Introduction to RStudio Lab

Due: 12am night before lab

Instructions

Work through the problems below, and submit this document as a knitted .pdf file with answers to the Lab 1 assignment on your lab section gradescope.

**In some problems, an incomplete R chunk has been provided. To evaluate the chunk, follow the given instructions, and then change `eval=FALSE` to `eval=TRUE`.

For each problem, put your solution between the bars of red stars.

Introduction

R is a wonderful programming language that we will frequently use in our course to visualize and tidy data, create programs to assist computations, create data models, and perform statistical inference.

To begin, note that there are 3 closely related computer processes that we will work with: *R*, *RStudio*, and *RMarkdown*. It's worth taking a moment to distinguish between them.

- R is a programming language, which can be accessed by typing particular commands into the console or at the command prompt.
- RStudio is an integrated development environment, and provides a clean user interface for writing and executing R code. This is the main tool we will use to interact with the R language.
- RMarkdown is a file type produced by RStudio, which can record R code, text, and graphical outputs. One of the easiest ways to share your results with others is to distribute your RMarkdown (or .rmd) file.

RStudio comes in two varieties: RStudio Desktop and RStudio Server. The desktop version is a program you can install on your personal computer, and uses your computer's resources to perform calculations. On the other hand, the server version can be accessed just using a web-browser, and will perform calculations on Reed's server; there is no need to install any additional programs. For class purposes, we will use the Rstudio Server.

Accessing Lab Assignments

- All lab files will be available on Nate Wells' course website.
- Before downloading the first lab, create a folder called math141 in your Home directory on RStudio.

1. Use your web browser to navigate to the Reed RStudio serve at rstudio.reed.edu.
2. Log-in using your Reed username and password (the one you use for Moodle, etc.).
3. Open the course website and navigate to the Lab page
4. Download the relevant lab. The filetype will be .Rmd.
5. In the Files Pane (typically lower right) of RStudio, press upload and choose the lab you just downloaded to your personal directory (it is called Home on RStudio).
6. Follow the instructions on this new document you just opened.

Now we are ready to begin working in R Markdown. This .pdf file was created from an RMarkdown file!

Main Components of RStudio Lay-Out

- When you first open *RStudio*, your screen will be partitioned into 3 main windows: *Console*, *Environment*, *Files*. Once you create or open a new document, a fourth window for *Source Files* will appear.

Console

- The **Console** can be found in the **left/lower left**. The sideways carrot is called the *prompt* and that is where we can type code we want to run. Click in the console and see what happens when you type the following (make sure to hit "return/enter" after each line):
  - `3 + sqrt(4)/2`
  - `log(4)*5`
  - `mean(1:8)`
- Two useful features:
  - Up arrow: Gives previously used commands.
  - Tab completion: Fills in commands.
- Once you compile an *R markdown* document, you will also have an **R Markdown** tab. If a document won't compile, the error will be there.

Environment

In the **top right**, the **Environment** pane stores all of the items that you load into a session.

Files, Packages, etc.

In the **bottom right**, **Files** tab provides access files in our own directory and in the shared directory. The **Packages** tab is for installing and loading packages, which we do soon. The **Plots** tab will contain graphs we create and the **Help** tab will display help files.

Source files

- The **top left** window displays the content files and objects, such as *R Markdown* documents (abbreviated *Rmd*).
- We will rarely type R code directly into the **Console**. Instead we will type R code into *R Markdown* documents and then will send that code to the **Console**. We will practice doing this soon.

Find this spot in the Rmd file so we can work directly in the document for the rest of the hand-out.

Intro to R Markdown

- Why are we using R Markdown (instead of Word, a Tex editor, etc)?
  - It allows for a fully reproducible workflow. This makes it easier to share your work with others and to keep track of changes.
  - It is fairly easy to learn.
  - You can insert LaTeX code: $z = \frac{x-\mu}{\sigma}$.
  - You put everything (code, output, write-up) all in one document and then when you compile (i.e. knit) the document, you get a pretty output file.

R code and output

- To include R code and output, we need to create an **R chunk**. Here's an example R chunk:

```
# Compute the mean
mean(1:10)
```

```
## [1] 5.5
```

```r
# Create a vector, called x, of the numbers 1 to 8
x <- 1:8
x
```

```
## [1] 1 2 3 4 5 6 7 8
```

- Notice that the three back ticks with `r` in curly brackets start the R chunk and three more back ticks close the R chunk.
- In the upper right-hand corner of the R chunks, there are three items. The first allows you to set more options. The second runs all the R chunks above the one you are working in. The third runs the current chunk. Other options for running code:
  - put the cursor on the same line as the code and hit Command+Enter (for a Mac) and Control+Enter (for a PC).
  - put the cursor on the same line as the code and hit the Run button.
- You can add your own chunk by clicking on the green box with a "C". Try that now. Then run code to evaluate 1 + 2.

Data types- a brief intro

- So far you have made a numeric vector `x`. There many other types of data objects you can make in R.

The following code chunk makes a character called favorite_disney_movie. Think of characters as text as opposed to numerical values. Note that I told R that this was a character by putting quotation marks around Frozen_II.

```r
favorite_disney_movie <- "Frozen_II"
```

Knitting

- Find the "Knit" button and click it. You may get a message to install some Markdown packages. Allow it. You may also get a message about allowing popups. Allow them.
  - Examine the output file.
  - Notice that a **new** file appeared in your Files tab. This is the output document.
- Click on the arrow next to **Knit** and notice that you can compile an Rmd file into a pdf, html, or Word document.

Workflow

- Modify the Rmd file.
- If the modifications are code, run the code in the console to debug.
- Knit.
- Look over the output document.
- Repeat.

