Week 0 Discussion (Intro to Jupyter 1)

September 27, 2021

1 Group Number:

Name:

Name:

Name:

Name:

Name:

Name:

Roles

Scribe (2), Facilitator (5), Manager (4), Reporter (3), Contributor (1), Contributor(6)

2 1A: Introduction to Jupyter Notebooks

2.1 1.0 - What is a Jupyter Notebook?

You have been learning R in the textbook using *DataCamp* exercises and the *DataCamp* sandbox. But real statisticians and data scientists don't use *DataCamp* in their work. They use either RStudio or Jupyter Notebooks. That's why we'll be learning Jupyter Notebooks too!

So… what is a Jupyter Notebook (JNB)? - It's kind of like a word document made up of different blocks called "cells". - There are two kinds of cells: CODE cells and MARKDOWN cells. - The nice thing about a JNB is that when you run some code, the output will appear right below it. - They are a great way to showcase and share your data analyses with others!

Click on this text. This is a MARKDOWN cell. If you double-click inside this cell you can edit it!

Try it! Write your name and your favorite food here:

(After writing, try hitting SHIFT + RETURN. You could also hit the "Run" button at the top of the notebook.)

[ ]:
```
# This is a CODE cell!
# You can use it as a calculator or for running R code

4 * 25
```

```
# Click inside this code cell
# Then press SHIFT + RETURN or "Run"
```

Notice that two things happened when you ran the code cell: - The output of the code (100) appeared right below the code. - The `In [ ]` changed to have a number inside of it.

Try running the cell again. How does it change? What do you think the number inside `In [ ]` means?

Notice that when you are editing a MARKDOWN cell, it sometimes looks like code (because the font is different), but it is not the same as R code. It's just regular writing.

But MARKDOWN and CODE cells are similar because both kinds of cells need to be "Run" (with the "Run" button or with SHIFT + RETURN).

2.2 2.0 - What are JNBs for?

Since data science is about answering interesting and important questions using data, these notebooks are a great way to give a report on your data that you can share with others!

A good notebook is NOT just a bunch of code and numbers and graphs.

A good notebook has regular writing (mixed in with the code, numbers, and graphs) to help people understand the questions you are trying to answer and what you did to answer those questions.

2.3 3.0 - Adding new MARKDOWN and CODE cells

3.1 - You can insert a new cell anywhere in the notebook.

Select this cell by clicking on it (the bar along the left side will turn blue).

Then, add a new cell below this one by just pressing the letter **b** on your keyboard.

Try it!

3.2 - What kind of cell did you insert?

Can you figure out how to change it from Code to Markdown (or from Markdown to Code)?

3.3 - You may also want to delete a cell.

Select this cell (side should turn blue). Then delete it by pressing the letters **d d** on your keyboard.

2.4 4.0 - Order Matters

Since a notebook is a way to tell a story, the order of the cells matters, especially for our code cells.

One thing to keep in mind is that all the cells in a Jupyter Notebook are connected to each other. So you want to organize your code cells in order from top to bottom.

Let's try an example.

[ ]:
```
# 4.1
# Read this code. Predict which number will print out.

```

```
mymoney <- 15
mymoney <- 2

mymoney
```

4.2 - Why did it print that number?

Let's say the 2 stands for 2 dollars.

Hm, how much would that be in cents instead of dollars?

[ ]:
```
# Run this code.

mymoney <- mymoney * 100

mymoney
```

Oops! We made a mistake.

Instead of *dividing* by 100, we should have multiplied by 100.

Let's change the division sign (/) above into multiplication (*) and re-run the code cell.

4.3 - Do you get a number closer to what is expected?

Why not?

Well, we still don't get the answer because **mymoney** had our mistake (0.02) in it. So when we ran our code, we multiplied 0.02 with 100… which leads us back to 2.

To fix this, go to **Cell** from the menu bar and then select **Run All**. This will re-run all the cells in the whole notebook in order.

2.5 5.0 - Debugging

Everyone (even experts) will make errors during coding. So a big part programming is debugging code that someone (students, teachers, professionals) wrote.

[ ]:
```
# 5.1
# Run the wrong code first
# Then try debugging it

farm <- c("horse", "goat", "pig", "chicken")
Farm
```

[ ]:
```
# 5.2
# Can you figure out how to debug this one based on the error message?

farm <- c("horse", "goat", "pig", "chicken")
jungle <- c("jaguar", elephant", "hippo")

animals <- c(farm, jungle)
```

[ ]:
```r
# 5.3
# Try this one
# Does this error message help?

farm <- c("horse", "goat", "pig", "chicken")
jungle <- c("jaguar", "elephant", "hippo")

animals <- c(farm; jungle)
```

[ ]:
```r
# 5.4
# How about this one?

farm[1]
animals(1)
```

[ ]:
```r
# 5.5
# What happened here?

farm[1] == anmals[1]
```

The R error messages sometimes use a lot of jargon and are not always easy to understand, so don't worry if they don't always make much sense. Just think of them as clues from the computer that you have to figure out! It's kind of like trying to solve a little mystery.

In the examples above, what are some coding errors you saw? - describe some of - the errors you - saw here

2.6 6.0 - Explore!

If you hover above each of the buttons in the toolbar, you will see text show up to tell you what they are. Try using the buttons to do the following:

6.1 - Save your work (frequently)! Which button would you use for that?

6.2 - Move a cell up or down.

6.3 - Use the buttons to add a cell and delete a cell.

6.4 - Copy a cell and paste it somewhere else in the notebook.

6.5 - What does a **hashtag** (#) at the beginning of a line do in a **markdown** cell? What does it do in a **code** cell?

2.7 7.0 - Closing up

Remember, Jupyter notebooks are for *your use* so you can customize it in a way that is most helpful to YOU! For example, you can add Markdown cells for your own notes or answers; you can write some answers right below questions; you can use comments (that start with #) within code cells to annotate your answers. If any questions come up that you want to ask later, you can write those questions in as well!

And finally, when you exit, *be sure to Close and Halt* the notebook instead of just closing the browser. This will ensure that you do not run out of memory. If too many people forget to do this, these jupyter notebooks will be slow for everyone!

To close a notebook, go to **File -> Close and Halt**.

Maybe say to yourself: **"Don't be at fault, CLOSE and HALT!"**