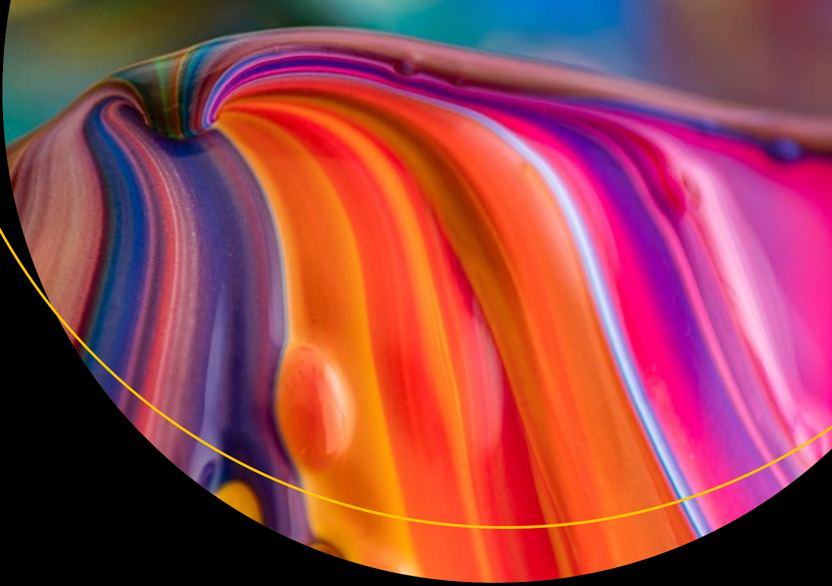


Beginning Data Science in R 4

Data Analysis, Visualization, and Modelling for the Data Scientist

Second Edition

Thomas Mailund

Apress®

Beginning Data Science in R 4

Data Analysis, Visualization, and Modelling for the Data Scientist

Second Edition

Thomas Mailund

Apress®

Beginning Data Science in R 4: Data Analysis, Visualization, and Modelling for the Data Scientist

Thomas Mailund
Aarhus, Denmark

ISBN-13 (pbk): 978-1-4842-8154-3 ISBN-13 (electronic): 978-1-4842-8155-0
https://doi.org/10.1007/978-1-4842-8155-0

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Steve Anglin
Development Editor: James Markham
Coordinating Editor: Mark Powers

Cover designed by eStudioCalamar

Cover image by Pixabay (www.pixabay.com)

Printed on acid-free paper

Table of Contents

About the Author

Thomas Mailund is an associate professor in bioinformatics at Aarhus University, Denmark. His background is in math and computer science, but for the last decade, his main focus has been on genetics and evolutionary studies, particularly comparative genomics, speciation, and gene flow between emerging species.

About the Technical Reviewer

Jon Westfall is an associate professor of psychology at Delta State University. He has authored *Set Up and Manage Your Virtual Private Server*, *Practical R 4*, *Beginning Android Web Apps Development*, *Windows Phone 7 Made Simple*, and several works of fiction including *One in the Same*, *Mandate*, and *Franklin: The Ghost Who Successfully Evicted Hipsters from His Home and Other Short Stories*. He lives in Cleveland, Mississippi, with his wife.

Acknowledgments

I would like to thank Asger Hobolth for many valuable comments on earlier versions of this manuscript that helped me improve the writing and the presentation of the material.

Introduction

Welcome to *Beginning Data Science in R 4*. I wrote this book from a set of lecture notes for two classes I taught a few years back, "Data Science: Visualization and Analysis" and "Data Science: Software Development and Testing." The book is written to fit the structure of these classes, where each class consists of seven weeks of lectures followed by project work. This means that the book's first half consists of eight chapters with core material, where the first seven focus on data analysis and the eighth is an example of a data analysis project. The data analysis chapters are followed by seven chapters on developing reusable software for data science and then a second project that ties the software development chapters together. At the end of the book, you should have a good sense of what data science can be, both as a field covering analysis and developing new methods and reusable software products.

What Is Data Science?

That is a difficult question. I don't know if it is easy to find someone who is entirely sure what data science is, but I am pretty sure that it would be difficult to find two people without having three opinions about it. It is undoubtedly a popular buzzword, and everyone wants to hire data scientists these days, so data science skills are helpful to have on the CV. But what is it?

