

R Module 1

Alex Fout¹

2020-06-28

¹Department of Statistics, Colorado State University, fout@colostate.edu

Contents

1	Welcome!	5
1.1	Associated CSU Course	6
2	Course Preliminaries	7
2.1	Course Topics & Syllabus	8
2.2	Running your first R Code	9
2.3	What do you hope to get out of this course?	11
3	Installing R	13
3.1	Computer Basics	13
3.2	Install R & R Studio	14
3.3	Successfull Installation	15
3.4	Running Code in RStudio Console	15
3.5	Workspace setup	15
3.6	Reflection & Assignment	16
4	The R Ecosystem	17
4.1	The R Programming Language	17
4.2	The R Ecosystem	17
4.3	R Interfaces	18
4.4	Assignment	18
4.5	The R Community	18
5	R Programming Fundamentals	19
5.1	Programming Preliminaries	19
5.2	(Atomic) Data Types	25
5.3	Data Structures	25
5.4	Practice	25
5.5	R Objects	25
5.6	Quiz	26
5.7	Loading / Saving Data	26
5.8	Downloading and Saving	26
5.9	Working With Data	26

5.10 Practice	27
5.11 Basic Control Flow	27
5.12 Advanced Control Flow	27
5.13 Writing FUnctions	27
5.14 Working With Popular Packages	28
5.15 Assignment	28

Chapter 1

Welcome!

Hi, and welcome to the R Module 1 course at Colorado State University!

This course is the first of three 1 credit courses intended to introduce the R programming language to those with little or no programming experience.

Through these Modules (courses), we'll explore how R can be used to do the following:

1. Perform basic computations and logic, just like any other programming language
2. Load, clean, analyze, and visualise data
3. Run scripts
4. Create reproducible reports so you can explain your work in a narrative form

In addition, you'll also be exposed to some aspects of the broader R community, including:

1. R as free, open source software
2. The RStudio free software
3. Publicly available packages which extend the capability of R
4. Events and community groups which advocate for the use of R and the support of R users

More detail will be provided in the Course Topics laid out in the next chapter.

1.0.1 How To Navigate This Book

To move quickly to different portions of the book, click on the appropriate chapter or section in the the table of contents on the left. The buttons at the top of the page allow you to show/hide the table of contents, search the book, change font settings, download a pdf or ebook copy of this book, or get hints

on various sections of the book. The faint left and right arrows at the sides of each page (or bottom of the page if it's narrow enough) allow you to step to the next/previous section. Here's what they look like:



Figure 1.1: Left and right navigation arrows

1.1 Associated CSU Course

This bookdown book is intended to accompany the associated course at Colorado State University, but the curriculum is free for anyone to access and use. If you're reading the PDF or EPUB version of this book, you can find the “live” version at <https://csu-r.github.io/Module1/>, and all of the source files for this book can be found at <https://github.com/CSU-R/Module1>.

Chapter 2

Course Preliminaries

This course is presented as a bookdown document, and is divided into chapters and sections. Each week, you'll be expected to read through the chapter and complete any associated exercises, quizzes, or assignments.

2.0.1 Special Boxes

Throughout the book, you'll encounter special boxes, each with a special meaning. Here is an example of each type of box:

💡 **Reflect:** This box will prompt you to pause and reflect on your experience and/or learning. No feedback will be given, but this may be graded on completion.

✍️ This box will signify a quiz or assignment which you will turn in for grading, on which the instructor will provide feedback.

📖 This box is for checking your understanding, to make sure you are ready for what follows.

📺 This box is for displaying/linking to videos in order to help illustrate or communicate concepts.

⚠ This box will warn you of possible problems or pitfalls you may encounter!

☀ **Bonus:** This box is to provide material going beyond the main course content, or material which will be revisited later in more depth.

💬 This box will prompt for your feedback on the organization of the course, so we can improve the material for everyone!

2.0.2 How This Book Displays Code

In addition, you may see R code either as part of a sentence like this: `1+1`, or as a separate block like so:

```
1+1
```

```
[1] 2
```

Sometimes (as in this example) we will also show the **output**, that is, the result of running the R code. In this case the code `1+1` produced the output `2`. If you hover over a code block with your mouse, you will see the option to copy the code to your clipboard, like this:

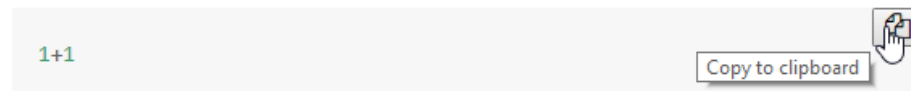


Figure 2.1: copying code from this book

This will be useful when you are asked to run code on your computer.