Basic Formatting

- One of the best things about R Markdown is how simple the formatting is. We can make things **bold** or *italized* easily.
- To create a list, just start numbering.

1. First thing.
2. Second thing.
3. Third thing. Knit and notice that I can mess up the numbering but it will still be correct in the output document.

- To make headers, use #. More #'s decreases the size.

Big Header

Moderately Big Header

Not-so-big Header

- We can include a hyperlink by putting the words we want to appear in square brackets and the webpage in round brackets. Here's an example linking to the R Markdown page.

Packages

- Base R contains LOTS of useful commands to analyze, describe, and manipulate data. However, other people have developed new commands which are not packaged in the standard version of the program. Therefore, people create packages which contain their new commands. We will use the `dplyr` package quite often in this class so it is time to learn how to load a package.
- The following chunk loads the `dplyr` package into your R Markdown document. You also need to run the code if you want `dplyr` to be loaded in your RStudio session.

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

- Go to the **Packages** Tab. Notice that `dplyr` is listed and the box next to `dplyr` is checked if it is loaded.
- You may want to use a package that is NOT listed in the Packages Tab. In that case, you will need to installed it using the "Install" button. Once a package is installed, it will show up in your packages list.

Loading Data

- For most labs a first step will be to load in a dataset for analysis. Below is an example of loading a dataset of all recorded volcanic eruptions:

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.6.2
```

```
marathon <- read_csv("https://reed-statistics.github.io/math141s22-wells-website/data/GVP_Eruption_Resul
```

```
## Warning: 3 parsing failures.
##  row                col           expected actual
## 4468 EndYearUncertainty 1/0/T/F/TRUE/FALSE     5  'https://reed-statistics.github.io/math141s22-wells
## 4651 EndYearUncertainty 1/0/T/F/TRUE/FALSE     2  'https://reed-statistics.github.io/math141s22-wells
## 7304 EndYearUncertainty 1/0/T/F/TRUE/FALSE     10 'https://reed-statistics.github.io/math141s22-wells
```

- Run the code to load the data into the RStudio Session. Ignore any parsing failures error messages.

Stored Objects

- You should now have two objects in the *Environment* tab: **eruptions** and **x**. Notice that the number of observations and number of variables are listed for datasets. Click on **eruptions** to view the dataset.

Avoiding Problems

- Knit often! Knitting saves your work and will tell you if there is a bug.
- If the document won't knit...
  - Your Rmd file must be self-sufficient. Therefore, if you read in a file in the console but didn't put the `read_csv(...)` command into your Rmd, then it won't knit. If you defined a variable in the console and not the document, it won't knit. Each time a document knits, it only uses the functions and objects provided in the document.
  - Make sure you didn't put "interactive" functions such as *View(eruptions)* and *?mean* in the Rmd file.
  - Check that you changed `eval=FALSE` to `eval=TRUE` in R chunks where you filled in the code. It may be that an object created in the unevaluated chunk is needed in a later R chunk.

Lab Exercises

To complete this lab, work through the exercises below, and submit this document as a knitted .pdf file with answers to the Lab1_Intro_RStudio assignment on Gradescope.

If you need to answer an Exercise with text, type the text below the header, on the next line, in the white part, and if you need to answer an Exercise with some code, insert a code chunk below the header, and put the code in the greyed out box.

Exercise 1

Answer the following with code in a code chunk (no text necessary). Remember that the code is just instructions for R. You need to run the code chunk to make R execute those instructions!

- Create a variable called y with the value of 7
- Multiply x by y, and store the answer in a variable named z like so: z <- x * y
- You should now see favorite_disney_movie, x, y, and z all in your Environment pane

Exercise 2

- Run the following mathematical operation in a code chunk: 6 + 3
- Where does the answer appear? (please answer with text)

Exercise 3

- Now add a code chunk, and save the results of 6 + 3 as a variable called a.
- Does the answer appear? (please answer with text)
- Where does the object a show up? (please answer with text)
- Next type a into the code chunk and re-run the code chunk. What happens? (please answer with text)

Exercise 4

- Run following command in a new code chunk. a^2.
- What does the ^ operator do? (please answer with text)

Exercise 5

- Type the following command into a new code chunk. sum(a, x, y)
- sum is a function. Based on the output, what do you think the sum function does? (please answer with text)

Exercise 6

- Click the little broom icon in the upper right hand corner of the Environment pane. Click yes on the window that opens.
- What happened? (please answer with text, and don't freak out)

Exercise 7

- Go to the Run button at the top right of the R Markdown pane, and choose Run All (the last option)
- What happened? (please answer with text)

Exercise 8

- Recall the vector `x` we created earlier. Copy, paste and run the following in a code chunk. What does this code accomplish? (please answer with text)

```
x + 2
```

Exercise 9

- Copy, paste, and run the following code to make a vector called music, that contains music genres. Recall a vector is a data object that has multiple elements of the same type. Here the data type is a character. Look in the environment pane. How does R tell us that this vector contains characters, not numbers? (please answer with text)

```
music <- c("bluegrass", "funk", "folk")
```

Application

In this problem, we are going to explore a data frame in R.

a. Run the following code to load the `pdxTrees` library/package.

```
library(pdxTrees)
pdxTrees_parks <- get_pdxTrees_parks()
```

b. Within the `pdxTrees` library, there is a data frame called `pdxTrees`. Run the following code to view the data and look at the help file for the data.

```
View(pdxTrees_parks)
?get_pdxTrees_parks
```

What does a row of the dataset represent?

c. List three quantitative variables found in `pdxTrees`.

d. List three categorical variables found in `pdxTrees`.