Lab Intro

Paul Holaway, Abhi Thanvi

June 21st, 2022

Contents

Lab Intro

Welcome

You are in your first R Markdown for Discovering Data Science. These allow us to combine text with `R` code. You will complete a R Markdown file like this for all of the labs and later your project. Just like learning a new spoken language, you will not learn the language without practice. Labs are an important part of this course. Collaboration on labs is **extremely encouraged**. If you find yourself stuck for more than a few minutes, ask a neighbor or course staff for help. When you are giving help to your neighbor, explain the **idea and approach** to the problem without sharing the answer itself so they can figure it out on their own. This will be better for them and for you. For them because it will stick more and they will have a better understanding of the concept. For you because if you can explain it to other students, that means you understand it better too.

Types of Text

R Markdown files contain three different types of text.

1. **Markdown Text**, which is just like typing into a word document.

- Any text with `**` around it will appear bold when converted to a PDF.
- Any text with `*` around it will appear in italics when converted to a PDF.
- Any text with `~` around it will appear crossed out when converted to a PDF.
- Any text with ` around it will appear in coding font when converted to a PDF.

2. **LaTex**, which is a mathematical expression language.

- You are **NOT** required to learn LaTex, but you may outside of the course if you wish.
- Any text with `$` around it will appear in LaTex.
- Hover over the expression with your cursor to see what it is. Two examples are below.
  - $sin(2x)$
  - $\frac{1}{2}$

3. **Code**, which will appear in a cell like the one below.

```
#This is a code cell.
```

Remember that to run a cell you must click the green arrow in the top right-hand corner of the cell.

Code Cells

R cells are cells that contain `R` code. When you "run" a cell, you will run a snippet of code. `R` is a statistical programming language that was designed to be easy and simple to use, and the R Markdown files make it even easier and more convenient to use. The most basic operations that a computer can do are simple math. The next cell is your first `R` cell, which we already asked the computer to calculate `5 + 4`.

```
5+4
```

```
## [1] 9
```

Notice above that the output is printed out below automatically. Now you try it.

Problem 1

Part 1: Addition

In the next cell, use `R` to compute the sum of 3 and 8.

```
3 + 8
```

```
## [1] 11
```

Part 2: Multplication

In the next cell, use `R` to compute the product of 7 and 5.

```
7 * 5
```

```
## [1] 35
```

Part 3: Powers

In addition to addition (`+`), subtraction (`-`), multiplication (`*`), and division (`/`), `R` can also raise a number to a power (2^4). Recall that the symbol to do this is `^`.

```
2^4
```

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

Part 4: Order of Operations (PEMDAS)

Using everything you know, use the following cell to compute the value of $(3+5)*9^2$.

```
(3+5)*9^2
```

```
## [1] 648
```

Problem 2

`R` by default will print anything that is not being assigned to a variable, but if you want to print variables, you will have to tell it to.

```
x = 2
```

Notice how when you run the cell above that it does not print out the value of `x`. It only stores it in the local environment. Now let's print out the value of `x`. This can be done by typing out the name of the variable after assigning it.

```
x = 2
x
```

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

There, now it prints. However, what happens if you want to print out letters or words? Letters or words are called "strings" in `R` and for these you will need to use the `print()` function. Let's do the CS cliche of printing out "Hello World".

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

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

Notice how the words are typed in `" "`. This is because you have to tell the computer that what you are typing is a string. Quotation marks are how the computer knows what you are typing are strings.

Part 1: Printing Strings

Using the `print()` function, print your name in the output of the next cell.

```
print("Paul C. Holaway")
```

```
## [1] "Paul C. Holaway"
```

Part 2: Printing Numbers

Calculate the three values. Do **NOT** assign them to a variable name. - $(4 * 3)^2$ - $8 - 3$ - $(100/4) + \frac{1}{2}$

```
(4*3)^2
```

```
## [1] 144
```

```
8-3
```

```
## [1] 5
```

```
(100/4)+1/2
```

```
## [1] 25.5
```

Problem 3

In natural language, we have terminology that lets us quickly reference very complicated concepts. We don not say, "That's a large, slow, mammal with brown fur, and three toes.", we just say "sloth". Similarly, an effective strategy for writing code is to define names for data as we compute it. In `R`, we do this with assignment statements. An assignment statement has a name on the left-hand side of an `=` sign and an expression to be evaluated on the right.

```
ten = 3*2+4
```

When you run the cell, `R` first evaluates the first line. It computes the value of the expression `3 * 2 + 4`, which is 10. Then it gives that value the name `ten`. At that point, the code in the cell is done running. After you run that cell, the value 10 is bound to the name `ten`.

```
ten
```

```
## [1] 10
```

The statement `ten = 3 * 2 + 4` is not asserting that ten is already equal to `3 * 2 + 4`, as we might expevt by analogy with math notation. Rather, that line of code changes what `ten` means. It now refers to the value 10, whereas before it meant nothing at all. You may name your variables anything that you like for example, you could do `fish = 3 * 2 + 4`.

```
fish = 3*2+4
fish
```

```
## [1] 10
```

Part 1: Variables in Expressions 1

You may use a variable that you have already created in making a new variable. Make a new variable using whatever name you like that is `ten * fish`. Print out the result.

```
whatever = ten*fish
whatever
```

```
## [1] 100
```

Part 2: Variables in Expressions 2

Now do the same thing as in 3.1, but have the variable be `ten / fish`. Print out the result again.

```
whocares = ten / fish
whocares
```

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

Another common practice is that a series of lines in a single cell will build up a complex computation in stages, naming the intermediate results (like the paycheck example in the Lecture 1 notes). Here we will calculate the volume of a cylinder knowing it has a radius of 1.5 inches and a height of 3 inches.

HINT: π is already stored in `R`, you can use it by typing in `pi` for your calculation.

```
#Assign Variables
radius = 1.5
height = 3
#Calculate Volume
Volume = pi * radius^2 * height
Volume
```

```
## [1] 21.20575
```

Some things to note about naming variables in R. They may have upper or lower case letters, (R counts upper and lower case as different), underscores (_), and numbers. However, the first character of the variable cannot be a number. It confuses R. Names also cannot contain spaces since spaces are used to separate pieces of code from each other. The code below does the same thing as the code above, just with different variable names.

```
#Assign Variables
r = 1.5
h = 3
#Calculate Volume
V = pi * r^2 * h
V
```

```
## [1] 21.20575
```

Part 3: Seconds in a Decade

Assign the variable sec_dec to the number of seconds between midnight January 1st, 2010 and midnight January 1st, 2020. Print out the result. - You will want to think about how this is computed before you start writing code. A pencil and paper work great. - Remember that leap years exist.

```
#Assigning Variables
sec = 60
min = 60
hr = 24
day = 365
#Calculations
year_norm = sec*min*hr*day
year_leap = year_norm + sec*min*hr
sec_dec = 8*year_norm + 2*year_leap
#Answer
sec_dec
```

```
## [1] 315532800
```

Submission

Once you have finished your lab...

1. Go to the top left and click File and Save.
2. Click on the Knit button to convert this file to a PDF.
3. Submit **BOTH** the .Rmd file and .pdf file to Blackboard by 11:59 PM tonight.