

Stat 5100 in R: Setup and Introduction

Contents

1 Introduction

2 Should I take this class using R?

3 R Setup

- 3.1 Download R
- 3.2 Download RStudio
- 3.3 Download the Stat 5100 R Package
- 3.4 Test your R installation

4 R Basics

- 4.1 Running commands and scripts
- 4.2 Getting help on functions
- 4.3 Reading in data
 - 4.3.1 Reading in Stat 5100 datasets
 - 4.3.2 Reading in data by CSV
 - 4.3.3 Creating data by hand
- 4.4 Altering datasets
- 4.5 Filtering datasets

5 Miscellaneous Notes and FAQ

1 Introduction

As of Spring 2021, the Stat 5100 course is being offered in R for the very first time. You have the option to complete this course in either SAS or R. Both languages are very popular and useful for the material you will learn in this class, but are very different in style.

Here are the main things you should know about each language:

SAS

- Costs a lot of money (You get to use SAS for free in this class because you are affiliated with an academic institution)
- Very useful “out of the box.” Many powerful features are available right away.
- Commonly used at large corporations with large datasets.
- The syntax (the rules of the language) are incredibly different from nearly every other programming language, so learning SAS will not necessarily help you learn general programming skills.
- The philosophy of SAS is “let’s give you a huge amount of output, we’ll let you figure out which parts of the output you need.” If you create a regression model in SAS, you will be thrown a huge amount of output and diagnostics right away by default. Although the large amount of output can feel overwhelming, it can be very nice to have when you’re not sure what you’re looking for.

R

- Completely free and open-source for everyone.
- A very lightweight and small language (compared to SAS), so many powerful features are not available natively.
- The most commonly used language for data science and statistics in both academia and industry. Behind Python, R is the second most commonly used language for data science in industry. (We would love to offer this course in Python eventually!)
- The syntax of R is similar to some other scripting languages such as Python, so learning R will help you learn general programming skills.
- Requires stronger programming skills to get started.
- The philosophy of R is “We assume you know what you’re looking for, so we’re going to give you the bare minimum.” In contrast with SAS, R does not give very much output and diagnostics. To obtain the same things that SAS gives you, you must know exactly what you are looking for and run appropriate functions to give you the output you want.
- To expand the power of R, we use what are called “packages.” Packages are code that is written by other people, and we can use many of these packages to do the same things that SAS can do.

2 Should I take this class using R?

Take this class using R if:

1. You have used R before and you are comfortable programming. If you have taken Stat 5050: Introduction to R here at USU, then you should be well-prepared to take this class using R. Ideally you should know the basics of manipulating dataframes and running simple functions.
2. You are okay being a guinea pig: this is the first semester this course is being offered in R so we can not guarantee a 100% smooth experience in all cases.

3 R Setup

To get ready to complete this class using R, we’ll need to set up your system to have both R and RStudio.

R is the language itself, and RStudio is simply a program that interacts with the language and provides a nice and clean environment for working with R. If you are familiar with programming, RStudio is just an IDE (integrated development environment) for R. RStudio is by far the most popular IDE for R. You can download and work with R without using RStudio, but you are restricted to doing so in your system’s terminal which can be very hard to work with.

3.1 Download R

Navigate yourself to [the official R download page](#). Click on any of the mirror links, and then click “Download R” for whatever operating system you use. Follow the instructions of the installer, and then you should be good to go.

3.2 Download RStudio

Navigate yourself to [the official RStudio download page](#). Click on “RStudio Desktop,” then click on “Download RStudio Desktop” underneath the open source edition. Click on the download button again under the free version. When you are set up, you should be able to open up RStudio and have it automatically identify your R installation.

3.3 Download the Stat 5100 R Package

Next, you will want to download the Stat 5100 R package. This package contains all of the data that we will use in this class, as well as many complex functions and plotting routines. The goal of this package is to take the focus off of the programming and move the focus to the topic of the class: regression.