Since I can't give you an agreed-upon definition, I will just give you my own: data science is the science of learning from data.

This definition is very broad—almost too broad to be useful. I realize this. But then, I think data science is an incredibly general field. I don't have a problem with that. Of course, you could argue that any science is all about getting information out of data, and you might be right. However, I would say that there is more to science than just transforming raw data into useful information. The sciences focus on answering specific questions about the world, while data science focuses on how to manipulate data efficiently and effectively. The primary focus is not which questions to ask of the data but how we can answer them, whatever they may be. It is more like computer science and mathematics than it is like natural sciences, in this way. It isn't so much about studying the natural world as it is about computing efficiently on data and learning patterns from the data.

Included in data science is also the design of experiments. With the right data, we can address the questions in which we are interested. This can be difficult with a poor design of experiments or a poor choice of which data we gather. Study design might be the most critical aspect of data science but is not the topic of this book. In this book, I focus on the analysis of data, once gathered.

Computer science is mainly the study of computations, hinted at in the name, but is a bit broader. It is also about representing and manipulating data. The name "computer science" focuses on computation, while "data science" emphasizes data. But of course, the fields overlap. If you are writing a sorting algorithm, are you then focusing on the computation or the data? Is that even a meaningful question to ask?

There is considerable overlap between computer science and data science, and, naturally, the skill sets you need overlap as well. To efficiently manipulate data, you need the tools for doing that, so computer programming skills are a must, and some knowledge about algorithms and data structures usually is as well. For data science, though, the focus is always on the data. A data analysis project focuses on how the data flows from its raw form through various manipulations until it is summarized in some helpful way. Although the difference can be subtle, the focus is not on what operations a program does during the analysis but how the data flows and is transformed. It is also focused on why we do certain data transformations, what purpose those changes serve, and how they help us gain knowledge about the data. It is as much about deciding what to do with the data as it is about how to do it efficiently.

Statistics is, of course, also closely related to data science. So closely linked that many consider data science as nothing more than a fancy word for statistics that looks slightly more modern and sexy. I can't say that I strongly disagree with this—data science does sound hotter than statistics—but just as data science is slightly different from computer science, data science is also somewhat different from statistics. Only, perhaps, somewhat less so than computer science is.

A large part of doing statistics is building mathematical models for your data and fitting the models to the data to learn about the data in this way. That is also what we do in data science. As long as the focus is on the data, I am happy to call statistics data science. But suppose the focus changes to the models and the mathematics. In that case, we are drifting away from data science into something else—just as if the focus shifts from the data to computations, we are straying from data science to computer science.

Data science is also related to machine learning and artificial intelligence—and again, there are huge overlaps. Perhaps not surprising since something like machine learning has its home both in computer science and statistics; if it focuses on data analysis, it is also at home in data science. To be honest, it has never been clear to me when a mathematical model changes from being a plain old statistical model to becoming machine learning anyway.

For this book, we are just going to go with my definition, and, as long as we are focusing on analyzing data, we will call it data science.

Prerequisites for Reading This Book

For the first eight chapters in this book, the focus is on data analysis and not programming. For those eight chapters, I do not assume a detailed familiarity with software design, algorithms, data structures, etc. I do not expect you to have any experience with the R programming language either. However, I assume that you have had some experience with programming, mathematical modelling, and statistics.

Programming R can be quite tricky at times if you are familiar with scripting languages or object-oriented languages. R is a functional language that does not allow you to modify data. While it does have systems for object-oriented programming, it handles this programming paradigm very differently from languages you are likely to have seen, such as Java or Python.

For the data analysis part of this book, the first eight chapters, we will only use R for very straightforward programming tasks, so none of this should pose a problem. We will have to write simple scripts for manipulating and summarizing data, so you should be familiar with how to write basic expressions like function calls, if statements, loops, and such—these things you will have to be comfortable with. I will introduce every such construction in the book when we need them to let you see how they are written in R, but I will not spend much time explaining them. Mostly, I will expect you to be able to pick it up from examples.

Similarly, I do not expect you to already know how to fit data and compare models in R. I do assume that you have had enough introduction to statistics to be comfortable with basic terms like parameter estimation, model fitting, explanatory and response variables, and model comparison. If not, I expect you to at least be able to pick up what we are talking about when you need to.

