COMP1204: Data Management
Coursework One: Hurricane Monitoring

Arjun Srinivasan
34469184

March 8, 2024

1 Introduction

The aim of this work was to code a bash script that created a csv file that contained useful information about storm reports detailed in a kml file that would be input into the script. The csv file to be created would include data about: the UTC timestamp, location, minimum sea level pressure, and maximum intensity, information that was provided in the storm reports contained within the input kml file to the bash script. The csv files containing the neatly formatted information, were used to create storm plots, that can be used to provide a visual representation of each of the storms' location on the world map.

Furthermore, the use of git allowed for regular commits of this report and the bash script. Hence, using version control meant that changes to the bash script and this report were regularly saved on the git repository, and that older versions could be reverted back to easily. Conflicts from an additional bash scripts were also resolved and commits were created about the changes made to the script causing the conflict.

2 Create CSV Script

2.1 Bash Script

```
#!/bin/bash

# Store arguments to script inside the following variables

arg1=$1
arg2=$2

# Following command prints out header line, and writes this to the top line of the output
csv file, which is passed in as the second argument

echo "Timestamp,Latitude,Longitude,MinSeaLevelPressure,MaxIntensity" >"$arg2"



# The following line assigns, all UTC timestamps for each hurricane report in the input kml
file to the variable timestampField

# Firstly the grep command searches for patterns in the input kml file that begin with the
```

```
tag <name>, followed by a UTC timestamp, and ending with the tag </name>, and the -o option
is used to make sure that we find lines that contain exactly the information just mentioned
and nothing extra. This is done in order to remove leading whitespaces from each line.
We can extract the UTC timestamp from each of these lines. The check for this actual timestamp
between the two tags is achieved through the use of regular expressions, where we first search
for zero or more occurrences for numbers in the range 0 to 9 (which acts as the time of the
timestamp), then for the string 'UTC', followed by zero or more occurrences for letters from
A to Z (which acts as the month of the timestamp), and finally again for zero or more
occurrences for numbers in the range 0 to 9 (which acts as the date of the timestamp).
Therefore, all possible patterns produced from this grep command will be every line in the
input kml file, that starts with the tag <name>, followed by a UTC timestamp, and ending
with the tag </name>.

# Secondly, all these patterns are piped into the sed command, that removes all <name>, and
</name> tags from every line, and replaces it with nothing, so we're now left with just the
UTC timestamp. To remove </name> tags we use '#' as the delimiter to remove ambiguity with the
'/' present in this tag.

# Finally, all the UTC timestamps are piped into the awk command, where a comma is added after
every timestamp, to lead to the next field of the output csv file

timestampField=$(grep -o -e "<name>[0-9]* UTC [A-Z]* [0-9]*</name>" "$arg1" |
sed -e 's/<name>//g' -e 's#</name>##g' |
awk '{print $0 ","}')



# The following line assigns, all latitude and longitude values for each hurricane report
obtained from the input kml file to the variable latitudeAndLongitudeField

# Firstly the first grep command uses the -A option to retrieve the next line after every line
in the input kml file that contains the string 'Storm Location', as well as every line that
contains this string. This is done as the latitude and longitude values of each hurricane
report are contained in every line after this string. Each occurrence of these pairs of lines
