for solving data problems with greater clarity and ease
- **Explore**—examine your data, generate hypotheses, and quickly test them
- **Model**—provide a low-dimensional summary that captures true "signals" in your dataset
- **Communicate**—learn R Markdown for integrating prose, code, and results

"Hadley Wickham is a legend in the data science field for having invented a completely new way of doing data analysis that no one had thought of before. This new book with Garrett Grolemund codifies this novel approach and will serve as the Bible for a generation of data analysts."

—Roger D. Peng
Professor of Biostatistics, Johns Hopkins Bloomberg School of Public Health

Hadley Wickham is Chief Scientist at RStudio and a member of the R Foundation. He builds tools (both computational and cognitive) that make data science easier, faster, and more fun. Learn more on his website, http://hadley.nz.

Garrett Grolemund is a statistician, teacher, and Master Instructor at RStudio. He is the author of *Hands-On Programming with R* (O'Reilly). Many of Garrett's instructional videos are available on oreilly.com/safari.

DATA ANALYSIS/STATISTICAL SOFTWARE

US $39.99 **CAN $45.99**
ISBN: 978-1-491-91039-9

Twitter: @oreillymedia
facebook.com/oreilly

R for Data Science

Import, Tidy, Transform, Visualize, and Model Data

Hadley Wickham and Garrett Grolemund

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

R for Data Science

by Hadley Wickham and Garrett Grolemund

Printed in Canada.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com/safari*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editors: Marie Beaugureau and Mike Loukides
Production Editor: Nicholas Adams
Copyeditor: Kim Cofer
Proofreader: Charles Roumeliotis

Indexer: Wendy Catalano
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

December 2016: First Edition

Revision History for the First Edition

2016-12-06: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781491910399* for release details.

978-1-491-91039-9

[TI]

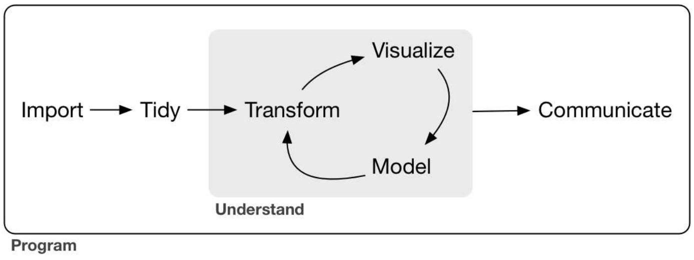
Table of Contents

Preface

Data science is an exciting discipline that allows you to turn raw data into understanding, insight, and knowledge. The goal of *R for Data Science* is to help you learn the most important tools in R that will allow you to do data science. After reading this book, you'll have the tools to tackle a wide variety of data science challenges, using the best parts of R.

What You Will Learn

Data science is a huge field, and there's no way you can master it by reading a single book. The goal of this book is to give you a solid foundation in the most important tools. Our model of the tools needed in a typical data science project looks something like this:



First you must *import* your data into R. This typically means that you take data stored in a file, database, or web API, and load it into a data frame in R. If you can't get your data into R, you can't do data science on it!

Once you've imported your data, it is a good idea to *tidy* it. Tidying your data means storing it in a consistent form that matches the semantics of the dataset with the way it is stored. In brief, when your data is tidy, each column is a variable, and each row is an observation. Tidy data is important because the consistent structure lets you focus your struggle on questions about the data, not fighting to get the data into the right form for different functions.

Once you have tidy data, a common first step is to *transform* it. Transformation includes narrowing in on observations of interest (like all people in one city, or all data from the last year), creating new variables that are functions of existing variables (like computing velocity from speed and time), and calculating a set of summary statistics (like counts or means). Together, tidying and transforming are called *wrangling*, because getting your data in a form that's natural to work with often feels like a fight!

Once you have tidy data with the variables you need, there are two main engines of knowledge generation: visualization and modeling. These have complementary strengths and weaknesses so any real analysis will iterate between them many times.

Visualization is a fundamentally human activity. A good visualization will show you things that you did not expect, or raise new questions about the data. A good visualization might also hint that you're asking the wrong question, or you need to collect different data. Visualizations can surprise you, but don't scale particularly well because they require a human to interpret them.