I won't expect you to know a lot about statistics and programming, but this isn't "Data Science for Dummies," so I expect you to figure out examples without me explaining everything in detail.

After the first seven chapters is a short description of a data analysis project that one of my students did for my class the first time I held it. It shows how such a project could look, but I suggest that you do not wait until you have finished the first seven chapters to start doing such analysis yourself. To get the most benefit out of reading this book, you should continuously apply what you learn. Already when you begin reading, I suggest that you find a data set that you would be interested in finding out more about and then apply what you learn in each chapter to that data.

For the following eight chapters of the book, the focus is on programming. To read this part, you should be familiar with object-oriented programming—I will explain how we handle it in R and how it differs from languages such as Python, Java, or C++. Still, I will expect you to be familiar with terms such as class hierarchies, inheritance, and polymorphic methods. I will not expect you to be already familiar with functional programming (but if you are, there should still be plenty to learn in those chapters if you are not already familiar with R programming). The final chapter is yet another project description.

Plan for the Book

In the book, we will cover basic data manipulation:

- Filtering and selecting relevant data
- Transforming data into shapes readily analyzable
- Summarizing data
- Visualization data in informative ways both for exploring data and presenting results
- Model building

These are the critical aspects of doing analysis in data science. After this, we will cover how to develop R code that is reusable and works well with existing packages and that is easy to extend, and we will see how to build new R packages that other people will be able to use in their projects. These are the essential skills you will need to develop your own methods and share them with the world.

R is one of the most popular (and open source) data analysis programming languages around at the moment. Of course, popularity doesn't imply quality. Still, because R is so popular, it has a rich ecosystem of extensions (called "packages" in R) for just about any kind of analysis you could be interested in. People who develop statistical methods often implement them as R packages, so you can usually get the state-of-the-art techniques very easily in R. The popularity also means that there is a large community of people who can help if you have problems. Most problems you run into can be solved with a few minutes on Google or Stack Overflow because you are unlikely to be the first to run into any particular issue. There are also plenty of online tutorials for learning more about R and specialized packages. And there are plenty of books you can buy if you want to learn more.

Data Analysis and Visualization

The topics focusing on data analysis and visualization I cover in the first eight chapters:

1. Introduction to R Programming: In this chapter, we learn how to work with data and write data pipelines.
2. Reproducible Analysis: In this chapter, we find out how to integrate documentation and analysis in a single document and how to use such documents to produce reproducible research.
3. Data Manipulation: In this chapter, we learn how to import data, tidy up data, transform, and compute summaries from data.
4. Visualizing Data: In this chapter, we learn how to make plots for exploring data features and presenting data features and analysis results.
5. Working with Large Data Sets: In this chapter, we see how to deal with data where the number of observations makes our usual approaches too slow.
6. Supervised Learning: In this chapter, we learn how to train models when we have data sets with known classes or regression values.
7. Unsupervised Learning: In this chapter, we learn how to search for patterns we are not aware of in data.

8. Project 1: Hitting the Bottle: Following these chapters is the first project, an analysis of physicochemical features of wine, where we see the various techniques in use.

Software Development

The next nine chapters cover software and package development:

1. Deeper into R Programming: In this chapter, we explore more advanced features of the R programming language.
2. Working with Vectors and Lists: In this chapter, we explore two essential data structures, namely, vectors and lists.
3. Functional Programming: In this chapter, we explore an advanced feature of the R programming language, namely, functional programming.
4. Object-Oriented Programming: In this chapter, we learn how R handles object orientation and how we can use it to write more generic code.
5. Building an R Package: In this chapter, we learn the necessary components of an R package and how we can program our own.
6. Testing and Package Checking: In this chapter, we learn techniques for testing our R code and checking our R packages' consistency.
7. Version Control: In this chapter, we learn how to manage code under version control and how to collaborate using GitHub.
8. Profiling and Optimizing: In this chapter, we learn how to identify code hotspots where inefficient solutions are slowing us down and techniques for alleviating this.
9. Project 2: Bayesian Linear Regression: In the final chapter, we get to the second project, where we build a package for Bayesian linear regression.

Getting R and RStudio

You will need to install R on your computer to do the exercises in this book. I suggest that you get an integrated environment since it can be slightly easier to keep track of a project when you have your plots, documentation, code, etc., all in the same program.