are piped into the second grep command, which filters out all instances of the string '--',
to remove this from the pattern.

# Following this the pairs of lines (where one line consists of the string 'Storm Location' and
the other is the line containing the latitude and longitude values of that specific hurricane
report) are piped into the following awk command. The lines containing the string
'Storm Location' are removed by checking if the line number is first even, and these are the
lines that contain the location information of each storm (latitude and longitude values).
The next gsub filter is used to remove any whitespaces which are replaced with nothing, and
following this a comma is added to the latitude and longitude values extracted, in order to
lead into the next field.

# Finally all these latitude and longitude pairs of values are piped into the sed command,
which is used to remove any redundant tags before and after the values, and these lines are
piped into the next sed command which removes any whitespace between the value and the unit.



latitudeAndLongitudeField=$(grep -A 1 "Storm Location" "$arg1" |
grep -v '^--$' |
```

```
awk 'NR%2==0 {gsub(/[[:space:]]*/, ""); print $0 ","}' |
sed -e 's/<tr><td><B>//g' -e 's#</B></td></tr>##g' |
sed -e 's/\([0-9]\+\)N/\1 N/g' -e 's/\([0-9]\+\)S/\1 S/g' -e 's/\([0-9]\+\)E/\1 E/g'
-e 's/\([0-9]\+\)W/\1 W/g')



# The following line assigns all values of the minimum sea level pressure measured in
millibars from the input kml file to the variable minSeaLevelPressureField

# The logic to this line of code is similar to the previous one where we obtained the latitude
and longitude values for each hurricane report. This time each line containing the minimum sea
level pressure is after a line in the kml file containing the string 'Min Sea Level Pressure'.
We again use the grep command to get these pairs of lines, and these pairs are piped to a second
grep command to remove the redundant string '--' from some lines.

# Like before, now we pipe the pairs of lines (one containing the string 'Min Sea Level Pressure'
with some kml tags, and one containing the minimum sea level pressure measured in various units
along with some kml tags) to the awk command to remove the line from each pair containing the
string 'Min Sea Level Pressure' as we don't need this information, only the actual value.
Again this is achieved through the use of a modulo check to see if the current line number is even,
as all lines containing the minimum sea level pressure are even. We also remove any whitespaces
which are replaced with nothing (used to get rid of leading whitespaces), and append a comma at
the end of each line to lead into the next field.

# Now we're just left with the line containing the minimum sea level pressure, but this line
contains this value in the unit Hg as well as mb, but we only want it in millibars. These lines
also contain redundant kml tags that we can remove. Hence, all these lines are piped to a first
sed command that removes in the first check, the kml tags <tr><td> which are always before the
relevant data value of the minimum sea level pressure that we need (measured in mb), and the
second check removes all the redundant information following the 'mb' part each line, where this
information includes the minimum sea level pressure measured in the other unit in Hg, followed by
some kml tags. The check for these decimal values measured in Hg is completed through the use of
regular expressions, where we first check for zero or more occurrences of numbers between 0 to 9,
followed by a decimal point, and a further zero or more occurrences of numbers from 0 to 9.
This is followed by the remaining piece of text in the line (containing the string 'inHg' and
further kml tags), and all of this is removed and replaced with nothing as we don't need any of