To download and install the package, type the following into the console in RStudio:

```
install.packages("devtools")
devtools::install_github("ethanancell/stat5100package")
```

3.4 Test your R installation

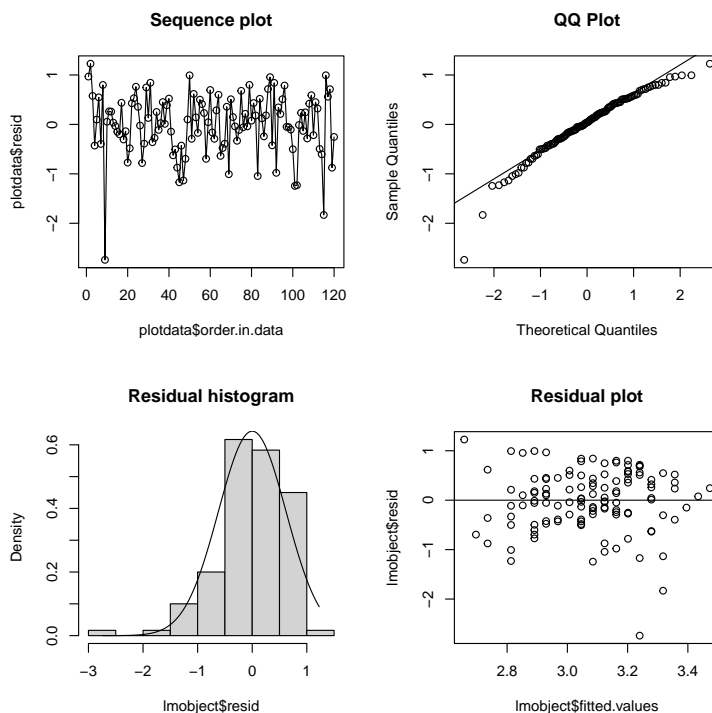
Try the following in your R console to make sure that you have your installation set up correctly. When you plot an image, you should see it pop up in the bottom-right corner of your screen under the “Plots” tab of RStudio.

```
# Load the Stat 5100 package into memory
library(stat5100)

# Load an included dataset from the library
data("college")

# Fit a regression model on the college dataset
college_lm <- lm(gpa ~ act, data = college)

# Show some plots
visual_assumptions(college_lm)
```



If you get the same results as the above, then your installation should be correctly setup and you will be good to go!

4 R Basics

Here is quick review of some basic skills you should be familiar with in R. If you feel comfortable doing all of the things below, then you should be just fine in this class.

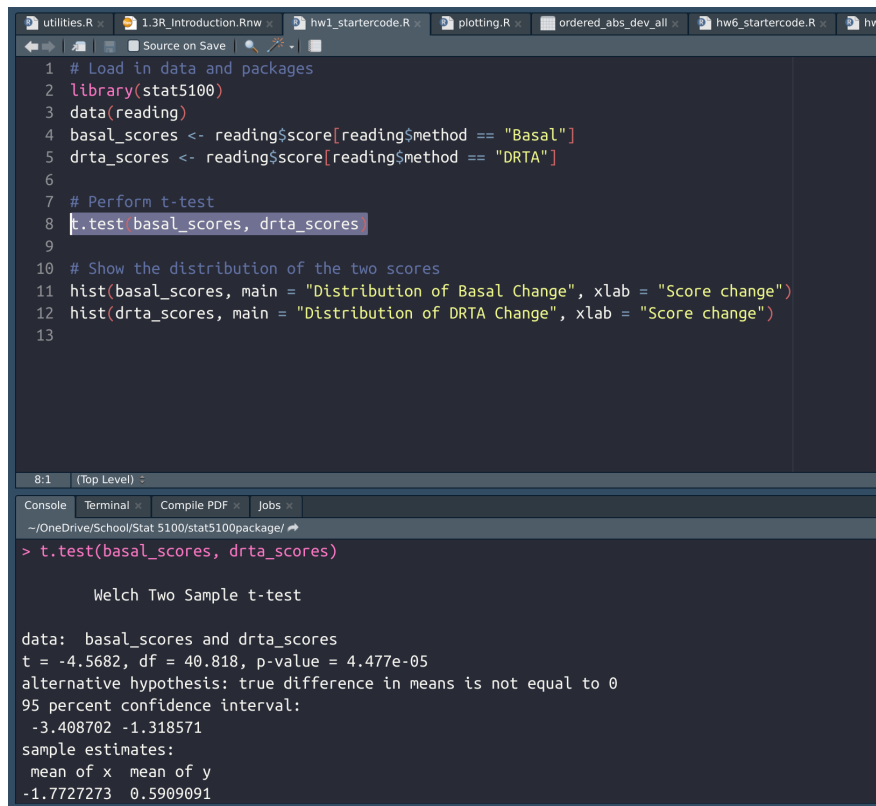
4.1 Running commands and scripts

In RStudio, there are two places where you see code. You have a console, and you should also have a script window. By default, just the console window will be open. In there, you can type R commands and there will be ran in an interactive manner.

```
> print("Running some basic commands in my console window")
[1] "Running some basic commands in my console window"
> 3+5
[1] 8
> rnorm(4)
[1] 1.0571192 0.2427443 2.1226940 -0.2715278
> |
```

You can also open R code inside a script window. If desired, you can run the entire R script. More often, you will be running bits and pieces of an R script at a time. Putting your code in a script has an advantage in that it's very easy to jump around your code and test little bits at a time without needing to type out your code every time.

