R Lab 1: Introduction to R and R Notebooks

Learning objectives

In this lab, you will learn how to do the following: 1. How to set up an R Notebook and how to organize R Notebooks on your computer 2. How to install packages 3. How to enter R code in **chunks** 4. How to enter notes to yourself in **annotations** 5. How to do basic R commands 6. How to create objects to hold lists of numbers using the `c()` command

Introduction

In this class, we will be using R, along with R Studio to analyze data. RStudio is an IDE (interactive development enviroment) that runs with R and makes it easier to use. Before you start this lab, you should install R and RStudio on your computer and ensure that it is running.

One of the biggest issues in data analysis, across any science, is reproducibility. It is one thing if a researcher has trouble reproducing another researcher's results. But what if a person can't reproduce their own results?

In grad school, I would do analyses and get results, and then have no idea how I could get those same results again. Or I had no way to know how I got from my raw data to the finalized data after cleaning and editing it. I thought I was following the same steps, and doing the same things, but I didn't know what I was doing.

Another issue is that I could have been making mistakes and not knowing it. I made several mistakes that I caught, but what if there were others? I took good notes, but what if I clicked the wrong button and didn't know it.

One of the strenghts of R is that you can create a text-based log of all your analyses through R Notebooks. RStudio makes this easy, allowing you to combine your R commands and your notes for your commands into a single document that you can export.

This assignment

For this assignment, I want you to create a basic R notebook, place some code in chunks and some notes in annotations, and then to knit it.

Organizing Assignments and Creating an R notebook

The way I suggest you organize your assignments is to create a folder for this class on your computer, and then create a subfolder inside that folder that holds each assignment. R notebooks create additional files when they are knitted and the folder for each assignment will hold these folders in an organized place. In addition, you can place any data files required for each assignment in that folder.

Go ahead and create a folder for this class if you have not already done so and in that folder, create a subfolder that is named 'Lab 1'.

Now we will create an R notebook. Go ahead and open R Studio. Open a new notebook file by clicking File -> New File -> New Notebook. The first time you open an R notebook, RStudio may ask you to install several packages. Go ahead and accept this and wait until it is finished. After you create the R Notebook, RStudio creates a template notebook with a lot of information there. Delete everything except the first few lines with the front matter. This is the portion that is between the triple dashes '—'.

Now, save this new notebook on your computer in the folder you created for this assignment by clicking File -> Save As.

In the new notebook that opens, change the title in the front space to a title which is appropriate for this assignment

Installing Packages

The first chunk I have in any R Notebook is where I load packages. A package (or library) is an additional add-on to R which allows R to have more commands. We will use the `tidyverse` package in other assignments. However, we will go over how to load package in this assignment.

Before you use a library, you must install it first. You can do this by clicking the packages tab and then choosing "install". Then this will pop up a dialog box where you can install the package. Type the name "tidyverse" in the box that appears and then choose install. There will be a lot of message that appear in your console. If there are any messages that ask you to make any choices, just follow the prompts and accept them.

You only have to install a package once. However you will have to load the packages you need each time you do an R notebook.

Before entering a code chunk, create an annotation. This is just typing text that says what you are doing. You should type something like "loading packages". All the text that is not in chunks will not be run and only exists as notes to yourself.

Now go ahead and create a code chunk if you have not done so. Make a code chunk by going to Code -> Insert Chunk (or pressing "Control-Alt-I" (Windows) or "Command-Option-I" on Mac). In that code chunk, type `library(tidyverse)` just like indicated below.

```
library(tidyverse)
```

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

Once you type the code chunk, you can run the chunk by pushing the tiny green play arrow in the top-right corner of the chunk. This will give some messages. If you run a code chunk, you will either get an output, an error, or nothing.

R Commands in Chunks

Now we will go over doing some basic R commands. When I give you the R commands that are highlighted below, please place them in the chunks and run them. Answer the questions by typing annotations in between the code chunks. You should always annotate your code so that you know what each chunk is doing.

Type the following command as a code chunk. What do you get when you run it?

```
1
```

```
3+4
```

```
x
```

For the first two commands, you should get an output. If you input a number, you get that number as an output. The last command should give you an error. That's because R doesn't know what `x` means. We'll fix that in a minute.