this information. Therefore, now we're just left with the value of the minimum sea level pressure
measured in mb, but since we removed all whitespaces in all lines containing the minimum sea
level pressure, there will be no space between the value and the unit.

# Because of this we pipe all these lines to another sed command that adds a space between
the value of the minimum sea level pressure measured in milliibars, and the unit (mb).

minSeaLevelPressureField=$(grep -A 1 "Min Sea Level Pressure" "$arg1" |
grep -v '^--$' |
awk 'NR%2==0 {gsub(/[[:space:]]*/, ""); print $0 ","}'|
sed -e 's/<tr><td>//g' -e 's#;[0-9]*\.[0-9]*inHg</td></tr>##g' |
sed 's/\([0-9]\+\)mb/\1 mb/g')

# The following line assigns all values of the maximum velocity of the winds in each
hurricane report, from the input kml file to the variable maximumIntensityField. The values
retrieved are only measured in knots.
```

```
# The logic to this line of code is similar to the previous two fields, where we first use
grep to get the next line after every line in the input kml file that contains the string
'Maxium Intensity' (using the -A option), as well as every line that contains this string.
This is done, as like before the maximum intensity values for each hurricane report are
contained in every line after this string, and each occurrence of these pairs of lines are
piped into the second grep command, which filters out all instances of the string '--', to
remove this from the pattern.

# The pairs of lines (where one line consists of the string 'Maxium Intensity' and the other
is the line containing the maximum intensity values, measured in different units for the
specific hurricane report) are piped into the following awk command. The awk command is used in a
similar fashion as before in the field latitudeAndLongitudeField as well as the
minSeaLevelPressureField, where lines containing the string 'Maxium Intensity' are removed, and
any whitespaces in the remaining lines containing values for the maximum intensity values are
replaced with nothing (hence they're also removed). The logic for completing this action
was as before.

# Again these lines, now containing the maximum intensity values of various hurricane report,
but measured in different units (knots, kph, and mph) as well as having additional kml tags
that we need to remove, are piped into a sed command. This command like before, uses
the same logic to remove any redundant kml tags and extra information of the maximum
intensity of the hurricane report measured in different units. After this command is
completed, we're now just left with the value of the maximum intensity measured in knots, but
since we removed all whitespaces in all lines containing the maximum intensity, there will be
no space between the value and the unit.

# Because of this we pipe all these lines to another sed command like before, that adds a space
between the value of the maximum intensity measured in knots, and the unit (knots).

maximumIntensityField=$(grep -A 1 "Maxium Intensity" "$arg1" |
grep -v '^--$' |
awk 'NR%2==0 {gsub(/[[:space:]]*/, ""); print $0}' |
sed -e 's/<tr><td>//g' -e's#;[0-9]*mph;[0-9]*kph</td></tr></table>]]>##g' |
sed 's/\([0-9]\+\)knots/\1 knots/g')



# The paste command is used to concatenate the lines from timestampField,
latitudeAndLongitudeField, minSeaLevelPressureField, and maximumIntensityField side be side
without any delimiter. The '<()' syntax is used to treat the output of the echo commands as
if they were files, and to ensure the fields are combined correctly. This produces the correct
CSV format with seperate columns for each field, and this is appended to the output csv file.


paste -d '\0' <(echo "$timestampField")
<(echo "$latitudeAndLongitudeField")
<(echo "$minSeaLevelPressureField")
<(echo "$maximumIntensityField") >>"$arg2"
```

