Homework Assignment 1

Mohammad Mundiwala

Table of contents

1 Setting up Git and Quarto: Tutorial

Git is a powerful tool that allows programmers and data scientists to collaborate on projects both neatly and efficiently. As a novice programmer, it may be difficult to imagine that this is in fact an efficient tool; the primary benefit lies in the users' ability to work on the same project **without** overwriting someone else's work. Quarto is another powerful tool that gives maximum agency to a developers so they can present code, results, and conclusions thoughtfully and professionally with complete reproducibility.

1.1 Setup an SSH authentication between your computer and GitHub

1. Make a GitHub account using your email address
2. If working on a Windows machine, download Git Bash. *You will eventually use this to communicate using Git - Exciting!*
3. Create an SSH authentication key so that you can work on secure project. In Git Bash terminal, type:

Terminal

```
ssh-keygen -t ed25519 -C "Jonathan_Husky@uconn.edu"
```

- Note: The terminal should prompt you to name the SSH key. I recommend using a name that is short and sweet. I used SSH_ids for example.
- Note: You will have the option to create an SSH key without password protection; initially I did this in the effort of saving time but in order to connect your computer to github classroom, it is required that you use a password-protected key.

4. Use the next code snippits to add the SSH key to your SSH agent - this stores the private key you just made in a secure place in the computer's memory.

Terminal

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

5. Now that you must copy the public SSH key and input it into GitHub to complete the connection. It will be a long string that looks like:

Terminal

```
ssh-ed25519
AAAAC3NzaC1lZHUSKY5AAAAICuJ0hqUstatfw6F6Wkids5255EJG2yfsjqN7R7P1+ENGR
Jonathan_Husky@uconn.edu
```

6. In GitHub settings, you will find a green button that says **New SSH key** where you can copy paste the key from your terminal.
7. Finally, use the code snippit below to ensure all the steps were completed correctly.

Terminal

```
ssh -T git@github.com
```

1.2 Install Quarto

1. Download Quarto for Windows
2. I prefer to use VS Code for my IDE so I have it downloaded already. You can simply download it here
3. Check to see if Quarto is correctly installed and accesible. In Git Bash terminal, type

Terminal

```
quarto --version
```

- Note: If you are that you downloaded Quarto yet it still says *command not found*, restarting the Git Bash terminal should fix this issue.

4. Check if you can render the templates/hw.qmd file provided to us by Professor Yan before starting the homework in case there is an issue.

- Note: Copy the file into your working directory and make sure to save it as a .qmd file. I simply saved it with the name **hw_1**.
- Note: Using the terminal in VS Code, I used the following command. You may also use this in Git Bash but terminal in VS code is more accesible at times.

Terminal

```
quarto render .\hw_1.qmd
```

- **Error**. This is because Quarto could not find PyYAML, jupyter, or numpy python packages which are being called in the template script... what now?

5. Venv! It is good practice to create a virtual environement for each project to clearly organize and keep track of which packages are required. This is especially useful when other people want to run your code and need to know what packages to install.

1.2.1 Create a virtual environement

1. In the directory of your project, use the following command:

Terminal

```
python -m venv hw_1_venv
```

2. Once you have created the virtual environement, simply activate it on the command line with:

Terminal

```
.\hw_1_venv\Scripts\activate
```

3. Once it is activated, the virtual environement name will be displayed in parenthesis to the left of the working directory path, in your terminal.
4. Finally, we can download the necessary packages using **pip**:

Terminal

```
pip install notebook numpy pyyaml matplotlib
```

With these packages successfully installed, we have a versatile toolset for python development.

1.3 Create a Line Plot using Quarto:

1. Ensure that the virtual environement is infact activated otherwise the python code will not run. If you cannot recall what you installed, use the following code in the terminal to check.

