COMP1204: Data Management
Coursework One: Hurricane Monitoring

Damien Ta
34830294

March 6, 2024

1 Introduction

As a data scientist for the National Oceanographic and Atmospheric Administraiton Centre, I was assigned to the tropical cyclone tracking team. I was tasked with extracting storm data from the tropical cyclone reports and producing maps of where the cyclones have taken place.

This report will talk about how I managed and cleaned the data I had been given from the tropical cyclone reports. I begun by extracting data from the KML files and I inputted the sorted data into a CSV file in the format that had been specified. Specific tags were chosen from the KML file such as Timestamp, Longitude, Latitude, Minimum Sea Level Pressure and Max Intensity. The report is written in LaTeX. Furthermore, I was tasked with also resolving a git conflict which I talk about later on in the report.

2 Create CSV Script

2.1 Introduction

I was tasked with extracting specific data from a set of storm reports. The reports contained lots of information and I coded the **create_csv.sh** script to extract the important information from the reports. The important information that I extracted included **Timestamp, Latitude, Longitude, Minimum Sea Level Pressure and MaxIntensity.**

2.2 How to run the script

Script to run create_csv.sh

./create_csv.sh <input_kml_file> <output_csv_file>

This command runs the create_csv.sh file which takes in a kml file and extracts specific header tag data. The command takes extracted data and saves it to a CSV file.

2.3 Full Script

create_csv.sh

```
1  #!/bin/bash
2  csv_input_path=$1
3  tmp_csv_output=$2
4
5  echo "Timestamp,Latitude,Longitude,MinSeaLevelPressure,MaxIntensity" > $tmp_csv_output
6  dtg="$(grep -R "<dtg>" $csv_input_path | sed 's/.*<dtg>//g' | sed 's/[</dtg>]//g')"
7  grep -R "<lat>" $csv_input_path| sed 's/.*<lat>//g' | sed 's/[</lat>]//g' > lat.csv
8  grep -R "<lon>" $csv_input_path | sed 's/.*<lon>//g' | sed 's/[</lon>]//g' > lon.csv
9  grep -R "<minSeaLevelPres>" $csv_input_path | sed 's/.*<minSeaLevelPres>//g' | /
10 sed 's/[</minSeaLevelPres>]//g' > minSeaLevelPres.csv
11 grep -R "<intensity>" $csv_input_path | sed 's/.*<intensity>//g' | /
12 sed 's/[</intensity>]//g' > intensity.csv
13
14 lat="$(sed "s/$/ N/" lat.csv)"
15 lon="$(sed "s/$/ W/" lon.csv)"
16 minSeaLevelPres="$(sed "s/$/ mb/" minSeaLevelPres.csv)"
17 maxIntensity="$(sed "s/$/ knots/" intensity.csv)"
18
19 paste -d',' <(echo "$dtg") <(echo "$lat") <(echo "$lon") <(echo "$minSeaLevelPres") \
20 <(echo "$maxIntensity") >> $tmp_csv_output
21
22 rm lat.csv
23 rm lon.csv
24 rm minSeaLevelPres.csv
25 rm intensity.csv
26 #I used / to continue command on next line
```

Figure 1: create_csv.sh

2.4 Script Breakdown

2.4.1 Bash Initialization and Passing Arguements into a variable

Bash Initialization and Passing Arguements into a variable

```
1 #!/bin/bash
2 csv_input_path=$1
3 tmp_csv_output=$2
```

#/bin/bash
tells the terminal that when you run the script it should use bash to execute it
csv_input_path=$1
tmp_csv_output=$2
These two lines of code allow inputs to be entered into the script

