Worksheet 1 : Introduction to R

Contents

Today's Agenda

- Using RStudio and RMarkdown
- R as a calculator
- Loading data into R
- Plotting data

During this worksheet you will explore R and RStudio, which you'll be using throughout the course both to learn the statistical concepts discussed in the course and to analyze real data and come to informed conclusions. To clarify: **R** is the name of the programming language itself and **RStudio** is a convenient interface.

A programming language is just the language for telling the computer what to do. The most frustrating thing about programming is that the computer will do exactly what you tell it to, so one tiny typo can make everything you are working on fail. But, the fact that computers do exactly what you tell them to is also what makes them great! The best thing you can do to avoid frustration is to test that the computer is doing what you're expecting often.

RStudio Interface

Launch RStudio either from jupyter.davidson.edu or locally from your laptop. Once it has booted up you should see an interface with a panel on the left and two panels on the right.

The panel on the left is the most important one; this is where R is working. The > indicates the line you are on with R ready to run. Test that R is working by running the command 3+5. R can be used as a calculator. Now try to run 3+*5 and notice that R tells you that it doesn't understand what you're saying to it. Errors often contain information about how to fix the error. This panel is called the console, we will explore more of what we can do here later.

If we execute code line by line, as we did in the last example, our work is not being saved. You will often want to re-run large chunks of your code and it would be a large waste of time to retype all of your commands every time you want to re-run code. Therefore, we should always be running our code in an R Markdown file. In the very top left of R Studio find the icon that is a sheet of paper with a green plus symbol on it and select a new R Markdown file and give it the name Testing. Once this has been opened you should have a new panel in the top left and the console you were using before is in the bottom left. Most the work we do from now on will be in an R Markdown file, not in the console, let's see how it works.

Let's delete everything below the part that says ## R Markdown (you can read this later by opening a new R Markdown file the same way as before).

The panel in the up per right has information about data you have stored in your computer. Try to run the following command a<-3+5. Now in a separate command type a . You have told your computer to remember the calculation you made and any time you want to see that calculation again you can just type the name you gave it. Also notice that the value of the variable a is shown in the upper right panel.

In the lower right panel has a lot of useful information. You can see the file manager where you can load data into R and download updated files if you are using jupyter. This is also where any plots you make will be displayed. Another useful feature is that the current packages you have loaded will also be displayed here.

R Packages

R is an open-source programming language, meaning that users can contribute packages that make our lives easier, and we can use them for free. Here are some common R packages we will use in this course:

- The suite of **tidyverse** packages: for data wrangling and data visualization
- **openintro**: for data and custom functions with the OpenIntro resources
- **gapminder**: for easy access to an excerpt of the Gapminder data on life expectancy, GDP per capita, and population by country

If these packages are not already available in your R environment, install them by typing the following lines of code into the console of your RStudio session, pressing the enter/return key after each one. Note that you can check to see which packages (and which versions) are installed by inspecting the Packages tab in the lower right panel of RStudio.

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

You may need to select a server from which to download; any of them will work. Next, you need to load these packages in your working environment. We do this with the library function. Run the following lines in your console.

```
library(tidyverse)
library(openintro)
library(gapminder)
```

You only need to *install* packages once, but you need to *load* them each time you relaunch RStudio.

The Tidyverse packages share common philosophies and are designed to work together. You can find more about the packages in the tidyverse at https://www.tidyverse.org.

Data Basics

We can obtain data a number of ways. One way is to create data explicitly, we will learn more about that later. Another way is to load data from a library. Since we have already loaded the gapminder package, let's get some data from it.

```
view(gapminder)
```

```
head(gapminder)
```

Now let's load the data from last class. If you are using jupyter, the file should already be in your file manager. If so, right click to import the data. Otherwise, you will need to download it from Moodle and run the following:

```
questionnaire <- read.table(file = "questionnaire_104.csv", sep = ",", header = TRUE)
```

This code may seem a little complicated but let's analyze it a bit. We are creating a varaible called questionnaire to store the data we collected in class. Then we call the function read.table() and tell the

computer which file we want to load. The next two bits of info tell the computer to sep arate the data at each comma and that our data has a header row (a top row that tells you the name of all the columns).

Variable Names

- Should not include spaces
- Should not be too long
- Should not contain special characters
- Cannot start with a number
- Should be concise, but descriptive

Inconsistent/Incorrect Formatting

- Dates
- Character variables with similar values
  - Different cases
  - Misspellings
  - Extra spaces

Class Activity

We are going to practice finding, loading, and dealing with messy data.

- About the data
  - United Nations Office for the Coordination of Humanitarian Affairs
  - Humanitarian Data Exchange (HDX)
  - https://data.humdata.org
  - https://data.humdata.org/about/terms
  - What are the benefits of "open data?"

Getting the Data

- Go to: https://data.humdata.org/dataset
  - Search for "Caribbean"
  - Select: "Caribbean Hurricanes: Regional Who is Doing What Where (3W)"
  - Download the data
  - Load the `tidyverse` package

Utilize This Code with a Partner

```
#Figure out what working directory R is currently utilizing
getwd()
```

```
#Where did you download the data (i.e., desktop, download folder, etc.)
#Do you need to change your working directory to ensure that you can load the data?
setwd("insert your working directory here")
```

```
#Once you have set your working directory, make sure to read the data into R
#I just changed the name of the file to "caribbean_hurricanes" feel free to name the data set whatever
carib <- read_csv("caribbean_hurricanes.csv")
```

```
glimpse(carib)
```

1. What is going on with the first row? Let's remove it.

```
#We are going to delete the first row of the data
#carib[rownumber, columnnumber]
carib <- carib[-1, ]
#
#Check to see if it worked
View(carib)
```

Do we notice any issues with the data set?

```
View(carib)
```

The first thing you should notice is that most of the variable types are still characters. Additionally, most of the variable names are too long and complicated.

```
class(carib$`Island Name`)
```

2. What does R mean when it says a variable (in this case the column for "Island Names") is a character class?
3. What do you notice about the column "Island Names"?

```
carib$`Island Name`
```

4. What is the code below showing you?

```
carib$`Island Name`[1:5]
```

5. Which island names are represented in this dataset?

```
unique(carib$`Island Name`)
```

Let's create a plot

```
ggplot(carib, aes(x = `Input Date (yyyy/mm/dd)`)) +
  geom_bar() +
  coord_flip()
```

Reformat the date variable so that they appear as dates with a consistent format

```
install.packages("lubridate")
library(lubridate)

carib$`Input Date (yyyy/mm/dd)` <- ymd(carib$`Input Date (yyyy/mm/dd)`)
#
#Now we are going to create a plot of dates
ggplot(carib, aes(x = `Input Date (yyyy/mm/dd)`)) + geom_bar() + coord_flip()
```

6. Rename the "Area of Expertise" and "Assessment" variables

```
colnames(dataFrame)[colnames(dataFrame) == "oldName"] <- "newName"
# OR
rename(dataFrame, NewName = OldName)
```