2.0.3 Next Steps

When you're ready, go to the next section to learn about the course syllabus and grading policies.

2.1 Course Topics & Syllabus

TODO: coming soon!

2.1.1 Schedule

Week	Weekday	Date	Reading	Assignments
1	Monday	July 13		
1	Wednesday	July 15		
1	Friday	July 17		
2	Monday	July 20		
2	Wednesday	July 22		
2	Friday	July 24		
3	Monday	July 27		
3	Wednesday	July 29		
3	Friday	July 31		
4	Monday	August 03		
4	Wednesday	August 05		
4	Friday	August 07		

2.1.2 Syllabus

2.1.3 Approach To Learning

- growth mindset
- do-first

2.1.4 Grading

2.2 Running your first R Code

Enough of the boring stuff, let's run some R code! Normally you will run R on your computer, but since you may not have R installed yet, let's run some R code using a website first. As you run code, you'll see some of the things R can do. In a browser, navigate to rdr.io/snippets, where you'll see a box that looks like this:

The box comes with some code entered already, but we want to use our own code instead, so delete all the text, starting with `library(ggplot2)` and ending with `factor(cyl))`. In its place, type `1+1`, then click the big green “Run” button. You should see the `[1] 2` displayed below. So if you give R a math expression, it will evaluate it and give the result. Note: the “correct answer” to `1 + 1` is 2, but the output also displays `[1]`, which we won't explain until later(TODO), so you can ignore that for now.

Next, delete the code you just wrote and type (or copy/paste) the following, and run it:

```
factorial(10)
```

```

library(ggplot2)

# Use stdout as per normal...
print("Hello, world!")

# Use plots...
plot(cars)

# Even ggplot!
qplot(wt, mpg, data = mtcars, colour = factor(cyl))

```

Run (Ctrl-Enter)

Figure 2.2: rdrv code entry box

The result should be a very large number, which is equivalent to $10!$, that is, $10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$. This is an example of an R *function*, which we will discuss more in Section (TODO: insert ref).

Aside from math, R can produce plots. Try copy/pasting the following code into the website:

```

x <- -10:10
plot(x, x^2)

```

You should see points in a scatter plot which follow a parabola. Here's a more complicated example, which you should copy/paste into the website and run:

```

library(ggplot2)
theme_set(theme_bw())
ggplot(mtcars, aes(y=mpg, fill=as.factor(cyl))) +
  geom_boxplot() +
  labs(title="Engine Fuel Efficiency vs. Number of Cylinders", y="MPG", fill="Cylinders")
theme(legend.position="bottom",
      axis.ticks.x = element_blank(),
      axis.text.x = element_blank())

```

R can be used to make many types of visualizations, which you will do more of in Section (TODO: insert ref).

☀ **Bonus:** This may be the first time you've seen R, so it's okay if you don't understand how to read this code. We'll talk more later about what each statement is doing, but for now, here is a brief description of some of the code above:

- `-10:10` This creates a sequence of numbers starting from -10 and ending at 10. That is, $-10, -9, -8, \dots, 8, 9, 10$.
- `library` This is a function which loads an R *package*. R packages provide extra abilities to R

2.3 What do you hope to get out of this course?

To close out this chapter, it would be healthy for you to reflect on what you'd like to get from this course. Take some time to think through each question below, and write down your answers. It is fine if your honest answer is *I don't know*. In that case, try to come up with some possible answers that *might* be true.

💡 **Reflect:**

1. Why are you taking this course?
2. If this course is required for your major, how do you think it is supposed to benefit you in your studies?
3. What types of data sets related to your field of study may require data analysis?
4. What skills do you hope to develop in this course, and how might they be applied in your major and career?

TODO: canvas assignment?

Store your answers in a safe place, and refer to them periodically as you progress through the course. You may find that you aren't achieving your goals and that some adjustment to how you are approaching the course may be necessary. Or you may find that your goals have changed, which is fine! Just update your goals so that you have something to refer back to.

Chapter 3

Installing R

In the previous chapter, you ran R code on a website. The purpose of this chapter is to install R on your own computer, so that you can run R without needing access to the internet.

3.1 Computer Basics

If you're new to computers, this section will be important for you to get set up. If you understand the basics of operating systems, directory structures on your computer, and downloading/installing files, then you can safely skip to the next section.