If you go to File → Open File, you can open an R file inside RStudio. To run one chunk of code, select it with your mouse and then click either “Run” at the top right part of the script window, or Ctrl+Enter (my favorite method and much quicker than clicking run!)



The screenshot shows the RStudio interface. The top pane displays an R script with the following code:

```
1 # Load in data and packages
2 library(stat5100)
3 data(reading)
4 basal_scores <- reading$score[reading$method == "Basal"]
5 drta_scores <- reading$score[reading$method == "DRTA"]
6
7 # Perform t-test
8 t.test(basal_scores, drta_scores)
9
10 # Show the distribution of the two scores
11 hist(basal_scores, main = "Distribution of Basal Change", xlab = "Score change")
12 hist(drta_scores, main = "Distribution of DRTA Change", xlab = "Score change")
13
```

The bottom pane shows the console output for the command `t.test(basal_scores, drta_scores)`:

```
> t.test(basal_scores, drta_scores)

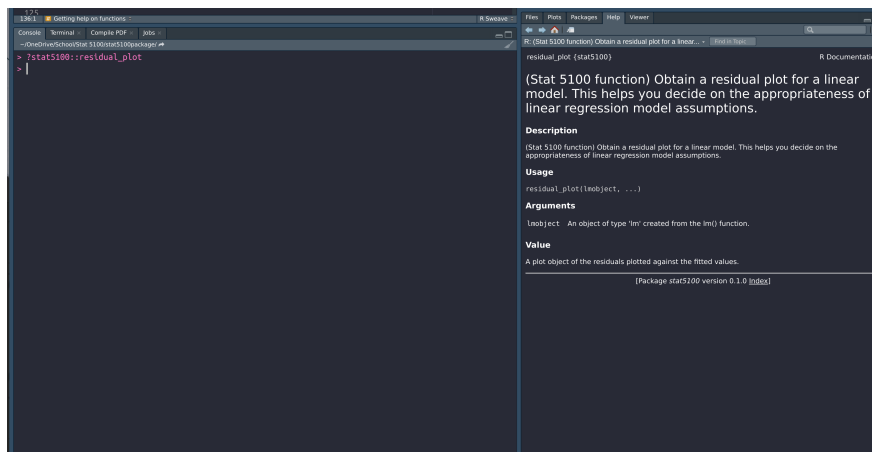
Welch Two Sample t-test

data: basal_scores and drta_scores
t = -4.5682, df = 40.818, p-value = 4.477e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.408702 -1.318571
sample estimates:
mean of x mean of y
-1.772723 0.5909091
```

4.2 Getting help on functions

Not even the best R programmers memorize how to use every single function. Very often, you will know that a function exists but you forget the exact parameters to enter and how to use the function. Fortunately, R

has integrated a very easy help page lookup feature. To use this lookup feature, prepend the command you want to learn about with a question mark “?” and enter it into the console. For example, to get help on the `stat5100::residual_plot()` function, type “`?stat5100residual_plot`” into the console.



Every single function that you will ever use in this class will come with a help page that can be accessed this way.

4.3 Reading in data

4.3.1 Reading in Stat 5100 datasets

Nearly all of the datasets that we use in this class (with a few exceptions) are made available right away in the `stat5100` package. To load them into memory, simply use the `data()` function with the name of the dataset in the parenthesis.

```
data(surgical)
```

Note that for this to work, you *must* have the Stat 5100 package loaded already:

```
library(stat5100)
```

4.3.2 Reading in data by CSV

To read in a CSV file, it is a little bit more tricky but still very easy. To load a CSV file, we use the `read.csv()` function in R with the location of the file as a string inside the parenthesis. Note that this function requires we assign the CSV data to an object name.

```
my_data_name <- read.csv(".././../././data_csv/grocery.csv")
```

Note that the file location is specified relevant to your current working directory. To access a file that is “up” a directory, use `..` to indicate this (you see this in the above command!)

To find out where your current working directory is, run the following:

```
getwd()
```

```
## [1] "/home/ethan/OneDrive/School/Stat 5100/teaching/Stat5100/notes/Module 1"
```

To set your working directory, you can use the `setwd()` function, but it is easier to go to Session → Set Working Directory → Choose Directory inside of RStudio.