Models are complementary tools to visualization. Once you have made your questions sufficiently precise, you can use a model to answer them. Models are a fundamentally mathematical or computational tool, so they generally scale well. Even when they don't, it's usually cheaper to buy more computers than it is to buy more brains! But every model makes assumptions, and by its very nature a model cannot question its own assumptions. That means a model cannot fundamentally surprise you.

The last step of data science is *communication*, an absolutely critical part of any data analysis project. It doesn't matter how well your models and visualization have led you to understand the data unless you can also communicate your results to others.

Surrounding all these tools is *programming*. Programming is a cross-cutting tool that you use in every part of the project. You don't need to be an expert programmer to be a data scientist, but learning more about programming pays off because becoming a better programmer allows you to automate common tasks, and solve new problems with greater ease.

You'll use these tools in every data science project, but for most projects they're not enough. There's a rough 80-20 rule at play; you can tackle about 80% of every project using the tools that you'll learn in this book, but you'll need other tools to tackle the remaining 20%. Throughout this book we'll point you to resources where you can learn more.

How This Book Is Organized

The previous description of the tools of data science is organized roughly according to the order in which you use them in an analysis (although of course you'll iterate through them multiple times). In our experience, however, this is not the best way to learn them:

- Starting with data ingest and tidying is suboptimal because 80% of the time it's routine and boring, and the other 20% of the time it's weird and frustrating. That's a bad place to start learning a new subject! Instead, we'll start with visualization and transformation of data that's already been imported and tidied. That way, when you ingest and tidy your own data, your motivation will stay high because you know the pain is worth it.
- Some topics are best explained with other tools. For example, we believe that it's easier to understand how models work if you already know about visualization, tidy data, and programming.
- Programming tools are not necessarily interesting in their own right, but do allow you to tackle considerably more challenging problems. We'll give you a selection of programming tools in the middle of the book, and then you'll see they can combine with the data science tools to tackle interesting modeling problems.

Within each chapter, we try to stick to a similar pattern: start with some motivating examples so you can see the bigger picture, and then dive into the details. Each section of the book is paired with exercises to help you practice what you've learned. While it's tempting to skip the exercises, there's no better way to learn than practicing on real problems.

What You Won't Learn

There are some important topics that this book doesn't cover. We believe it's important to stay ruthlessly focused on the essentials so you can get up and running as quickly as possible. That means this book can't cover every important topic.

Big Data

This book proudly focuses on small, in-memory datasets. This is the right place to start because you can't tackle big data unless you have experience with small data. The tools you learn in this book will easily handle hundreds of megabytes of data, and with a little care you can typically use them to work with 1–2 Gb of data. If you're routinely working with larger data (10–100 Gb, say), you should learn more about data.table. This book doesn't teach data.table because it has a very concise interface, which makes it harder to learn since it offers fewer linguistic cues. But if you're working with large data, the performance payoff is worth the extra effort required to learn it.

If your data is bigger than this, carefully consider if your big data problem might actually be a small data problem in disguise. While the complete data might be big, often the data needed to answer a specific question is small. You might be able to find a subset, subsample, or summary that fits in memory and still allows you to answer the question that you're interested in. The challenge here is finding the right small data, which often requires a lot of iteration.

Another possibility is that your big data problem is actually a large number of small data problems. Each individual problem might fit in memory, but you have millions of them. For example, you might want to fit a model to each person in your dataset. That would be trivial if you had just 10 or 100 people, but instead you have a million. Fortunately each problem is independent of the others (a setup that is sometimes called embarrassingly parallel), so you just need a system (like Hadoop or Spark) that allows you to send different datasets to different computers for processing. Once you've figured out how to answer the question for a single subset using the tools described in this book, you learn new tools like sparklyr, rhipe, and ddr to solve it for the full dataset.

Python, Julia, and Friends

In this book, you won't learn anything about Python, Julia, or any other programming language useful for data science. This isn't because we think these tools are bad. They're not! And in practice, most data science teams use a mix of languages, often at least R and Python.

However, we strongly believe that it's best to master one tool at a time. You will get better faster if you dive deep, rather than spreading yourself thinly over many topics. This doesn't mean you should only know one thing, just that you'll generally learn faster if you stick to one thing at a time. You should strive to learn new things throughout your career, but make sure your understanding is solid before you move on to the next interesting thing.