3.1.1 Operating Systems

R works on Windows, Mac OS X, and several Linux-based operating systems, so if you have one of these operating systems, you'll be able to install and use R. At least, this is mostly true.

🚧 Some versions of Windows that run on ARM processors cannot install R, and installing R on a Chromebook will likely be more complicated (see [here](#)).

3.1.2 Downloads and Installations

To install R, you'll have to download a file from the internet which performs the installation. After you install R, you shouldn't have to download anything to run R.

TODO: take out?

3.2 Install R & R Studio

You'll actually be installing *two* separate programs, both of which are free to use. The *first* is the R programming language. The *second* is a separate program called R Studio. R Studio will be the primary way in which you interact with R in this class, we will say more about this later.

3.2.1 Installing R

Installation will look slightly different depending on the operating system, but the major steps are the same.

- First, navigate to the CRAN Mirrors Site, which lists several locations from which R can be downloaded.
- Find a location near you (or not, this isn't critical) and click on the link to be brought to the mirror site.

From this point, this will change depending on your operating system.

3.2.1.1 Windows

- Click "Download R for Windows", then click "base".
- Finally, Click "Download R X.Y.Z for Windows", where X, Y, and Z will be numbers. These numbers indicate which version of R you'll be installing. As of the publishing of this book, R is on version **r_version**.
- any other settings. Your computer might prompt for the location on your computer that you would like to save the file. Select a location (reasonable options are your Downloads folder or the Desktop) and select "save".
- When the download completes, find the downloaded file in the File Explorer and double click to run it. This will start the installation process.
- Follow the on screen prompts. For the most part you can click "continue", "agree", "install" as appropriate, and you don't have to worry about changing any installation settings.

■ Create windows R install Video

3.2.1.2 Mac OS X

- Click "Download R for (Mac) OS X"
- Click "R-X.Y.Z.pkg", where X, Y, and Z will be numbers. These numbers indicate which version of R you'll be installing. As of the publishing of this book, R is on version **r_version**.
- Your computer might prompt for the location on your computer that you would like to save the file. Select a location and select "save".
- When the download completes, find the downloaded file in the Finder and double click to run it. This will start the installation process.

- Follow the on screen prompts. For the most part you can click “continue”, “agree”, “install” as appropriate, and you don’t have to worry about changing any installation settings.

■ ?Ben? install from Mac OS X?

3.2.2 Installing R Studio

3.3 Successful Installation

TODO: add some basic tasks here to make sure R and RStudio work. (screenshots?)

■ todo: video orienting to R Studio (Matt video?)

✎ todo: create assessment

3.4 Running Code in RStudio Console

TODO: repeat online examples + Pull from here: <https://geanders.github.io/RProgrammingForResearch/r-preliminaries.html>

^v todo: create assessment

3.5 Workspace setup

3.5.1 Recommended Settings

- don’t save/load workspace
- how to adjust font size (in options and with ctrl +/-)
- dark mode!

3.5.2 Setting working directory

-

3.5.3 Create R Studio Project and directories for class

TODO: pull from: <https://geanders.github.io/RProgrammingForResearch/r-preliminaries.html#r-scripts>

Note: This will automatically set your working directory


Folder structure:

- Raw Data
- Assignments
-

3.6 Reflection & Assignment

 **Reflect:** Before moving on to the next section

TODO: Kathleen thought this could be a progress check, I think we can have it be the first homework.

 todo: create R script to do some math, then turn in. provide template with comment block at the top

Chapter 4

The R Ecosystem

4.1 The R Programming Language

There are many important qualities to R, here are some of the most important

4.1.1 R is a Programming Language

☀ Bonus:

4.1.2 R is Free

4.1.3 R is Open Source

4.2 The R Ecosystem

4.2.1 R Packages

☀ Bonus: Other notable packages

4.3 R Interfaces

4.3.1 Where can R be run?

🎥 Video comparing the three, perhaps showing the difference between the three.

4.4 Assignment



4.4.1 R Interfaces

- RStudio
- RGui
- Terminal/Command line

4.5 The R Community

- Users
- Package developers

Popular “places” that R users hang out:

Interact

TODO: need more ideas here.

Helping each other out

Where you can get help:

- Stack Overflow
- Google!

☀ **Bonus:** link to some cool visualizations, communities, etc.

