# COMP1204: Data Management
# Coursework One: Hurricane Monitoring

Constantinos Psaras
cp2n23

February 21, 2024

## Contents

## 1 Introduction

In response to the rising challenges posed by tropical cyclones, the National Oceanographic and Atmospheric Administration (NOAA) has dedicated efforts to bolster its capabilities in storm tracking and analysis. As part of this initiative, our data science team at the NOAA Centre has embarked on a mission to extract crucial storm data from tropical cyclone reports. Focused on harnessing the power of advanced data analysis techniques, our goal is to transform raw storm information into insightful visualizations and maps. Through this technical report, we aim to showcase the methodologies employed, the data extraction processes implemented, and the resulting visual representations, contributing to a more comprehensive understanding of tropical cyclone behavior and aiding in proactive disaster management efforts. My role was to write a bash script which takes the captured readings from kml files and extracts the specified data into a csv file. Another provided bash script taked my csv files and generates map plots based on the readings.

## 2 Create CSV Script

```
#!/bin/bash
#Script that extracts information from a KML file and generates a CSV file

#Checks if 2 arguements were passed, if not exits
if [ "$#" -ne 2 ]; then
    echo "Usage: $0 <input.kml> <output.csv>"
    exit 1
fi

#Assigns the 2 arguements to input_file and output_file respectively
input_file="$1"
output_file="$2"

#Adds field headers to the output CSV file
echo "Timestamp,Latitude,Longitude,MinSeaLevelPressure,MaxIntensity" >
   "$output_file"

#Awk extracts all measurements enclosed in specified tags and adds units
   of measurement accordingly
#At the end, if all required fields have values, they get appended to the
   specified output csv file
awk -F '[<>]' '
    /<dtg>/ { timestamp = $3 }
    /<lat>/ { latitude = $3 " N" }
    /<lon>/ { longitude = $3 " W" }
    /<minSeaLevelPres>/ {min_pressure = $3 " mb" }
    /<intensity>/ { max_intensity = $3 " knots" }
    /<\/Placemark>/ {

    if (timestamp && latitude && longitude && min_pressure &&
       max_intensity) {
        print timestamp "," latitude "," longitude "," min_pressure ","
           max_intensity
    }
        timestamp = ""
        latitude = ""
        longitude = ""
        min_pressure = ""
        max_intensity = ""
    }


' "$input_file" >> "$output_file"

#Prints confirmation uppon succesful creation of the csv file
echo "CSV file generated: $output_file"
```

Explanation:

- Beginning with an if statement, the script checks if the number of command-line arguments is exactly 2. If not, it prints a usage message and exits.
- The two arguments are assigned to `input_file` and `output_file` respectively.

- Field/column names are printed to the output file.
- The `awk` command then processes the `input_file`. It sets the field separator as any character between `<`, and `>` using `-F '[<>]'`.
- For each line, it checks for specific tags (`<dtg>, <lat>, <lon>, <minSeaLevelPres>, <intensity>`), extracts their values and adds an appropriate unit of measurement.
- When it encounters `</Placemark>`, it checks if all required fields have values. If yes, it appends aquired values to the specified output csv file using using `>>`.
- Upon succesful processing and creation of a csv file, a confirmation message is printed to the console.