> Terminal
>
> ```
> pip list
> ```

- Note: If they are all accounted for then go ahead and plot something! You can keep it simple in Cartesian or get fancy… up to you

```
import numpy as np
import matplotlib.pyplot as plt

r= np.arange(0, 2, 0.01)
theta = 2* np.pi * r
theta1 = 2.5* np.pi * r
theta2 = 3* np.pi * r
fig, ax = plt.subplots(
subplot_kw = {'projection':'polar'}
)
ax.plot(theta, r, linewidth = 3, color = 'lightblue', zorder=-1, alpha=0.6)
ax.plot(theta1, r, linewidth = 3, color = 'skyblue', zorder=-1, alpha=0.6)
ax.plot(theta2, r, linewidth = 3, color = 'deepskyblue', zorder = -1, alpha=0.6)
ax.set_rticks([0.5, 1, 1.5, 2])
ax.tick_params(labelsize = 12, zorder=10)
ax.grid(True, zorder=-3, alpha = 0.5)
plt.show()
```

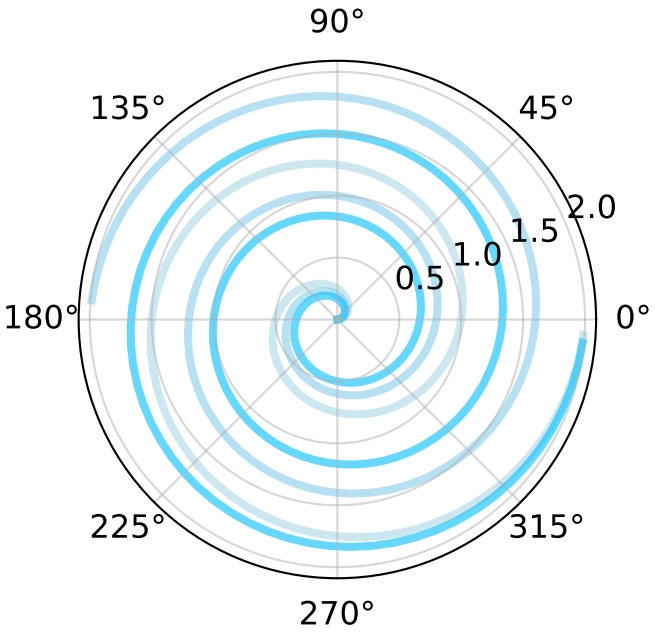

Figure 1: Colorful spirals on a polar axis using simple line plots

1.4 Render this homework submission

The command for rendering a .qmd file is shown below. Make sure you are in the directory of hw_1.qmd file. I had to download a latex package for the PDF; if you already have that, then omit the first line.

Terminal

```
quarto install tool tinytex
quarto render hw_1.qmd --to pdf
```

Finally, we use Git Bash to submit our files. We don't want to upload every single file to our repo; instead a submission of our pdf and the .qmd file for the TA to have as reference.

Terminal

```
git add hw_1.pdf
git add hw_1.qmd
git commit -m "first hw submission"
git push
```

- Note: As always, make sure you are in the directory of your project so that git can access the files you wish to push.

- Note: You can check if this worked by going online to Git Hub classroom. You should see the files which you uploaded.

1.4.1 Minor errors and issues:

1. Make sure there are no typos in the code snippits or main text, the file will not render.
2. When you do fix an issue, make sure to save the changes before running the command in the terminal again.
3. If you want to use an "_" symbol in the text, you must include "\" before the "_" or else it will cause an error.
4. Indentation matters! I spent many hours trying to figure out why my bash code blocks were not rendering as I would have liked and it is because you cannot indent at your own will. Keep the bullets and numbers lists in the same line. This type of error will not cause a coding error but creates an output that is sloppy.
5. An order list like this has no issues. I can use "1." for each bullet and Quarto will automatically order the list. However, I found that this function does not work well when you have code blocks or other bullets in between. I simply counted them up my self to make it simple. I spent more time trying to fix this than it actually takes to number them myself.
6. Bullet lists behave differently! When used an ident for the bullets, the text ran off the page instead of following the format of all other text.
7. Use line breaks in your scripts. We are not wasting any paper so take up more lines in your code to make sure it all fits on your screen.
8. There were other minor issues that came up which I may have forgotten about. I reccomend that you do not write the entire script and then try to render. Instead frequently render your work to make sure things are working as you intend.

All done! Now we are ready to take on anything!!