2.4.2 Using grep and sed to filter data of tag headers

Using grep and sed to filter data of tag headers

```
5  echo "Timestamp,Latitude,Longitude,MinSeaLevelPressure,MaxIntensity" > $tmp_csv_output
6  dtg="$(grep -R "<dtg>" $csv_input_path | sed 's/.*<dtg>//g' | sed 's/[</dtg>]//g')"
7  grep -R "<lat>" $csv_input_path| sed 's/.*<lat>//g' | sed 's/[</lat>]//g' > lat.csv
8  grep -R "<lon>" $csv_input_path | sed 's/.*<lon>//g' | sed 's/[</lon>]//g' > lon.csv
9  grep -R "<minSeaLevelPres>" $csv_input_path | sed 's/.*<minSeaLevelPres>//g' | \
10 sed 's/[</minSeaLevelPres>]//g' > minSeaLevelPres.csv
11 grep -R "<intensity>" $csv_input_path | sed 's/.*<intensity>//g' | \
12 sed 's/[</intensity>]//g' > intensity.csv
```

echo "Timestamp,Latitude,Longitude,MinSeaLevelPressure,MaxIntensity" > $tmp_csv_output
This command prints the tag headers and outputs it to the tmp_csv_output file.
dtg="$(grep -R "<dtg>" $csv_input_path | sed 's/.*<dtg>//g' | sed 's/[</dtg>]//g')"
The code creates variable dtg, we grep the kml file inputted at $1 and this searches recursively for keyword <dtg> and display its occurences. The sed 's/.*<dtg>//g' command would replace the pattern <dtg> and all of its occurences with nothing essentially deleting the characters, this includes any characters before the pattern. The sed 's/[</dtg>]//g' command would replace any pattern between the [], so < / d t g >. All the characters replaced would become an empty spaceessentially being deleted and the g stands for global.
grep -R "<lat>" $csv_input_path | sed 's/.*<lat>//g' | sed 's/[</lat>]//g' > lat.csv
grep recursively searches keyword <lat> from the kml file inputted and displays the occurences of the pattern. The sed 's/.*<lat>//g' command would replace the pattern <lat> and all of its occurences with nothing essentially deleting the characters, this includes any characters before the pattern. The sed 's/[</lat>]//g' command would replace any pattern between the **[]**, so < / l a t >. All the characters replaced would become an empty space essentially being deleted and the g stands for global. The outcome is then appended to a file called lat.csv for later use. This same code is then run on the rest of the tag headers Longitude, MinSeaLevelPressure and MaxIntensity.

2.4.3 Adding units to the end of certain tag headers

Adding units to the end of certain tag headers

```
14 lat="$(sed "s/$/ N/" lat.csv)"
15 lon="$(sed "s/$/ W/" lon.csv)"
16 minSeaLevelPres="$(sed "s/$/ mb/" minSeaLevelPres.csv)"
17 maxIntensity="$(sed "s/$/ knots/" intensity.csv)"
```

lat="$(sed "s/$/ N/" lat.csv)"
New variable lat is created which uses sed and adds on each line ($) " N" from file lat.csv
This is then run on each line of code for longitude, minSeaLevelPres and maxIntensity where each tag header has their respective unit of measure added to end of each line of their lines.

2.4.4 Outputting results to file

Outputting results to file

```
19 paste -d',' <(echo "$dtg") <(echo "$lat") <(echo "$lon") <(echo "$minSeaLevelPres") /
20 <(echo "$maxIntensity") >> $tmp_csv_output
```

paste -d',' <(echo "$dtg") <(echo "$lat") <(echo "$lon") <(echo "$minSeaLevelPres") /
<(echo "$maxIntensity") > > $tmp_csv_output
This line of code concatenates the variables we created earlier into the file at $2.
-d',' would add a delimiter which splits different sections of the tag headers, after each of the inputs. The < allows you to use an output of a command as if it were a file.

2.4.5 Removing temporary files from directory

Removing temporary files from directory

```
22 rm lat.csv
23 rm lon.csv
24 rm minSeaLevelPres.csv
25 rm intensity.csv
```

rm lat.csv
rm would remove the file from the current directory. It does this for all temporary files created, such as lon.csv, lat.csv etc.

3 Storm Plots

Below is the extracted data from the storm plots extracted by the create_csv.sh script. The extracted data is plotted on a world map and the plots are indicated by red circles.