I use RStudio (`www.rstudio.com/products/RStudio`), which I warmly recommend. You can get it for free—just follow the link—and I will assume that you have it when I need to refer to the software environment you are using in the following chapters. There won't be much RStudio specific, though, and most tools for working with R have mostly the same features, so if you want to use something else, you can probably follow the notes without any difficulties.

Projects

You cannot learn how to analyze data without analyzing data, and you cannot understand how to develop software without developing software either. Typing in examples from the book is nothing like writing code on your own. Even doing exercises from the book—which you really ought to do—is not the same as working on your own projects. Exercises, after all, cover minor isolated aspects of problems you have just been introduced to. There is not a chapter of material presented before every task you have to deal with in the real world. You need to work out by yourself what needs to be done and how. If you only do the exercises in this book, you will miss the most crucial lesson in analyzing data:

- How to explore the data and get a feeling for it
- How to do the detective work necessary to pull out some understanding from the data
- How to deal with all the noise and weirdness found in any data set

And for developing a package, you need to think through how to design and implement its functionality such that the various functions and data structures fit well together.

I will go through a data analysis project to show you what that can look like in this book. To learn how to analyze data on your own, you need to do it yourself as well—and you need to do it with a data set that I haven't explored for you. You might have a data set lying around you have worked on before, a data set from something you are just interested in, or you can probably find something interesting at a public data repository, for example, one of these:

- RDataMining.com: `www.rdatamining.com/resources/data`
- UCI Machine Learning Repository: `http://archive.ics.uci.edu/ml/`
- KDNuggets: `www.kdnuggets.com/datasets/index.html`
- Reddit R Data sets: `www.reddit.com/r/datasets`
- GitHub Awesome Public Data sets: `https://github.com/caesar0301/awesome-public-datasets`

I suggest that you find yourself a data set and that you, after each lesson, use the skills you have learned to explore this data set. Pick data structured as a table with observations as rows and variables as columns since that is the form of the data we will consider in this book. At the end of the first eight chapters, you will have analyzed this data. You can write a report about your analysis that others can evaluate to follow and maybe modify it: you will be doing reproducible science.

For the programming topics, I will describe another project illustrating the design and implementation issues involved in making an R package. There, you should be able to learn from implementing your own version of the project I use, but you will, of course, be more challenged by working on a project without any of my help at all. Whatever you do, to get the full benefit of this book, you really ought to make your own package while reading the programming chapters.