Chapter 5

R Programming Fundamentals

5.1 Programming Preliminaries

💡 Reflect:

💡 Reflect:

1. Look at a sentence in a language you don't know, look carefully at the symbols, spacing and characters.
2. Recall learning a foreign language, how you had to learn the syntax and grammar rules.
3. Now think about English (or another language you know well) and think about the syntax and grammar rules that you take for granted.

TODO: talk about syntax and grammar

All human languages rely on a set of rules called grammar, which describe how the language should be used to communicate. When two humans communicate with a language, they both must agree on the the rules of that language. R also has rules that must be followed in order for a human (*you*) to communicate with a computer, in order to tell the computer what to do. In human language, grammar is often fluid and evolving, and two people may have to adapt their use of the language in order to communicate. With R, the fules are fixed, and

the computer “knows” them perfectly. Therefore it is up to you to learn the rules in order to make the computer do exactly what you want it to do.

Therefore it’s important to cover some of the basic rules of the R programming language before you can learn all of the things R can do. So here they are:

5.1.1 R Commands

Like most programming languages, R consists of a set of *commands* which form the sequence of instructions which the computer completes. Here is an example of a command, followed by the result.

```
print("hello, world!")
```

```
[1] "hello, world!"
```

This command is telling R to print out a message. R code usually contains more than one command, and typically each command is put on a separate line. Here are multiple commands, each on a separate line:

```
print("The air is fine!")
print(1+1)
print(4 > 5)
```

```
[1] "The air is fine!"
[1] 2
[1] FALSE
```

The first command prints another message, the second command does some math then then prints the result, and the third command evaluates whether the statement is true or false and prints the result. Generally, it’s a good idea to put separate commands on separate lines, but you *can* put multiple commands on the same line, **as long as you separate them by a semicolon**. See this code for example:

```
x <- 1+1; print(x); print(x^2)
```

```
[1] 2
[1] 4
```

In this example, three commands are given on one line. The first command creates a new *variable* called `x`, the second command prints the value of `x`, and the third command prints the value of `x squared`. We see that the semicolon, `;`, serves as the command *termination*, because it tells R where one command ends and another begins. When a line contains a single command, no semicolon is necessary at the end, but including a semicolon doesn’t have any effect either.

```
print("This line doesn't have a semicolon")
print("This line does have a semicolon");
```

```
[1] "This line doesn't have a semicolon"
```

```
[1] "This line does have a semicolon"
```

🚧 Including multiple semicolons (e.g. `print("hello");;`) does not work!

TODO: make these code blocks display on multiple lines:

☀ **Bonus:** So far, we've seen that you can place one command on one line, multiple commands on multiple lines, multiple commands on one line, so you may ask: can you place one command on multiple lines? The answer is *sometimes*, depending on the command. Generally, if R is expecting the end of a command but doesn't see one by the time it gets to the end of a line, it will continue looking on the next line. Here are some examples: `1 + \ 1`
`print("This string is on multiple lines")`

🚧 At this point, we've introduced several new types of R commands (assigning a variable, squaring a number, etc.), and we will talk more specifically about these later. The important part of this section is how R code is arranged into different *commands*

Lastly, commands can be “grouped together” using left and right curly braces: { and }. Here's an example:

```
{
  print("here's some code that's all grouped together")
  print(2^3 - 7)
  w <- "hello"
  print(w)
}
```

```
[1] "here's some code that's all grouped together"
[1] 1
[1] "hello"
```

The above grouped code is indented so that it looks nice, but it doesn't have to be:

```
{
print("here's some code that's all grouped together")
print(2^3 - 7)
```

```
w <- "hello"
print(w)
}
```

```
[1] "here's some code that's all grouped together"
[1] 1
[1] "hello"
```

☀ **Bonus:** Indenting is an example of coding *style*, which are formatting decisions which don't affect the results of the code, but are meant to enhance readability. We'll talk more about coding style later. TODO: talk about style later. In some programming languages, Python for example, white space matters. That is, code indents and other spaces change the way the code runs. In R, white space *does not* matter, so things like indents are used purely for readability.

What does it mean to “group” code? At this point there is no practical difference, each command gets executed whether or not it is grouped inside curly braces. However, code grouping will become very important later on, when we discuss *control flow* in section (TODO: add ref).

🔗 In RStudio, open a new R script and type in all the R commands from this section, to verify that you get the same result. It's good practice!