3.1 Storm Plot al112020

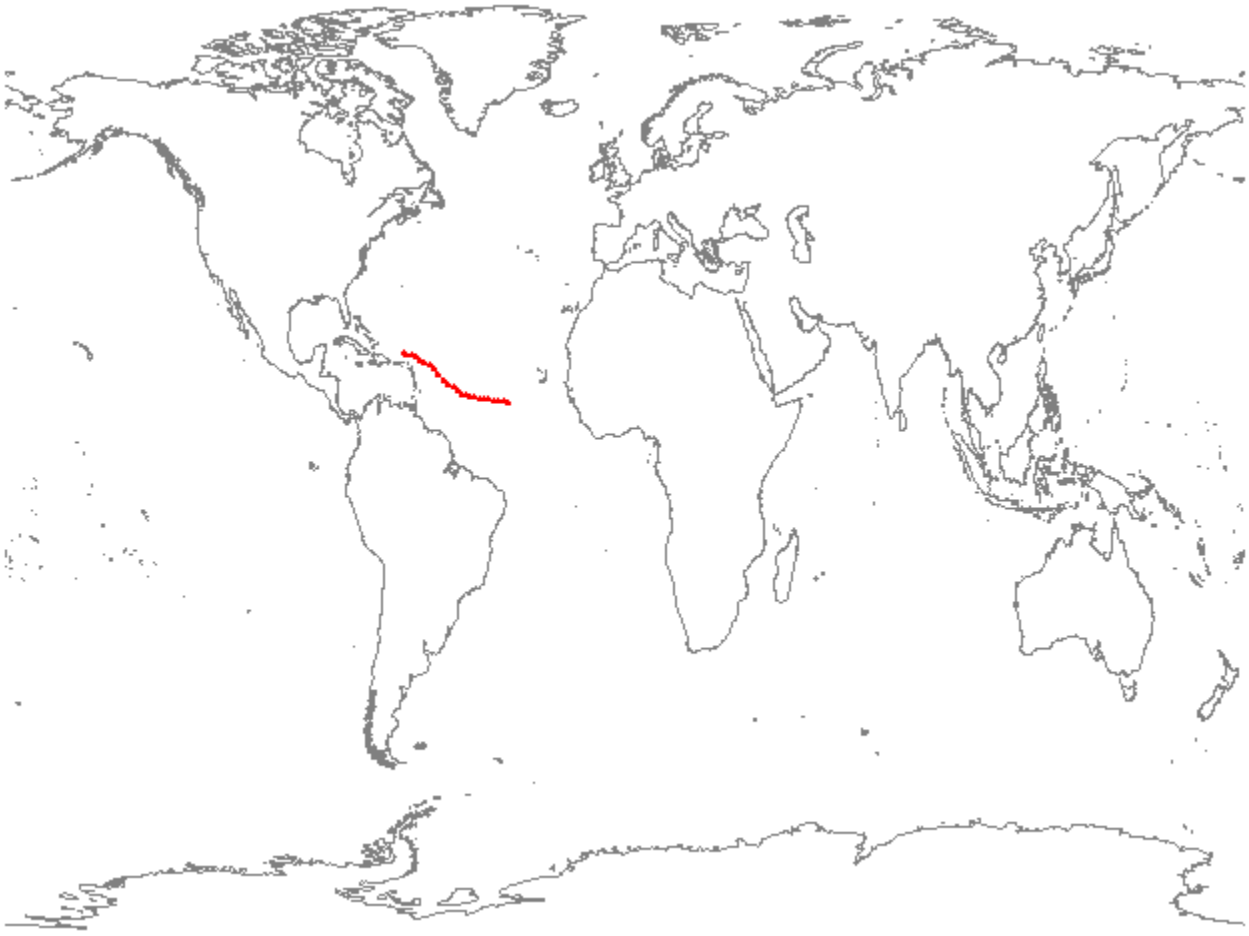

Figure 1: al112020.kml

3.2 Storm Plot al012020

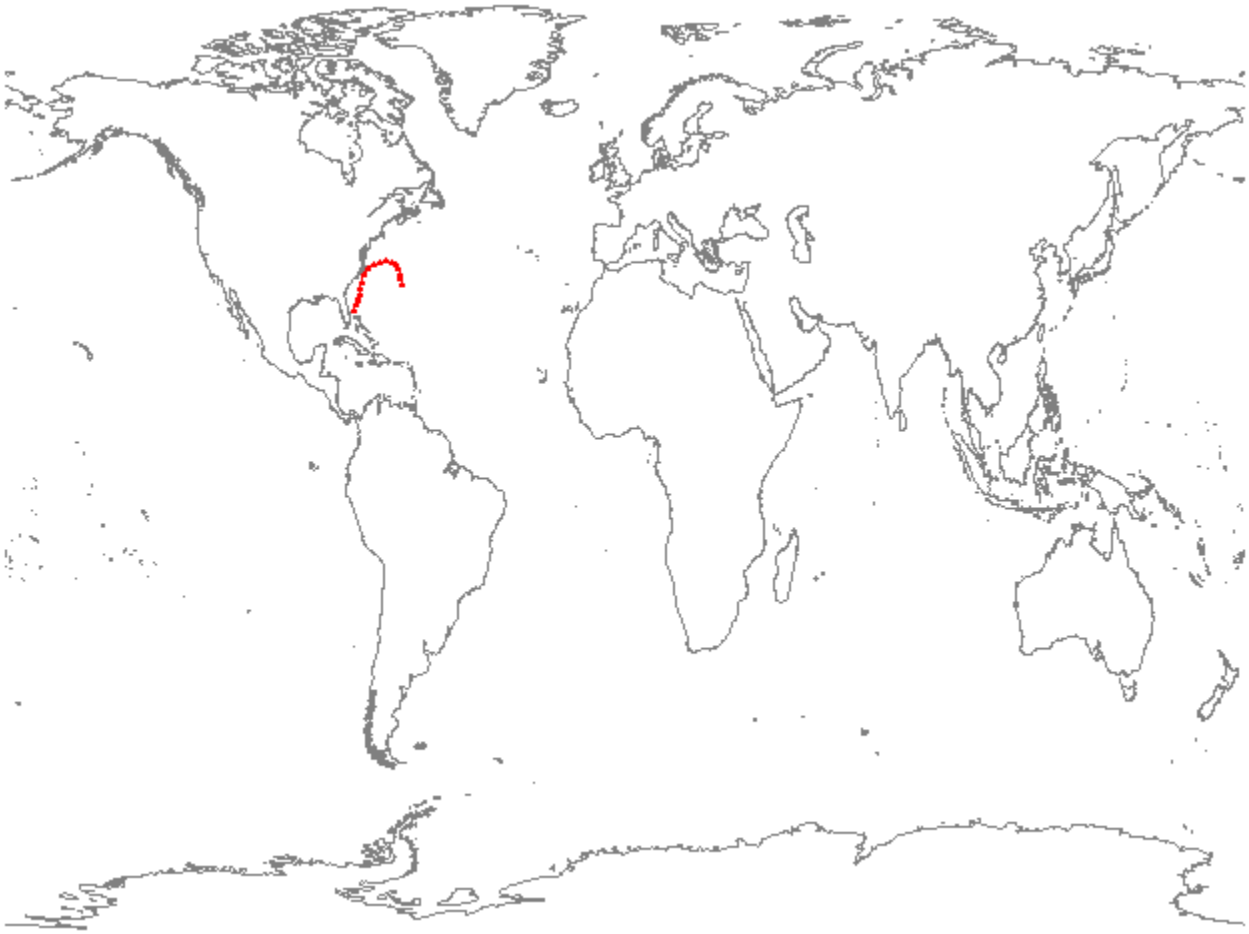

Figure 2: al012020.kml

