Lecture 1 Notes

Paul Holaway, Abhi Thanvi

June 21st, 2022

Welcome to Discovering Data Science

Welcome to Discovering Data Science. During the course of the summer, we are going to introduce you to a multitude of different statistical and data science concepts. However, before we can begin doing that, we are going to give you an introduction to `R` and `RStudio`. `RStudio` is an application we use to run `R` (the programming language) and do statistics and data science. Let us begin by familiarizing ourselves with `RStudio`'s interface.

The Windows

`RStudio` by default comes with four windows.

- Coding Window (Top Left)
  - This will only open once you open a file.
- Console/Terminal (Bottom Left)
  - You can toggle between the two using tabs.
  - You should **never** need nor use the terminal in this course.
- Local Environment (Top Right)
  - This shows all the variables, data sets, and functions that are currently stored in `R`'s memory.
    - We will get to custom function later in the course.
- Global Environment (Bottom Right)
  - This shows...
    - Your files
    - Your plots/charts/graphs
    - Your `R` packages
    - The `Help` window if you get stuck

Opening A File

For this course, it is highly recommended that you have a specific folder on your computer where you keep everything for this course. You may name it whatever you like. Put the folder wherever is most convenient for you, whether it be in your `Documents` folder or on your `Dekstop`.

To open a file, go to `Files` and navigate to your DPI folder. Simply click on the file you wish to open. Most of the documents will be given to you blank so you may fill them in as you go, but for your final project you will be starting from scratch. For that you will need to make your own R Markdown file. To create a new

file, go to the top left corner and click on the white square with the green plus sign. You will notice a lot of different options available, however, we will only be using R Markdown files in this course. Click on R Markdown and a new file will open. You will notice some default instructions and information will appear on the file. You may read them, but everything after the r setup code chunk may be deleted.

Console

The console is where you can run code and where output will be shown. However, because we are using R Markdown files, it will allow us to run the code and view the output like the console, so you probably will not use it too much.

Local Environment

This is where you will see all data sets you have imported, variables you have created, and custom functions that you have made. If one of these three things is not created or already in the local environment, then your code will not run. Please make sure if you are using a variable, data set, or function that you either create it before using it or it is already in the local environment.

RStudio Help

`RStudio` has its own built in help option for the different functions and commands. While you can ask your instructors or classmates for help, the built in help feature can assist you as well. You can access the help feature in two ways.

1. By typing `help(functionname)`
2. By clicking on `Help` on the mid-right and typing in your function in the search bar.

Please note that there will be a lot of information so if it is too dense of material do not get discouraged. Try to read it carefully and figure it out. If after doing this you still do not understand what to do, then feel free to reach out again for help in the Discord.

RStudio Packages

`RStudio` on its own can do many things, but sometimes we need help doing things that does not come in the base version of R. To remedy this, we will install packages that can aid us. To do this, go to the `Packages` tab and click `Install`. Then simply type in the package name that you wish to install and click "Install". We will tell you what packages to install (if any are needed to be) at the beginning of either lecture or lab. Do **NOT** download any package unless you are instructed to do so. If you find a package that you think would help you on your project, then consult your instructors first before downloading. `R` is a free open source language so we do not want you do download a virus disguised as a package by accident. It is very rare, but can happen if you are not careful.

Example 1; Downloading A Package

Let's download our first package. This package is called `tinytex`, a package that aids in typing out mathematical expressions and converting R Markdown files into PDFs. First, follow the steps to install the package `tinytex` onto `RStudio`.

1. Click `Packages`

2. Click `Install`
3. Type in `tinytex`
4. Click `Install`

Now, run the following code cell below to finish the installation process. To run R Markdown code cells, you will need to click on the green arrow on the top right hand corner of the cell. You will only need to run this once.

```
library(tinytex)
install_tinytex()
```

```
## tlmgr install grffile
```

One thing to note is the command `library()`. This function is what will tell `RStudio` that you want to use a specific package. Every time you close `RStudio`, all packages that are not on by default will have to be re-started. If you are using a function that is part of a specific package, do not forget to run the `library()` command with the package name every time you open `RStudio`.

Example 2; Basic Calculator

Let's move onto coding in `RStudio`. `RStudio` is basically a fancy calculator so you can use it to do many of the same things a really expensive TI-84 can.

Let's start by adding `2 + 2`. Don't forget to run the cell by clicking the green run arrow.

```
2+2
```

```
## [1] 4
```

Now try `2 - 2`.

```
2-2
```

```
## [1] 0
```

Now multiplication using `2 * 2`. `*` is the multiplication symbol.

```
2*2
```

```
## [1] 4
```

Division is done using the `/` symbol. Try `2 / 2`.

```
2/2
```

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

Now you can test out order of operations. Let's do $8*(9+6)$ and $2^2*(3+5)$. To raise a number to a power, use the `^` symbol.

```
8*(9+6)
```

```
## [1] 120
```

```
2^2*(3+5)
```

```
## [1] 32
```

What if you wish to take the square root of something? That can be done simply using the `sqrt()` function or by using exponent rules and raising the number to the $\frac{1}{2}$ power. Let's calculate $\sqrt{4}$ using both methods.

```
sqrt(4)
```

```
## [1] 2
```

```
4^0.5
```

```
## [1] 2
```

Example 3; Printing

There may come a time you would like to print text. If this happens, then you need to use the `print()` function. You will most likely not need it for this course that much, but it is still something you should know. Let's do the CS stereotypical printing of "Hello World". For this, you will need to have " " around everything you want printed. Otherwise it will not print.

```
print("Hello World")
```

```
## [1] "Hello World"
```

The `print()` function is only for printing text. If you wish to print a value, `RStudio` will by default print all values calculated that are not equated to a variable. If you wish to print the value of a variable, type in the name of that variable. We will see this later.

Example 4; Creating Variable

Sometimes we want to store a value so that we can use it later. We can do this by creating a new variable and assign a value to it. You may name it just about anything you want. `RStudio` is case sensitive so `A` and `a` can be assigned different values. Let's try doing this a few times.

```
A = 1
a = 0
```

Notice how these two now show up in your local environment with the values you assigned them. You can now use these variables in place of typing in the numbers. Now it may not seem like a big deal, but later on when you are dealing with less than nice numbers, this will be helpful. Let's now move onto working with variables.

Example 5; Working with Variables

Let's calculate $a^2 + b^2$ using variables.

```
#Assign the Variables
a = 2
b = 5
#Calculate the expression
a^2 + b^2
```

```
## [1] 29
```

One more thing of note here is the text after the # symbol. By typing # and typing in text afterwards, you are creating a comment. A comment is a note to yourself in your code that is ignored by the computer. Feel free to type as many or as little comments in your code as you feel necessary.

Example 6; Doing Complex Expressions with Variables

Below we are going to do a more complex expression using variables. It is common to run a series of lines in a single code cell that will build up a complex computation in stages, naming the intermediate results.

You have just gotten a job for the summer. They are paying you $12 an hour. It is a part time job so you will only be working 20 hours maximum per week. For this example assume you will work 20 hours a week every week. You would like to know how much money (before taxes) you will make this summer. Your current plan is to work two weeks in June, all of July, and two weeks in August. How much money (before taxes) will you make?

```
#Assign the Variables
wage = 12
hours = 20
#Calculate your summer earnings (before taxes)
week = 12 * 20
total = week * 8
total
```

```
## [1] 1920
```

```
#Another Way
week = 12 * 20
June = 2 * week
July = 4 * week
August = 2 * week
total = June + July + August
total
```

```
## [1] 1920
```

End of Lecure 1 Notes