CHAPTER 1

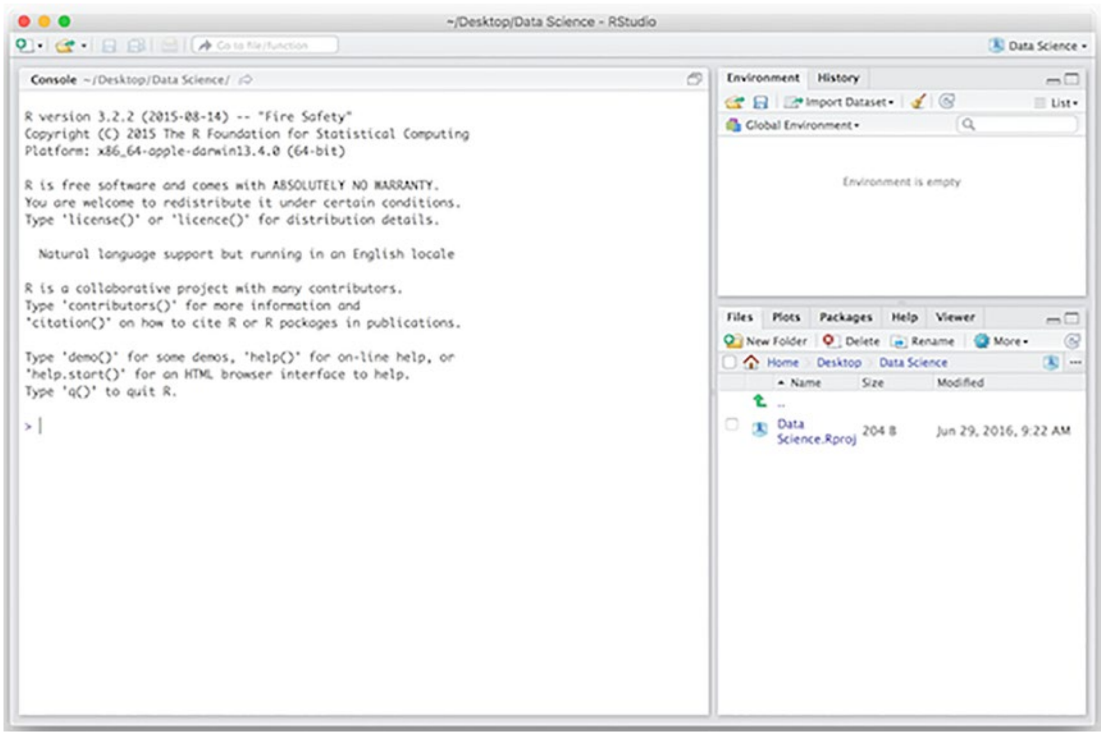
Introduction to R Programming

We will use R for our data analysis, so we need to know the basics of programming in the R language. R is a full programming language with both functional programming and object-oriented programming features, and learning the complete language is far beyond the scope of this chapter. We return to it later, when we have a little more experience using R. The good news is, though, that to use R for data analysis, we rarely need to do much programming. At least, if you do the right kind of programming, you won't need much.

For manipulating data—how to do this is the topic of the next chapter—you mainly have to string together a couple of operations, such as "group the data by this feature" followed by "calculate the mean value of these features within each group" and then "plot these means." Doing this used to be more complicated to do in R, but a couple of new ideas on how to structure data flow—and some clever implementations of these in packages such as `magrittr` and `dplyr`—have significantly simplified it. We will see some of this at the end of this chapter and more in the next chapter. First, though, we need to get a taste of R.

Basic Interaction with R

Start by downloading RStudio if you haven't done so already. If you open it, you should get a window similar to Figure 1-1. Well, except that you will be in an empty project while the figure shows (on the top right) that this RStudio is opened in a project called "Data Science." You always want to be working on a project. Projects keep track of the state of your analysis by remembering variables and functions you have written and keep track of which files you have opened and such. Go to File and then New Project to create a project. You can create a project from an existing directory, but if this is the first time you are working with R, you probably just want to create an empty project in a new directory, so do that.



```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

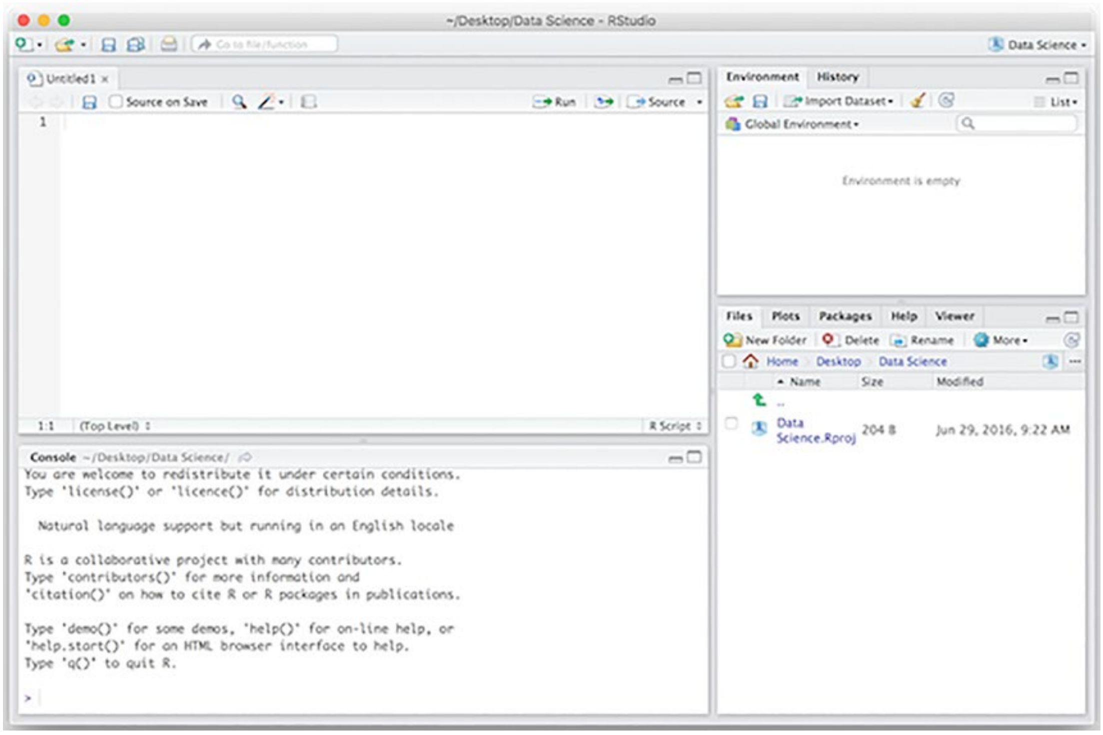
Name	Size	Modified
..		
Data Science.Rproj	204 B	Jun 29, 2016, 9:22 AM

Figure 1-1. *RStudio*

Once you have RStudio opened, you can type R expressions into the console, which is the frame on the left of the RStudio window. When you write an expression there, R will read it, evaluate it, and print the result. When you assign values to variables, and we will see how to do this shortly, they will appear in the Environment frame on the top right. At the bottom right, you have the directory where the project lives, and files you create will go there.

To create a new file, you go to File and then New File.... There you can select several different file types. Those we are interested in are the R Script, R Notebook, and R Markdown types. The former is the file type for pure R code, while the latter two we use for creating reports where documentation text is mixed with R code. For data analysis projects, I would recommend using either Notebook or Markdown files. Writing documentation for what you are doing is helpful when you need to go back to a project several months down the line.

For most of this chapter, you can just write R code in the console, or you can create an R Script file. If you create an R Script file, it will show up on the top left; see Figure 1-2. You can evaluate single expressions using the Run button on the top right of this frame or evaluate the entire file using the Source button. For writing longer expressions, you might want to write them in an R Script file for now. In the next chapter, we will talk about R Markdown, which is the better solution for data science projects.