Figure 1: Bash script of create_csv.sh that creates a csv file containing information about storm reports, from data that was extracted from kml files.

2.2 Overall textual description of code

To summarise the comments provided in the bash script code above:

- An input kml file containing the data to be extracted from, and the name of the output csv file which data is to be written to are passed in as two arguments.
- A header line containing the names of the fields of data that is to be extracted from the kml file are written to the output csv file.
- Grep, Sed, and Awk commands are used to appropriately extract and filter out the relevant data from the input kml file, and assign these values to the corresponding variables.
- A paste command is then used to concatenate the lines from each of these variables depicting the different fields, and ensure the fields are aligned correctly, before writing this data to the output csv file.

3 Storm Plots

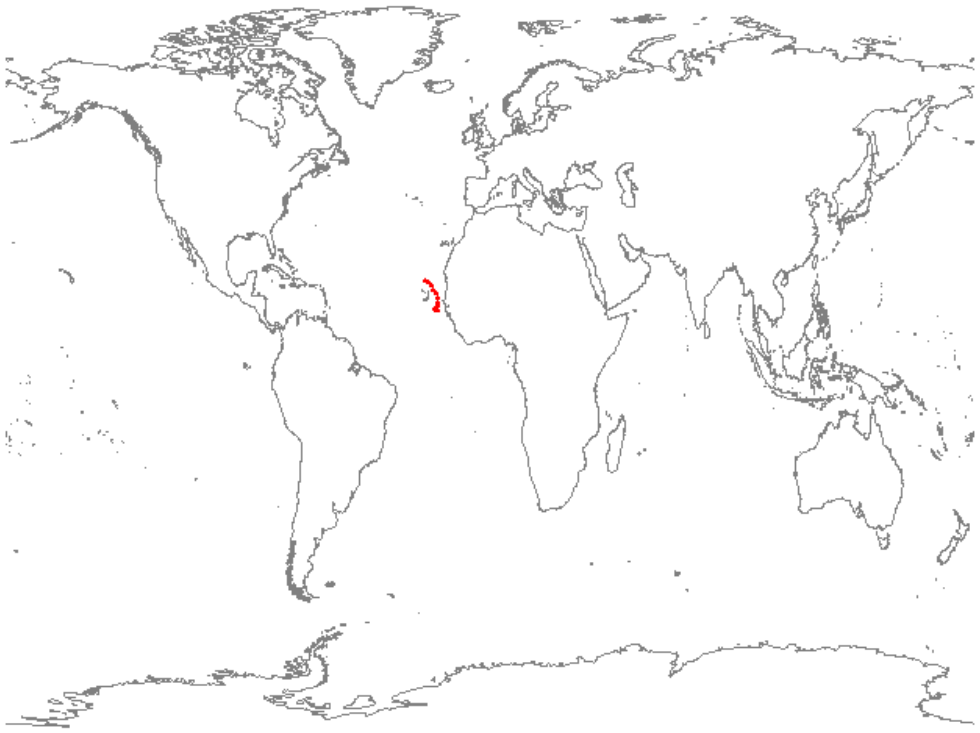

Figure 2: Storm plot of al102020.kml file

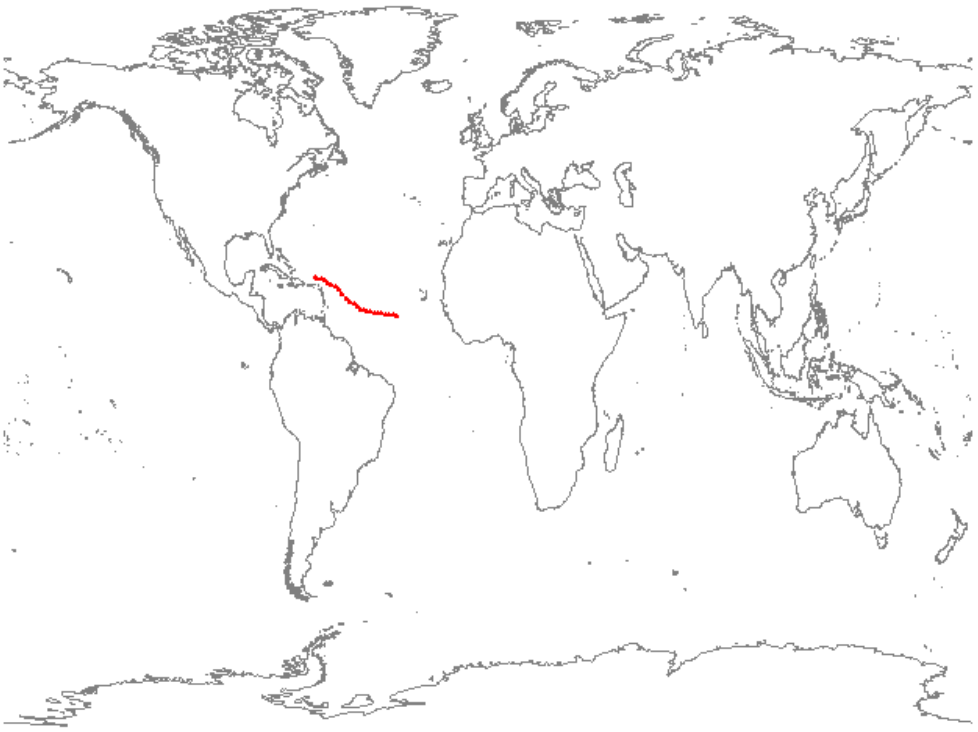

Figure 3: Storm plot of al112020.kml file

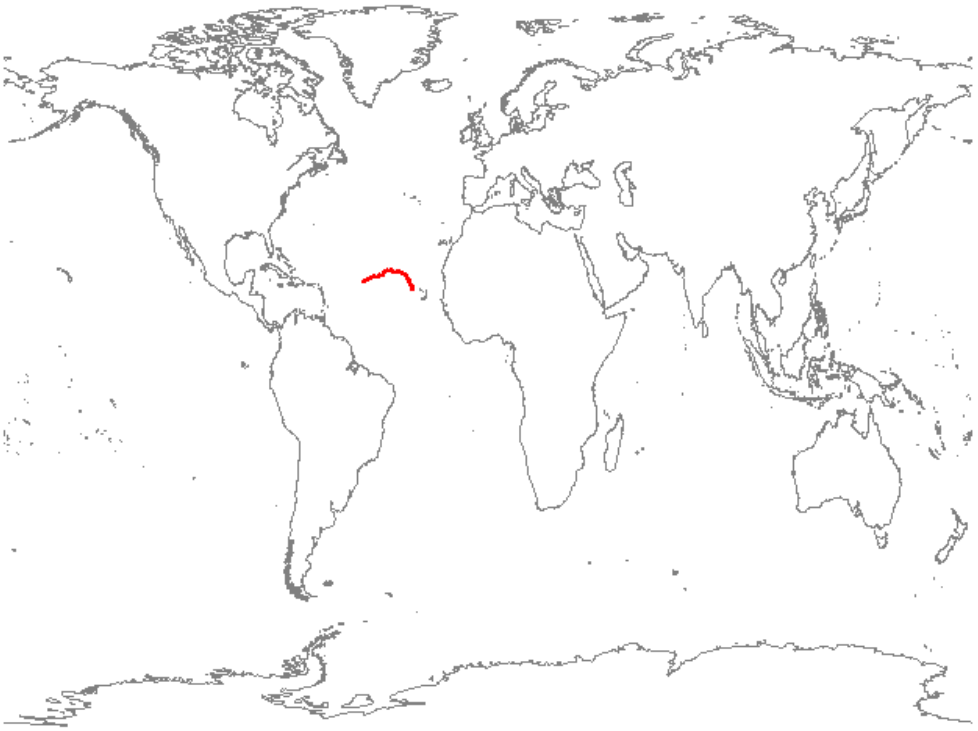

Figure 4: Storm plot of al212020.kml file

4 Git usage

4.1 Changes made to conflicted file

- First I removed all the code in between the '=======' and '`>>>>>>> python-addon`' for the first block code that involved several import statements, and defining a variable name user-key.
- Then I removed the tags '`<<<<<<< HEAD`', '=======' and '`>>>>>>> python-addon`' for the first block of code.
- I then repeated this process for the second block of code, in removing the code between the '=======' and '`>>>>>>> python-addon`' which included two methods.
