PSY6422 final self-assessment

Ruri Proto

1. Which topic did I find most useful or interesting? Explain why

Git & Github, and the whole idea of version control - I learned how important having a systematic way to document, update, share and store code can be. It was something I thought only IT engineers needed to know because they work on large-scale projects. How wrong I was. I am sure it is highly important for collaborative projects, but it was really important for me working alone on my data project - it allowed me to see exactly what changes I made, where, and why, and enabled me to recover old versions of my code at relatively dramatic moments where I could not identify the cause of some errors. I think it is important to have a trail not only of my analysis (i.e., the code), but also my steps throughout the process (i.e., the version control system) to allow back-and-forth movements without compromising the integrity of my project or storing lots of versions in a local folder and getting awkwardly lost in them.

2. Which topic did I find least useful, interesting or most difficult? Explain why

Graphs - not difficult, and definitely useful, but not very interesting - largely because we only covered relatively common graphs like histograms, bar charts, etc. I am pretty sure most times relying on a simple, intuitive graph is the way to go to enhance interpretability and promote simplicity. However, these common graphs also lack flexibility and creativity.

I think the greatest thing about plots is the creative element - where you do something that might be a bit more complicated, but attracts attention because it is an unusual way of presenting data. For example, I found out about bean plots and violin plots as alternatives to boxplots and I thought it was the most clever thing ever (and felt a bit ashamed that I never thought about boxplots as being limited in their ability to show the actual distribution of the data). Therefore, what I have tried to accomplish through my project, but did not quite find in the lecture on graphs, is this "surprise" element where one can present data in new, imaginative, unusual, weird ways.

3. What topics do I want to learn more about in the future? Explain why

I want to learn more about how to efficientize code - R has a reputation for being slow, and I noticed that when the script for my visualization is compiled, I need to wait for a few good seconds. This is not a deal breaker, but my data was not that big, so I wonder whether it could handle larger amounts of data without me having to wait a long time for a simple plot. I am aware I did some things that might not be great efficiency-wise (e.g., I ran code from another script in preparation for the current script). The more I think about it, the more I realise a balance might need to be struck between writing efficient code and readable code, such that the time saved compiling is not spent trying to understand what the code actually does! And I know there are rules that can enhance R code efficiency (e.g., using vectorised functions instead of for loops), but I have never studied this properly before.

I also want to learn to rely on the command line slightly more - I found it has fantastic flexibility and can do things I never knew it could - like communicating with servers or other computers to enhance processing power. This makes me revisit my old, vague plans to get myself a Linux computer and use command line coding to customise the operating system.

I also want to learn more about Python - and especially how it can be integrated with R to compensate for its shortcomings related to efficiency; perhaps also learn about other things Python can do that R cannot and that I am not aware of yet. I know, for example, that retrieving data via APIs and web scraping are typically done in Python, but I am not sure whether it is because R cannot do it or because R is just cumbersome when doing it. Having a critical eye for the choice of tool should assist me greatly in my future data projects.

And why not, learning about plots commonly used in other disciplines - the circular plot was commonly used in genetics to map associations between genes, and from there it was adopted by geography to reflect migration flows. I am now curious what other seemingly highly specialized plots can be adapted to present data in a novel way in behavioural science.

4. Please share one thing you have read and enjoyed or found useful on the topic of data management and/or visualisation

Definitely this. Learning about circlize after being so used to ggplot2 was really challenging, but I think it brought me some much needed cognitive flexibility. I think being able to use different packages to accomplish similar (or wildly different) things can make my life easier, particularly if I understand the differences between them. For example, I am annoyed with circlize because it has really poor legend-making features, but this is not a problem with ggplot2.

I just find circular plots fascinating now - one can create heatmaps, histograms, barplots, and all sorts of things, plotting them on a circle, while also plotting links between their corresponding entities. These plots just bring to life another layer of complexity in a way that would otherwise be tedious to implement.

On a philosophical note, this project made me reflect on what makes a good visualization.

- one that is straightforward to interpret, but relatively common and with limited ability to illustrate complex phenomena
- one that is somewhat more difficult to interpret without training, but which attracts attention because it is unusual and packs complexity in a visually appealing way

I feel like we're giving too much credit to the former at the expense of the latter. We probably should not dismiss visualizations that are a bit less intuitive, but powerful once we get to understand them.

I know short is sweet, but I can't help mentioning the "here" package - it was mind-blowing to me, because I had never even considered the difference between relative and absolute paths, and how they can interfere with one's ability to run code across platforms and devices.