Go ahead and delete the code chunk that gives you an error. When we knit the notebook, any code chunks that give errors will cause the knitting to fail.

When you see output, you may see a number in brackets, like if you type the output below:

```
3*5
```

```
## [1] 15
```

For now, ignore the number in brackets. This is a way of telling which element is output. The number in brackets helps us if we have a long list of numbers, like if I type the following which lists the numbers 1 through 100.

```
1:100
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##  [18]  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
##  [35]  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51
##  [52]  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
##  [69]  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85
##  [86]  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
```

The number in brackets tells me which value in the list is to the right of the number in brackets. So when I see, [18], I know that the number to the right is the 18th item in the list.

Doing basic math in R

You can use R like a calculator, using math notation. To add, use the plus key. To subtract, use the minus key. To multiply, use the asterisk key, and to divide, use the forward slash key. Exponents are used by the caret key. See below:

```
3+7
4-11
4*21
18/3
2^2
```

Go ahead and type code chunks to do the following:

Now in the space below, create chunks to how you would do the following operations:

9 multipled by 14

32 divided by 3

11 minus 4

Finally, R follows the basic order of operations, with information in parentheses done first, then exponents, then multiplication and division, followed last by addition and subtraction. So the following are different. Try typing each of these as code chunks and seeing the output.

```
3 + 11*5
```

```
## [1] 58
```

```
(3+11)*5
```

```
## [1] 70
```

Objects

The power of R though is that you can do a lot more than that. You can use things called **objects** which are like containers for data. An object can be anything from a single number to thousands and thousands of rows of data. Depending on what an object is holding, it can have different types.

The simplest type of object is a variable. It is simply a letter or word that stands for a number. To make a variable you use the following (replacing the bracketed number with an actual number (like 42).

To assign a value to an object, we simply use equals, with the object on the left and the value to assign on the right.

```
x = 1
```

When we assign a value to an object, we don't get an output, since there is nothing for R to say. To see what an object says, we simply type the object, or look at the environment in the top-right window of the RStudio console.

```
x
```

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

The name of objects can be any text that starts with letters. This can contain letters with numbers or dots in the middle. Objects should not contain spaces. This can be done but is very complex. Objects also can hold different types of data, such as letters, which are called strings. See the examples below. You do not have to type these, but you should at least try to understand what they are doing:

```
number = 4
name = "Michael"
day.of.week = "Tuesday"
question20 = 5
```

We can also do math on objects, just like numbers. Go ahead and type the following.

```
x = 3
y = 7
x*y
```

```
## [1] 21
```

An important thing to note is that objects will get replaced if you reuse them. If I type the following, I am only using one object, and just replacing the value stored there:

```
t = 3
t = 5
t = 11
```

Do the following in R.

1. Create an object that is labeled as your first name and have that object equal your age. Write down what you would type.
2. Make a second object that is the same as the name of the person sitting beside you, and have it equal to their age. If you're not doing this assignment with another person, just pick another person you know and use their name and age.
3. Add the two objects together. What would you type?

The power in R is that objects can have many different values, called **lists**. To make a list, we can use something called a **function**. This is simply a command that tells R that we are making a list, or an object with more than one number. Every function in R has the following format:

```
name(inputs)
```

The function name is immediately followed by a left-parentheses, and in the parentheses is where you put the input or inputs for the function. The simplest function in R is the concatenate function, or c(). It joins all

the **arguments** into one object, called a list. Here are a few examples. Type the following.

```
x = c(1,3,5,7)
ages = c(19,22,37,41,18)
IQ = c(97, 132, 88, 101, 94, 107)
presidents = c("George Washington","John Adams","Thomas Jefferson")
```

Once you type these, notice these objects are stored in the Environment tab of RStudio on the top right. You can click on these to view the objects.