- Then, like before I removed the tags '`<<<<<<< HEAD`', '=======' and '`>>>>>>> python-addon`' for the second block of code.
- Therefore, I was now just left with code that was edited in the main branch, and removed any code edited from the python-addon branch.

4.2 Commands used to resolve conflicts

The steps taken, and the commands used to resolve the conflicts are as follows:

- First I used the command `git merge python-addon` to attempt to merge the 'main' and 'python-addon' branches. This resulted in the file python-plot-script.py which contained the conflicting sections to be opened in vim, and I then followed the steps in the previous section to remove the conflicting parts of the code in the file.
- Then, I used the command `git add python-plot-script.py` to stage the resolved files.
- After this, the merge process of the two branches 'main' and 'python-addon' was completed through the use of the command `git merge --continue`.
- Then, the command `git commit` was used to write a message describing the changes made to resolving the conflicts.
- Finally, the command `git push` was used to push the changes to the remote repository.

4.3 Bash script of resolved file that contained the conflicts

```
import pandas as pd
import matplotlib.pyplot as plt
import os
import glob

import matplotlib.pyplot as plt
user_key= 8343

def plot_all_csv_pressure():
    path = os.getcwd()
    csv_files = glob.glob(os.path.join(path, '*.csv'))

    for f in csv_files:
        storm = pd.read_csv(f)
        storm['Pressure'].plot()
        plt.show()
```

Figure 5: Python code of python-plot-script.py