3.3 Storm Plot al102020

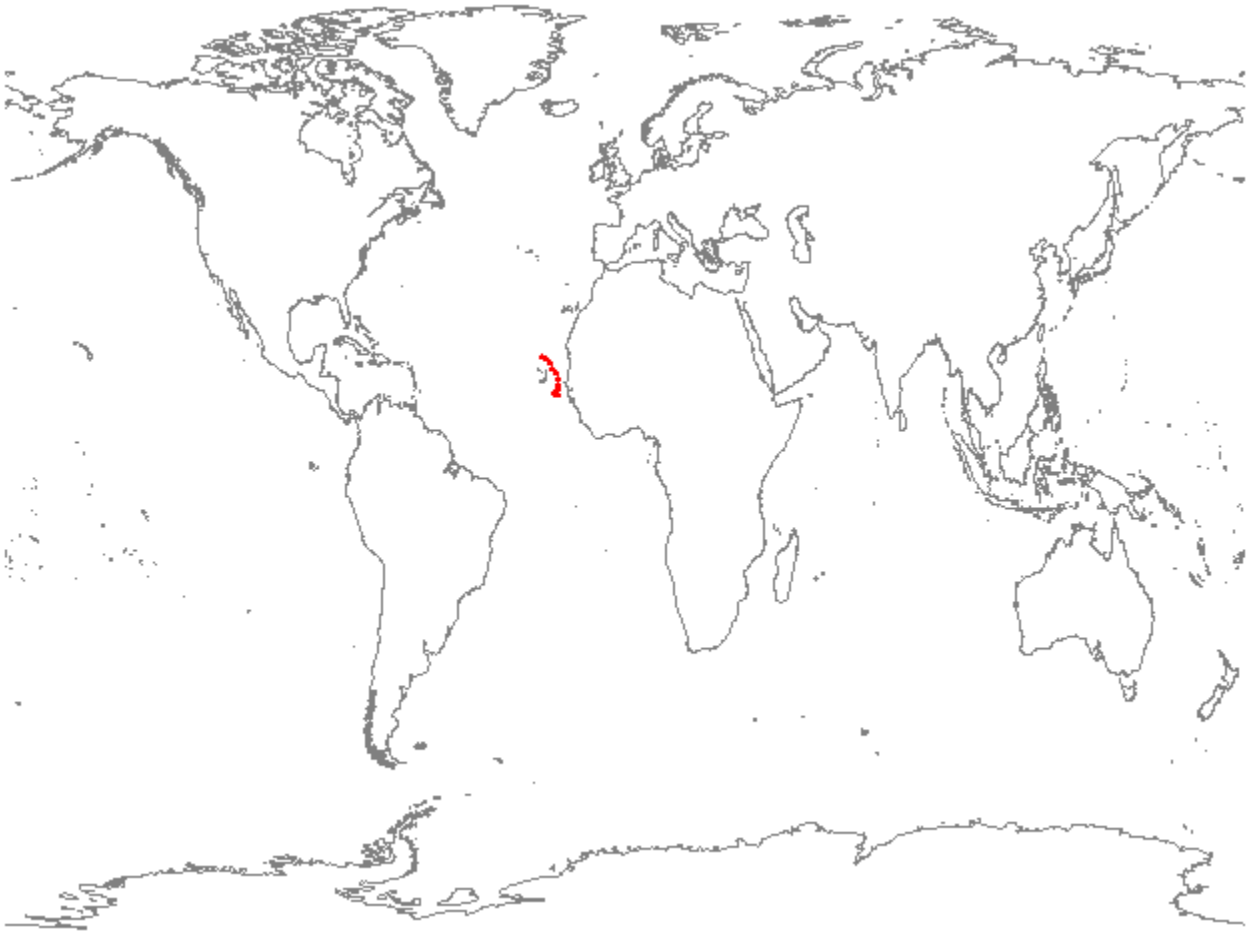

Figure 3: al102020.kml

4 Git Usage

4.1 Introduction

I was required to resolve a git conflict that was caused by running the command below. When attempting to merge both branches, a conflict was created due to the **python-plot-script.py** file being in two different branches.