We think R is a great place to start your data science journey because it is an environment designed from the ground up to support data science. R is not just a programming language, but it is also an interactive environment for doing data science. To support interaction, R is a much more flexible language than many of its peers. This flexibility comes with its downsides, but the big upside is how easy it is to evolve tailored grammars for specific parts of the data science process. These mini languages help you think about problems as a data scientist, while supporting fluent interaction between your brain and the computer.

Nonrectangular Data

This book focuses exclusively on rectangular data: collections of values that are each associated with a variable and an observation. There are lots of datasets that do not naturally fit in this paradigm: including images, sounds, trees, and text. But rectangular data frames are extremely common in science and industry, and we believe that they're a great place to start your data science journey.

Hypothesis Confirmation

It's possible to divide data analysis into two camps: hypothesis generation and hypothesis confirmation (sometimes called confirmatory analysis). The focus of this book is unabashedly on hypothesis generation, or data exploration. Here you'll look deeply at the data and, in combination with your subject knowledge, generate many interesting hypotheses to help explain why the data behaves the way it does. You evaluate the hypotheses informally, using your skepticism to challenge the data in multiple ways.

The complement of hypothesis generation is hypothesis confirmation. Hypothesis confirmation is hard for two reasons:

- You need a precise mathematical model in order to generate falsifiable predictions. This often requires considerable statistical sophistication.
- You can only use an observation once to confirm a hypothesis. As soon as you use it more than once you're back to doing exploratory analysis. This means to do hypothesis confirmation you need to "preregister" (write out in advance) your analysis plan, and not deviate from it even when you have seen the data. We'll talk a little about some strategies you can use to make this easier in Part IV.

It's common to think about modeling as a tool for hypothesis confirmation, and visualization as a tool for hypothesis generation. But that's a false dichotomy: models are often used for exploration, and with a little care you can use visualization for confirmation. The key difference is how often you look at each observation: if you look only once, it's confirmation; if you look more than once, it's exploration.

Prerequisites

We've made a few assumptions about what you already know in order to get the most out of this book. You should be generally numerically literate, and it's helpful if you have some programming experience already. If you've never programmed before, you might find *Hands-On Programming with R* by Garrett to be a useful adjunct to this book.

There are four things you need to run the code in this book: R, RStudio, a collection of R packages called the *tidyverse*, and a handful of other packages. Packages are the fundamental units of reproducible R code. They include reusable functions, the documentation that describes how to use them, and sample data.

R

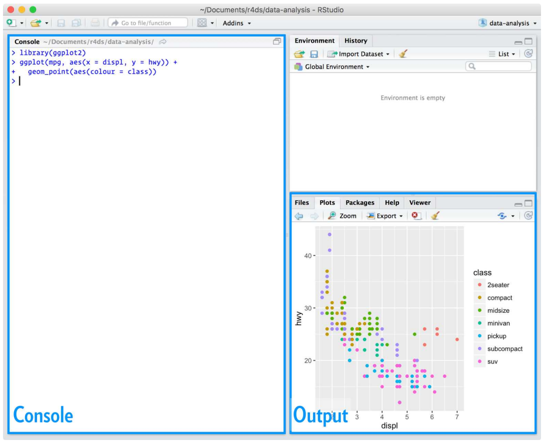
To download R, go to CRAN, the *comprehensive R archive network*. CRAN is composed of a set of mirror servers distributed around the world and is used to distribute R and R packages. Don't try and pick a mirror that's close to you: instead use the cloud mirror, *https://cloud.r-project.org*, which automatically figures it out for you.

A new major version of R comes out once a year, and there are 2–3 minor releases each year. It's a good idea to update regularly. Upgrading can be a bit of a hassle, especially for major versions, which require you to reinstall all your packages, but putting it off only makes it worse.

RStudio

RStudio is an integrated development environment, or IDE, for R programming. Download and install it from *http://www.rstudio.com/download*. RStudio is updated a couple of times a year. When a new version is available, RStudio will let you know. It's a good idea to upgrade regularly so you can take advantage of the latest and greatest features. For this book, make sure you have RStudio 1.0.0.

When you start RStudio, you'll see two key regions in the interface:



```
> library(ggplot2)
> ggplot(mpg, aes(x = displ, y = hwy)) +
+   geom_point(aes(colour = class))
> |
```

For now, all you need to know is that you type R code in the console pane, and press Enter to run it. You'll learn more as we go along!

