COMP1204: Data Management
Coursework One: Hurricane Monitoring

Abdullah Haitham Maghrabifetaih
3516005

March 7, 2024

1 Introduction

The main goal of our work is to apply data management skills to efficiently extract storms latitude,longitude, Min sea level pressure and Intensity and translating them into visual maps for a better storm tracking and analysis.Moreover, enhancing version control techniques by resolving the Git conflict which result in enhancing our collaboration and code management techniques.This approach makes us deal in a good way with complex data and group work on coding assessments.

2 Create CSV Script

Here is my script for extracting data from Kml files:

```bash
#!/bin/bash
# A condition check the number of arguments $# passed if it is not
# two then it print a message , The message will show the way of use and exit
if [ "$#" -ne 2 ]; then
    echo "Usage: $0 input_path.kml output_path.csv"
    exit 1
fi

# Assign the input file which is (kml) and output file as (csv)
inputFile="$1"
outputFile="$2"


# This to write as a header to the outputFile , as the structure for the data structure extracted.
echo "Timestamp,Latitude,Longitude,MinSeaLevelPressure,MaxIntensity" > "$outputFile"

#The main proccess of extrating the data is in this block:

# - I used the paste command to merge lines from multiple outputs , seperated by commas ,

# - extract the specific data from the input file using (grep) with Perl-compatible regular expressions (-oP).
    # - <dtg>  used to match the sequence in what i am intrested in
    # - '\k' is used to discard anything before it so to exclude <dtg>
    # - '[^<]+' is used to match one or more character and this # specifically [^<] means that i am only intrested in things without <
    # to achive the wanted data
    # - And the input file is the kml file that will be passed this was used for the rest of the data but the tag was changed with each wanted data the wanted information

# - sed is used to filter and transform text.
    # - ’s/$/ N/’
    # - s is used to substitute and match a specific patteren with a new text
    # - /$/ is used $ this is to match the unite with the end of the line
    # - N and it will find a place for the unit and add the N unit in the end of a text, this was used for all data with units so all except the timestamt.

# - >> "$outputFile"  navigate the formated output to the csv file.

paste -d ’,’ \
  <(grep -oP ’<dtg>\K[^<]+’ "$inputFile") \
  <(grep -oP ’<lat>\K[^<]+’ "$inputFile" | sed ’s/$/ N/’) \
  <(grep -oP ’<lon>\K[^<]+’ "$inputFile" | sed ’s/$/ W/’) \
  <(grep -oP ’<minSeaLevelPres>\K[^<]+’ "$inputFile" | sed ’s/$/ mb/’) \
  <(grep -oP ’<intensity>\K[^<]+’ "$inputFile" | sed ’s/$/ knots/’) \
  >> "$outputFile"

# - Once the proccess come to the end, print "Done" indicating that the process was successfull.
echo "Done"
```

Listing 1: KML to CSV Conversion Script breakline

The script automates the conversion of the data from KML to CSV, by concentrating on important data like Timestamps, latitude, longitude , minimum sea level pressure, maximum intensity. it begins with validating the passing arguments provided by the user, then extracting data from the KML file by using the ’grep’ command and ’sed’ for assigning the units of each data extracted.The use of the ’(grep -oP)’ allows for precise matching pattern matching, to make sure the accuracy of the data extracted.Moreover, the data extracted is formatted and written to a CSV file, making it ready for reading and analysing.

3 Storm Plots

The three figures represents the storm events that have been captured and convert it into visual data. the process of converting begin with this command `./create_csv.sh file.kml file.csv` . this command takes the arguments, extract the data from the kml files, and store the data in the csv file.Furthermore, `./create_map_plot.sh file.csv file.png` command was executed,using the csv file as and input to generate a png file for data visualization map.This processes was applied for the three different kml files.