```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Figure 1-2. *RStudio with a new R Script file open*

Using R As a Calculator

You can use the R console as a calculator where you type in an expression you want to calculate, hit "enter," and R gives you the result. You can play around with that a little bit to get familiar with how to write expressions in R—there is some explanation for how to write them in the following—and then moving from using R as a calculator to writing more sophisticated analysis programs is only a matter of degree. A data analysis program is little more than a sequence of calculations, after all.

Simple Expressions

Simple arithmetic expressions are written, as in most other programming languages, in the typical mathematical notation that you are used to:

```
1 + 2

## [1] 3

4 / 2

## [1] 2

(2 + 2) * 3

## [1] 12
```

Here, the lines that start with ## show the output that R will give you. By convention, and I don't really know why, these two hash symbols are often used to indicate that in R documentation.

It also works pretty much as you are used to, except, perhaps, that you might be used to integers behaving as integers in a division. At least in some programming languages, division between integers is integer division, but in R you can divide integers, and if there is a remainder, you will get a floating-point number back as the result:

```
4 / 3

## [1] 1.333333
```

When you write numbers like 4 and 3, they are always interpreted as floating-point numbers, even if they print as integers, that is, without a decimal point. To explicitly get an integer, you must write 4L and 3L:

```
class(4)

## [1] "numeric"

class(4L)

## [1] "integer"
```

It usually doesn't matter if you have an integer or a floating-point number, and everywhere you see numbers in R, they are likely to be floats.

You will still get a floating-point if you divide two integers, and there is no need to tell R explicitly that you want floating-point division. If you do want integer division, on the other hand, you need a different operator, %/%:

```
4 %/% 3
```

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

In many languages, % is used for getting the remainder of a division, but this doesn't quite work with R where % is used for something else (creating new infix operators), so in R the operator for this is %%:

```
4 %% 3
```

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

In addition to the basic arithmetic operators—addition, subtraction, multiplication, division, and the modulus operator we just saw—you also have an exponentiation operator for taking powers. For this, you can use either ^ or ** as infix operators:

```
2 ^ 2
```

```
## [1] 4
```

```
2 ** 2
```

```
## [1] 4
```

```
2 ^ 3
```

```
## [1] 8
```

```
2 ** 3
```

```
## [1] 8
```

There are some other data types besides numbers, but we won't go into an exhaustive list here. There are two types you do need to know about early, though, since they are frequently used and since not knowing about how they work can lead to all kinds of grief. Those are strings and "factors."