With lists, we can do functions which need more than one number, such as taking the sum of a list, the mean, and so forth. Try the functions below, which give the mean, the sum, and the standard deviation:

```
mean(IQ)
```

```
## [1] 103.1667
```

```
sum(x)
```

```
## [1] 16
```

```
sd(ages)
```

```
## [1] 10.78425
```

4. Create an object named `scores` with the values 23, 47, 39, 36, 41, 32, and 41. Calculate the mean and standard deviation of `scores`.

Creating data frames

Variables are nice, but sometimes we want to hold a lot of organized data together, like a table. In R, these are called dataframes, and they have variables as separate columns with each row being an observation.

Person	Name	Age	Worth
1	Alice	45	$5,000,000
2	Bob	39	$432,000
3	Cara	29	$87,000
4	David	66	$11,000
5	Ethan	54	$33,000

Each row refers to a set of observations, which is a single person. Each column is a different statistic.

Now we are going to make a dataframe. For example, I may want to make a dataframe with how many hours they slept and their number of items they got correct two different 25 question memory tasks.

Name	Sex	Sleep	Memory Test 1	Memory Test 2
Zoe	Female	9	22	19
Yannick	Male	10	17	20
Xavier	Male	6	15	23
Wendy	Female	5	13	18
Vance	Male	5	10	8

To do this, I have to make a list for each column and then combine them using the data.frame() command into the dataframe called "results". To do this, type the following (note I keep all the variable names in lower case; this is my convention but make sure you are consistent with capitalization):

```
name = c("Zoe", "Yannick", "Xavier", "Wendy", "Vance")
sex = c("Female","Male","Male","Female","Male")
sleep = c(9,10,6,5,5)
memory1 = c(22,17,15,13,10)
memory2 = c(19,20,23,18,8)
results = data.frame(name,sex,sleep,memory1,memory2)
```

If you get an error here, it might be because you didn't type each of the lists correctly. For example, if you miss a value, then one of the lists will have 4 values, instead of 5. Then when you put the values into a dataframe, R won't know what to do.

To see the dataframe, I can click on the data frame in the Environment tab. Click on the little icon to the right of where it says `results` that looks like a table. This will bring up another window that shows the data.

Additionally, another way to view a dataframe is to just create a chunk that contains the dataframe itself. If you type the following as a chunk, this will output the data. However, I do not often do this because some dataframes can be very big and I don't want to output the data when I knit the notebook.

```
results
```

If we want to look at a single column or variable in a data frame, we would select it by first listing the data frame name, then a dollar sign, then the column name. To select the column for names, I would type:

```
results$name
```

```
## [1] Zoe     Yannick Xavier  Wendy   Vance
## Levels: Vance Wendy Xavier Yannick Zoe
```

This is important because we can do math on an entire column at once.

We can do math to each of the columns. For instance, if I want to create a new column which is the sum of each of the three tests, I would type:

```
results$memory.total = results$memory1 + results$memory2
```

Now I have created a new column called memory.total. Notice that I used a period in the middle of the column name to separate the words. This is something you can do to avoid using spaces. However, the column name shouldn't start or end with a period.

5. What would you type to select the memory.total column? Add this as a chunk and make an annotation to answer the question.

6. How would you construct a column called memory.mean, which is the mean memory score for the two memory tests. Remember, you could just create this column by dividing the memory.total column by 2.

Knitting the R Notebook

Now you're done with Lab 1. It's time to knit the notebook. Knitting the notebook produces a nice output of your data, including graphs and other analyses. In fact, this lab that you're reading is a knitted R notebook.

When you knit your R notebook, you can do so by clicking File -> Knit Document or choosing the tiny drop-down menu above this menu where it says "Preview Notebook" or "Knit Document". From this menu, you can choose what format you want to use to knit the notebook. I like this option better.

Go ahead and knit the notebook. You will see some text appear in your console. If you see an error, look at the line number where the error is. Then go to that line number and see what is causing the error. There

is probably a typo in your code, or you accidentally deleted some of the punctuation around a code chunk. Sometimes you will have part of a code chunk left or put an annotation in a code chunk. Once you fix the error, try again.

Now you have finished, upload your assignment. If you can't get your document to knit, please let me know, or upload your .Rmd file, which is the text of your saved notebook. Then I can see why it's not knitting.