5.1.2 Comments

When writing R code, you may wish to include notes which explain the code to your future self or to other humans. This can be done with *comments*, which are ignored by R when it is running the code. The “#” comment Here's an example of some comments:

```
# Let's define y and z
y <- 8
z <- y + 5 # adding 5 to y and assigning the result to z
## This is still a comment, even though we're using two #'s
```

TODO: add note in strings section about special characters and blocks

Notice that a line can contain only a comment

5.1.3 Blank Lines

Blank lines in R are ignored, but they can be used to organize code and enhance readability:

```
print("The sky is blue")
# the blank line below here is ignored

print("The grass is green")
```

```
[1] "The sky is blue"
[1] "The grass is green"
```

5.1.4 CaSe SeNsItIvItY

In R, variables, functions, and other objects (which we'll talk about later), have names. These names are case sensitive, so you must be careful when referencing an object by name. Here we create two variables and give them different values, notice how they are different from each other:

```
A <- 4
a <- 5

print(a)
print(A)
```

```
[1] 5
[1] 4
```

This may seem obvious, but case sensitivity applies to functions (which we'll talk about later) too. We've been using the `print` function a lot in the above examples, which begins with a lower case p. There is no `Print` function:

```
Print("testing")
```

```
Error in Print("testing"): could not find function "Print"
```

5.1.5 ?

One *very* nice thing in R is the documentation that accompanies it. Every function included in R (like `print`) has documentation that explains how that function works. To access the documentation, use a `?` followed by the name of the function, like so:

```
?print
```

🔍 The output of the above code chunk is not shown, because the result of this code is best viewed in RStudio. Go to R Studio and type in `?print` and observe what happens!

5.1.6 ??

If you don't remember the exact name of a function, or would like to search for general matches to a topic, then you can use `??`. For example, trying `?Print` produces an error, because there is not `Print` function (remember, R is case sensitive), so there's no documentation to go with it. However, the following should still work:

```
??Print
```

☀ **Bonus:** Programmers have a sense of humor, too! Try running `???print` to see a small joke. Remember, comedic taste varies!

TODO: put these bonus blocks in more appropriate places:

☀ **Bonus:** <https://rstudio.com/wp-content/uploads/2016/10/r-cheat-sheet-3.pdf>

☀ **Bonus:** Want to know more about R syntax? Try typing `?Syntax` in the R console (then press **Enter**).

🏠 As we've seen, symbols and characters have specific meaning in R. You must be careful not to ignore things like semicolons, curly braces, parentheses, when reading R code. This takes practice!

Okay, now that we've covered some of the basics, it's time to start learning how to do useful things in R! The next few sections will describe the different types of data that R can handle.

5.2 (Atomic) Data Types

R can store and manipulate different pieces of information (“data”).

Give some examples.

what is a data type?

https://www.tutorialspoint.com/r/r_data_types.htm


<https://swcarpentry.github.io/r-novice-inflammation/13-supply-data-structures/#:~:text=R's%20basic%20data%20types%20are,%2C%20data%20frame%2C%20and%20factors.>

5.3 Data Structures

These are just other types of data

<https://adv-r.hadley.nz/vectors-chap.html>

5.4 Practice

A purple rectangular box containing a small icon of a person with arms raised and the text "asdf".

5.5 R Objects

5.5.1 *Everything* is an object in R

5.5.2 Assigning Objects


A black rectangular box containing a small icon of a sun and the text "Bonus: asdf".

5.5.3 Inspecting objects

str, names, dim

5.5.4 Null Objects

5.6 Quiz

 asdf

5.7 Loading / Saving Data

5.7.1 “Taster”(?) list of file forms and sources


5.7.2 reading/writing csv

5.7.3 Best practices

principles of Tidy Data (wickham 2014)

raw data as read only

5.8 Downloading and Saving

 downloading and saving example csv from canvas or website

5.9 Working With Data

5.9.1 Summarizing vectors

Mean, std, etc.

5.9.2 Summarizing matrices

5.9.3 Summarizing vectors

5.9.4 Basic Plotting

 **Bonus:** ggplot example

5.9.5 Basic indexing

5.9.6 Advanced indexing

☀ Bonus: dplyr example

5.10 Practice

^ asdf

5.11 Basic Control Flow

If/else

loops (for, while)

switch

5.12 Advanced Control Flow

*apply family

5.13 Writing Functions

Function scope

☀ Bonus:

- Pass by value, contrast with other languages

5.14 Working With Popular Packages

5.15 Assignment

✎ Make R Markdown PDF that covers most/all of CH4: read in data, do basic things with it.