4.2 How to run the conflict script

The command below ran the script and created a new branch called python-addon and added a file called python-plot-script.py.

```
./conflict-script.sh
```

4.3 Breakdown of how to fix the conflict

4.3.1 Check currrent branch

First thing I did, was check to see if I was on correct branch and to see if conflict script actually ran I saw that **python-addon** was one of the branches meaning the conflict script ran.

```
git branch
```

4.3.2 Merge the python-addon branch to main branch

As the conflict script has already run, we were tasked with merging the **python-addon** branch to the main branch. We did this using the command below. However, when we attempted to merge the two branches a conflict occured.

```
git merge python-addon
```

4.3.3 Resolving the conflict

I opened the **python-plot-script.py** in Visual Studio Code then resolved the conflict by compiling different parts of each scripts into one script. An error was that in the function **plot_all_csv_pressure()** one of the scripts had misspelt for and wrote it as fr. I picked the correct function and moved on. Lastly, the last error I changed was that one of the scripts was missing a whole function called **plot_all_csv_intensity()** which I added to the combined script.

4.3.4 Realising that I had forgotten a part in my python-plot-script.py

After attempting to review what I did and looking over the code again, I had just merged, I realised I had missed an import. The code was all correct apart from missing a part of the code **import math**. As a result, I decided to revert to the commit right before I solved the conflict in order to do the conflict again.

4.3.5 Conflict with readme file

When trying to revert to the commit right before, I had a conflict with the readme file. I attempted to delete the readme to avoid the conflict. I deleted the readme file with the command below.

```
git delete readme.txt
```

4.3.6 Continual conflicts with the readme file

When trying to use the command below, I kept getting a conflict message. I realised that even after deleting the readme file through git and file explorer, it would keep appearing. I decided to revert back to a commit where this error didn't occur, I updated the latest version to the reverted version using the command below and saved a local backup of the version I currently had, just in case something went wrong.

```
git rebase
```

4.3.7 Reverting to older version

In order to revert to a previous version where the readme file wasn't giving me conflicts, I checked my git logs. From there, I found the commit which sounded best to revert back to. I found the commit message code and entered the following code below to go back to the old commit. The old commit I went back to was called "Finished create_csv.sh added functionality to display units". I then reverted back to a way older version to when the readme didn't cause me an issue.

```
git revert 345f39a9
```

4.3.8 Removing the python-addon branch as it wasn't removed

After reverting, I checked my branches and I saw that even after the revert, the **python-addon** branch was still a visible branch. In order to overcome this I ran the code below to remove it.

```
git branch --delete python-addon
```

4.3.9 Running conflict script again and resolving the error

After deleting the python-addon branch, I re-ran the conflict script and gained the same conflict. I resolved it with the same changes I made previously but this time adding the **import math**, making sure not to miss anything.

```
./conflict-script.sh
```

4.3.10 Restoring report.tex and create_csv.sh codes

After attempting to add more to my **report.tex** file and removing some of the unnecessary comments I made in my **create_csv.sh** file, I saw that I had reverted too far back and had a very old version of my code that couldn't run as it was incorrect. When I saw this I decided to use my local backup that I stored and replaced both **report.tex** and **create_csv.sh** files with the correct versions

4.3.11 Adding resolved file

After resolving the conflict, I used the command below to add a change in the working directory to the staging area.

```
git add python-plot-script.py
```

4.3.12 Commititting the changes

Committing the changes would save the changes to the local repository. I committed this with an appopriate message about how the branches have been merged.

```
git commit -am "Merge branch 'python-addon'"
```

4.3.13 Pushing Changes to Remote Repository

I uploaded my local repository content to the remote repository

```
git push origin main
```

4.4 Final Merged Script

python-plot-script.py

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import os
4  import glob
5  import math
6  user_key = 1773
7
8  def plot_all_csv_pressure():
9      path = os.getcwd()
10     csv_files = glob.glob(os.path.join(path, '*.csv'))
11
12     for f in csv_files:
13         storm = pd.read_csv(f)
14         storm['Pressure'].plot()
15         plt.show()
16
17 def plot_all_csv_intensity():
18     path = os.getcwd()
19     csv_files = glob.glob(os.path.join(path, '*.csv'))
20
21     for f in csv_files:
22         storm = pd.read_csv(f)
23         storm['Intensity'].plot()
24         plt.show()
```

Figure 2: python-plot-script.py