4.3.3 Creating data by hand

Sometimes, we might need to create a dataframe by hand. Here is an example where we create a very dataframe and display its contents in the console:

```
my_amazing_dataframe <- data.frame(column_name1 = c("Yellow", "Blue", "Red"),
                                   some_numbers = c(5, 7, 89),
                                   more_numbers = c(4, 8, 1),
                                   some_logicals = c(TRUE, TRUE, FALSE))

my_amazing_dataframe
```

	column_name1	some_numbers	more_numbers	some_logicals
## 1	Yellow	5	4	TRUE
## 2	Blue	7	8	TRUE
## 3	Red	89	1	FALSE

4.4 Altering datasets

There are a variety of ways we might want to alter a dataset:

```
# Create a new column
my_amazing_dataframe <- cbind(my_amazing_dataframe,
                              muchos_numeros = my_amazing_dataframe$some_numbers *
                                                my_amazing_dataframe$more_numbers,
                              numbers_exponent = my_amazing_dataframe$some_numbers^5.5)

my_amazing_dataframe
```

	column_name1	some_numbers	more_numbers	some_logicals	muchos_numeros
## 1	Yellow	5	4	TRUE	20
## 2	Blue	7	8	TRUE	56
## 3	Red	89	1	FALSE	89

```
## numbers_exponent
## 1 6.987712e+03
## 2 4.446714e+04
## 3 5.267991e+10

# Set all instances of the number 1 inside the "more_numbers" column to be the
# number e instead.
my_amazing_dataframe$more_numbers[my_amazing_dataframe$more_numbers == 1] <- exp(1)

my_amazing_dataframe
```

	column_name1	some_numbers	more_numbers	some_logicals	muchos_numeros
## 1	Yellow	5	4.000000	TRUE	20
## 2	Blue	7	8.000000	TRUE	56
## 3	Red	89	2.718282	FALSE	89

```
## numbers_exponent
## 1 6.987712e+03
## 2 4.446714e+04
## 3 5.267991e+10
```

4.5 Filtering datasets

To conditionally access parts of a dataframe, you can run any of the following:

```

# print all rows where "some_numbers" was 5
my_amazing_dataframe[my_amazing_dataframe$some_numbers == 5, ]

##   column_name1 some_numbers more_numbers some_logicals muchos_numeros
## 1      Yellow           5           4           TRUE           20
##   numbers_exponent
## 1           6987.712

# Print columns 1 and 3
my_amazing_dataframe[, c(1,3)]

##   column_name1 more_numbers
## 1      Yellow      4.000000
## 2       Blue      8.000000
## 3       Red       2.718282

# Print all elements of some_numbers where some_numbers is either 5 or 7.
my_amazing_dataframe$some_numbers[(my_amazing_dataframe$some_numbers == 5 |
                                     my_amazing_dataframe$some_numbers == 7)]

## [1] 5 7

```

5 Miscellaneous Notes and FAQ

Here are some notes you should be aware of:

- To create a comment in your code (lines in a script that are notes and not actual code), place a `#` right before what you want to say. You probably have seen this in some of the code examples above.

Other random questions:

Q: Where is the best place to go for R help?

A: To be completely honest, the best place to ask questions about R is on Google. Most likely, any question you have has been asked by somebody else. Finding help for your code online is an incredibly important skill: anyone who programs in any capacity should become familiar with this skill eventually.

However, if you aren't finding what you need online or you want some human help, either the TA or the professor is very happy to answer your questions via email or office hours.

Q: Can I view all the R code that is in the Stat 5100 package?

A: Yes! If you are a more complex R user and you wish to see what is happening behind the scenes in some of these functions, then you are welcome to look in the source code of the R package to see what code lies behind these functions. For example, the `visual_assumptions()` function in the code snippet above does not exist natively in R, it is a function that comes from the Stat 5100 R package.

To see the code on Github, navigate to <http://www.github.com/ethanancell/stat5100package>. You are welcome to clone this repository or peek around the source code. All the R code is contained in the "R" folder, and is in either "utilities.R" or "plotting.R." The plotting file should be very self-explanatory: it is where all the plotting functions we'll use in this class live. In utilities.R, we have more numerical methods for different complex calculations we will need for various things we do in this class.

Feel free to clone the repository and peek around. If you have any questions about the package for the course, feel free to direct them to me at ethanancell@aggiemail.usu.edu.