## 3 Storm Plots

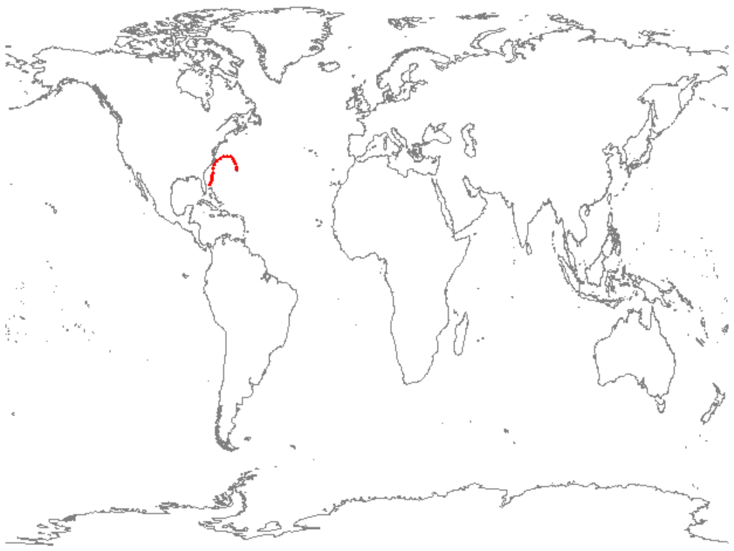

Figure 1: Plot generated from al012020.kml file

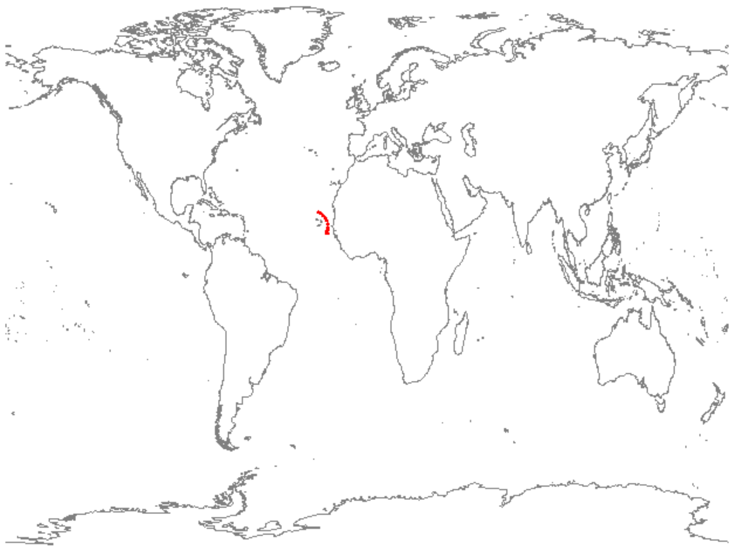

Figure 2: Plot generated from al102020.kml file

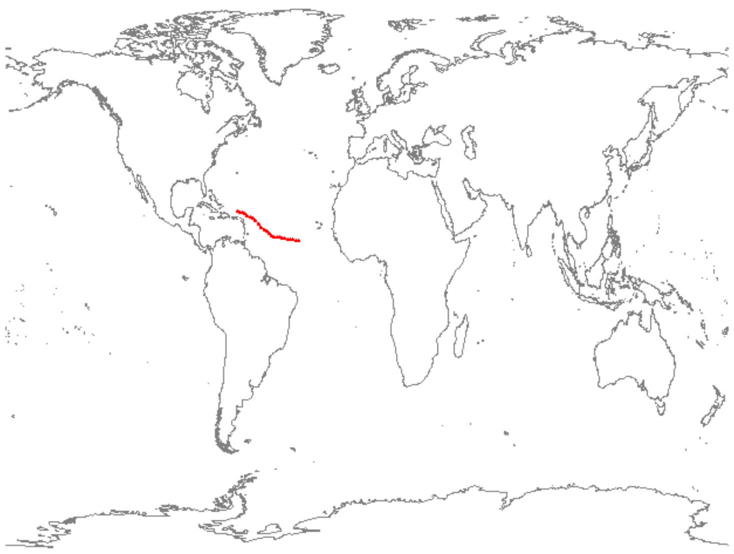

Figure 3: Plot generated from al112020.kml file

## 4 Git Conflict Handling

```
import pandas as pd
import math
import os
import glob
import matplotlib.pyplot as plt
user_key = 22653 #Kept the original user key.

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()
```

After running conflict-script.sh, a new branch called python-addon was added to my project with a file called python-plot-script.py. The file was also created in my main branch. I realized their contents differed after I got an error when trying to merge the 2 branches. I then switched to my main branch using the command 'git checkout main', then edited the file python-plot-script.py through the nano text editor. Then I executed the command 'git add python-plot-script.py' to add the file to the staging area and then executed 'git commit' to capture the staged changes. I finally pushed the changes back to my project without any errors. Commands used to resolve conflict:

- ./conflict-script.sh
- git checkout main
- git merge python-addon
- Conflict appear here, after resolving:
- git add python-plot-script.py
- git commit -m "Conflict resolved"
- git push origin main