3.1

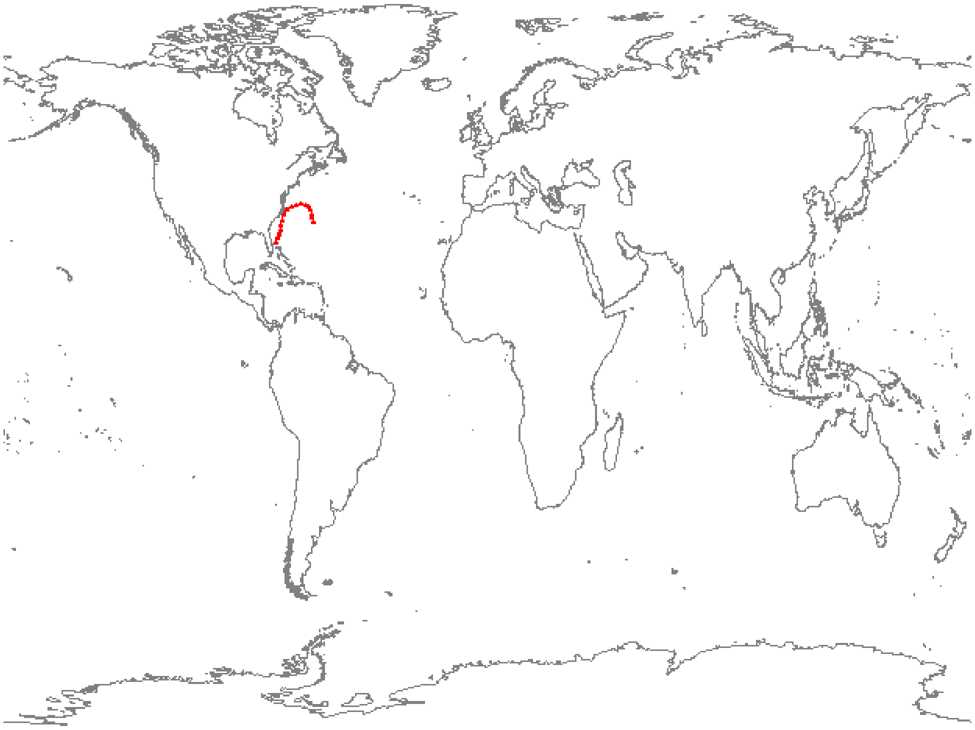

Figure 1: extracted from al012020.kml

3.2

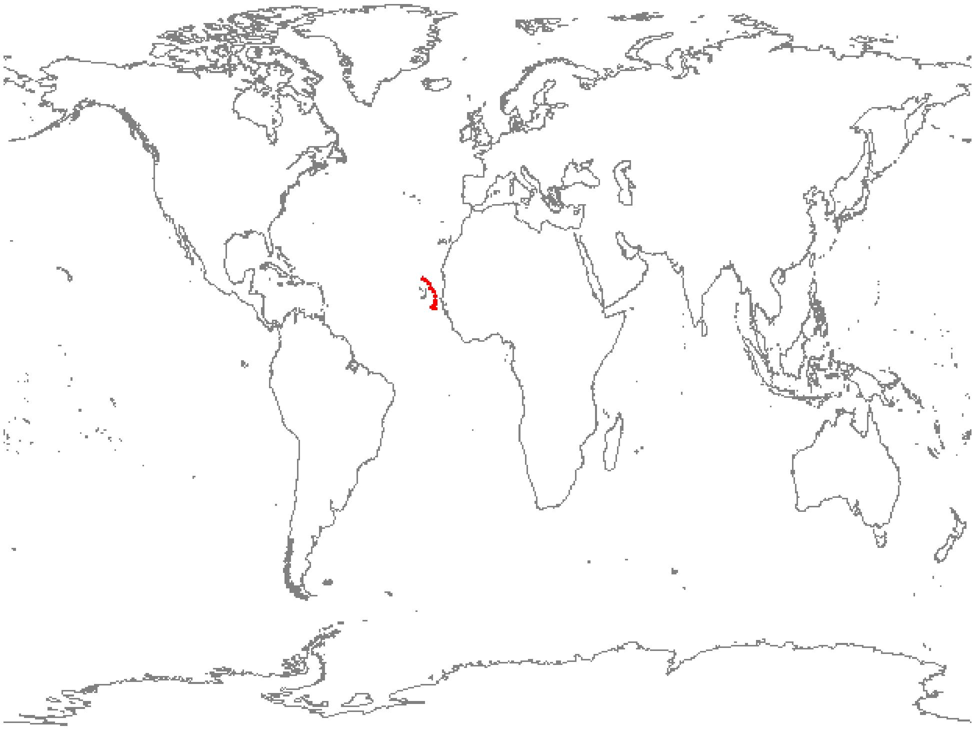

Figure 2: extracted from al102020.kml

3.3

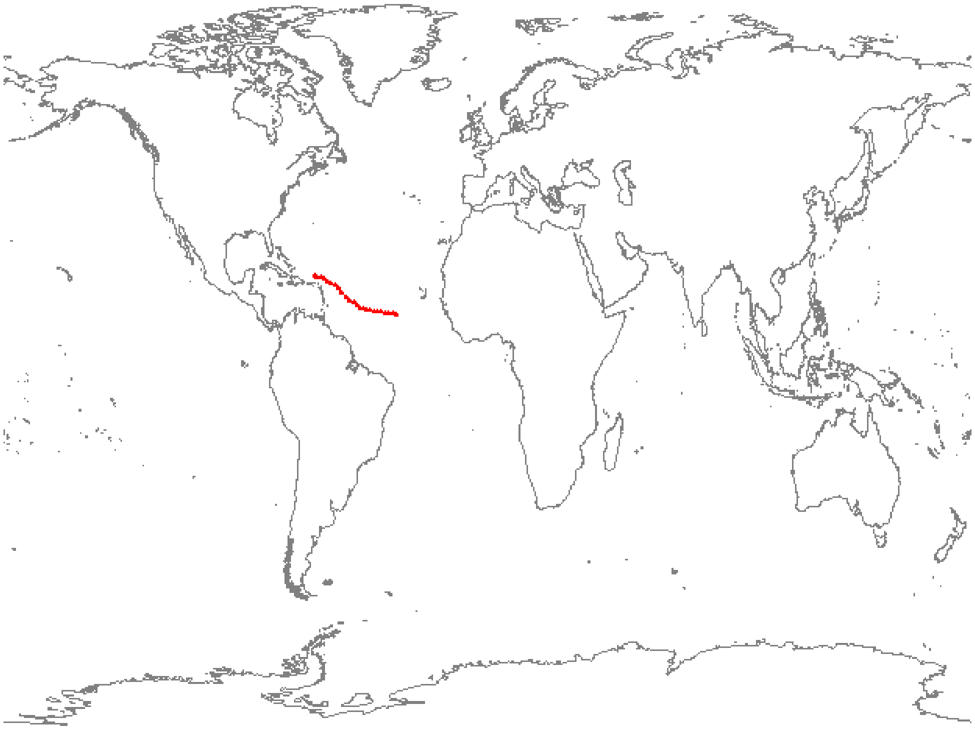

Figure 3: extracted from al112020.kml

4 Git Usage

4.1 Normal usage

The material used in the coursework material was staged, committed, and pushed to the GitLab repository using the following Git commands:

```
git add material/
git commit -m "initial commit"
git push
```

Git was also used to track updates made to the script file `create_csv.sh` The script was pushed to the remote repository, committed with a suitable message, and put to the staging area as follows. the code also contains a detailed comments to explain the functionality of the script.by the following commands.

```
git add create_csv.sh
git commit -m "This is my Script"
git push
```

To keep a complete version history of the project assets, all data visualization images were also committed and kept in the repository in addition .upon completion of the Latex report the same processes was made:

4.2 Resolving The Conflit

Initially,executed this command conflict-script.sh once to create the branch python-addon as instructions says by the following commands:

```
chmod +x conflict-script.sh
./conflict-script.sh
```

Moreover, moved to the python-addon branch to resolve the conflict:

```
git branch
git swith
```

Afterwards, there was a conflict in `python-plot-script.py` i moved to the version control to resolve the conflicts the file encounters,the conflict was because of some errors including syntax errors.within the script, wrong syntax was used in a loop, so i erased it.Furthermore, and also a randomly typing errors such as `"<<<<<<< Head"`, `">>>>>>> python addon "` and ”======”,these were also erased.Additionally,there was redundant import statements,which i also deleted.Nevertheless,there was a two different key variables but i did not change them because i thought it should be used later on.Finally,I pushed the resolving merge branches into the repository.Here is how the vim of `python-plot-script.py` looks like before and after resolving the conflict:

```
<<<<<<< HEAD
import matplotlib.pyplot as plt
import os
import glob
import math
user_key = 15256
=======
import os
import glob
import matplotlib.pyplot as plt
user_key= 10062
>>>>>>> python-addon

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

<<<<<<< HEAD
    for f in csv_files:
        storm = pd.read_csv(f)
        storm[’Pressure’].plot()
        plt.show()
=======
    fr f in csv_files:
        storm = pd.read_csv(f)
        storm[’Pressure’].plot()
        plt.show()

def plot_all_csv_intensity():
    path = os.getcwd()
    csv_files = glob.glob(os.path.join(path, ’*.csv’))

    for f in csv_files:
        storm = pd.read_csv(f)
        storm[’Intensity’].plot()
        plt.show()
>>>>>>> python-addon
```

Listing 2: The python script before resolving the conflict

```
import pandas as pd
import math
user_key = 15256
import os
import glob
import matplotlib.pyplot as plt
user_key= 10062


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()


def plot_all_csv_intensity():
    path = os.getcwd()
    csv_files = glob.glob(os.path.join(path, ’*.csv’))

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

Listing 3: The python Script after resolving the conflict

git switch, used this to change to the other branch after resolving the conflict.
git merge, merge was successful after resolving the conflict.