The Tidyverse

You'll also need to install some R packages. An R *package* is a collection of functions, data, and documentation that extends the capabilities of base R. Using packages is key to the successful use of R. The majority of the packages that you will learn in this book are part of the so-called tidyverse. The packages in the tidyverse share a common philosophy of data and R programming, and are designed to work together naturally.

You can install the complete tidyverse with a single line of code:

```
install.packages("tidyverse")
```

On your own computer, type that line of code in the console, and then press Enter to run it. R will download the packages from CRAN and install them onto your computer. If you have problems installing, make sure that you are connected to the internet, and that *https://cloud.r-project.org/* isn't blocked by your firewall or proxy.

You will not be able to use the functions, objects, and help files in a package until you load it with `library()`. Once you have installed a package, you can load it with the `library()` function:

```
library(tidyverse)
#> Loading tidyverse: ggplot2
#> Loading tidyverse: tibble
#> Loading tidyverse: tidyr
#> Loading tidyverse: readr
#> Loading tidyverse: purrr
#> Loading tidyverse: dplyr
#> Conflicts with tidy packages --------------------------------
#> filter(): dplyr, stats
#> lag():    dplyr, stats
```

This tells you that tidyverse is loading the **ggplot2**, **tibble**, **tidyr**, **readr**, **purrr**, and **dplyr** packages. These are considered to be the *core* of the tidyverse because you'll use them in almost every analysis.

Packages in the tidyverse change fairly frequently. You can see if updates are available, and optionally install them, by running `tidyverse_update()`.

Other Packages

There are many other excellent packages that are not part of the tidyverse, because they solve problems in a different domain, or are designed with a different set of underlying principles. This doesn't make them better or worse, just different. In other words, the complement to the tidyverse is not the messyverse, but many other universes of interrelated packages. As you tackle more data science projects with R, you'll learn new packages and new ways of thinking about data.

In this book we'll use three data packages from outside the tidyverse:

```
install.packages(c("nycflights13", "gapminder", "Lahman"))
```

These packages provide data on airline flights, world development, and baseball that we'll use to illustrate key data science ideas.

Running R Code

The previous section showed you a couple of examples of running R code. Code in the book looks like this:

```
1 + 2
#> [1] 3
```

If you run the same code in your local console, it will look like this:

```
> 1 + 2
[1] 3
```

There are two main differences. In your console, you type after the `>`, called the *prompt*; we don't show the prompt in the book. In the book, output is commented out with `#>`; in your console it appears directly after your code. These two differences mean that if you're working with an electronic version of the book, you can easily copy code out of the book and into the console.

Throughout the book we use a consistent set of conventions to refer to code:

- Functions are in a code font and followed by parentheses, like `sum()` or `mean()`.
- Other R objects (like data or function arguments) are in a code font, without parentheses, like `flights` or `x`.

- If we want to make it clear what package an object comes from, we'll use the package name followed by two colons, like `dplyr::mutate()` or `nycflights13::flights`. This is also valid R code.

Getting Help and Learning More

This book is not an island; there is no single resource that will allow you to master R. As you start to apply the techniques described in this book to your own data you will soon find questions that I do not answer. This section describes a few tips on how to get help, and to help you keep learning.

If you get stuck, start with Google. Typically, adding "R" to a query is enough to restrict it to relevant results: if the search isn't useful, it often means that there aren't any R-specific results available. Google is particularly useful for error messages. If you get an error message and you have no idea what it means, try googling it! Chances are that someone else has been confused by it in the past, and there will be help somewhere on the web. (If the error message isn't in English, run `Sys.setenv(LANGUAGE = "en")` and re-run the code; you're more likely to find help for English error messages.)

If Google doesn't help, try stackoverflow. Start by spending a little time searching for an existing answer; including `[R]` restricts your search to questions and answers that use R. If you don't find anything useful, prepare a minimal reproducible example or **reprex**. A good reprex makes it easier for other people to help you, and often you'll figure out the problem yourself in the course of making it.

There are three things you need to include to make your example reproducible: required packages, data, and code:

- *Packages* should be loaded at the top of the script, so it's easy to see which ones the example needs. This is a good time to check that you're using the latest version of each package; it's possible you've discovered a bug that's been fixed since you installed the package. For packages in the tidyverse, the easiest way to check is to run `tidyverse_update()`.
- The easiest way to include *data* in a question is to use `dput()` to generate the R code to re-create it. For example, to re-create the `mtcars` dataset in R, I'd perform the following steps: