UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Johann Pienaar - u24050505

MIT 805

Big Data: Assignment 2

29 October 2022

SCHOOL OF INFORMATION TECHNOLOGY

MASTER OF INFORMATION TECHNOLOGY

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

INDIVIDUAL ASSIGNMENT COVER PAGE

Name of Student	Johann Pienaar
Student Number	u24050505
Name of Module	Big data
Module Code	MIT 805
Name of Lecturer	Stacey Baror
Date of Submission	29 October 2022
Contact telephone number	(+27) 072 392 1960
E-mail address	jpienaar85@gmail.com / u24050505@tuks.co.za
Declaration:	*I declare that this assignment, submitted by me, is my own work and that I have referenced all the sources that I have used.*
Signature of Student	Johann Pienaar
Date received	
Signature of Administrator	
Mark	
Date	
Signature of Lecturer	

1 MapReduce

All of the code for the sections discussed below are available on my GitHub repository:

- https://github.com/jpienaar-tuks/MIT805

1.1 Algorithm considerations

For the purposes of this project, I decided to aggregate data by country and year. Furthermore, I decided to aggregate temperature data by season (winter/summer) since increases in temperature during these seasons should give a good indication of climate change since the temperatures for autumn/spring are expected to be in between and therefore less interesting. Finally, I decided to aggregate precipitation data by month since this will give insight on how precipitation may be changing in a country throughout the year (wetter/drier/or maybe the rainy season is shifting earlier/later). This could have an impact especially on the agriculture industry.

Since I didn't include country data (or hemisphere data for season aggregation) in my earlier preprocessing of the raw data into csv's, I decided to create a dictionary that the mapper processes can load at run time to infer this information. I decided to use the equinox months, Jun/Dec as the start of winter/summer months respectively in the southern hemisphere and inverted for the northern hemisphere with a season being defined as 3 months.

The mapper code was written in python (see appendix) and used Hadoop Streaming [1] to execute the process. For the reducer I just needed to calculate averages, for which I could leverage one of the built-in Hadoop functions (DoubleValueSum) by using the 'aggregate' argument in the argument list and being careful with the mapper output keys. I also needed to take care to include variable counters so that I can calculate averages in post-processing. Final Hadoop execution was therefore:

```
mapred streaming -input input/csv -output output/run1 \
      -mapper mapper.py \
      -reducer aggregate \
      -file station-dict.pickle \
      -file mapper.py
```

Using this approach I was able to reduce the original 7.68 Gb of data of uncompressed csv's to about 14 Mb of data that was suitable for post-processing, performing regressions and creating visualisations. See table 1 for an example of MapReduce output for SA data for 2021. Variables with a ‘_C’ suffix indicates a count and was used to calculate averages.

```
SF.2021.01.PRCP          13289.5
SF.2021.01.PRCP_C         4638.0
SF.2021.02.PRCP           8522.1
SF.2021.02.PRCP_C         3552.0
SF.2021.03.PRCP           6068.7
SF.2021.03.PRCP_C         4656.0
SF.2021.04.PRCP           3666.3
SF.2021.04.PRCP_C         4662.0
SF.2021.05.PRCP           4279.9
SF.2021.05.PRCP_C         4806.0
SF.2021.06.PRCP           3958.0
SF.2021.06.PRCP_C         4547.0
SF.2021.07.PRCP           3081.8
SF.2021.07.PRCP_C         4872.0
SF.2021.08.PRCP           4763.1
SF.2021.08.PRCP_C         4915.0
SF.2021.09.PRCP           3920.6
SF.2021.09.PRCP_C         4639.0
SF.2021.10.PRCP           7166.6
SF.2021.10.PRCP_C         4576.0
SF.2021.11.PRCP           7090.4
SF.2021.11.PRCP_C         4193.0
SF.2021.12.PRCP          14270.3
SF.2021.12.PRCP_C         4497.0
SF.2021.SUMMER.MAX_T    373577.0
SF.2021.SUMMER.MAX_T_C   12910.0
SF.2021.SUMMER.MEAN_T   275772.6
SF.2021.SUMMER.MEAN_T_C  12931.0
SF.2021.SUMMER.MIN_T    210480.1
SF.2021.SUMMER.MIN_T_C   12927.0
SF.2021.WINTER.MAX_T    310401.7
SF.2021.WINTER.MAX_T_C   14406.0
SF.2021.WINTER.MEAN_T   182469.0
SF.2021.WINTER.MEAN_T_C  14418.0
SF.2021.WINTER.MIN_T     81837.9
SF.2021.WINTER.MIN_T_C   14413.0
```

Table 1 Sample MapReduce output for SA, 2021

1.2 Postprocessing

During post-processing averages were calculated and regressions performed using the Scipy library per country per variable (min temperature, mean temperature, max temperature, and precipitation) and per season/month for temperature and precipitation data respectively. Of the output from the regressions, I'm largely interested in the slope (rate of change per year) and the p-statistic to indicate whether the relationship discovered is significant. One limitation that may be worth noting is that the number of stations grew over the period of the dataset – if these stations were erected in locations that weren't representative of the country's climate (disproportionately wet/dry/hot/cold) then it could skew the data.

Finally, for my visualisations, I used the plotly [2] library in python for which I needed world GeoJSON data [3] as well as a means of converting FIPS country codes to ISO_3166_3 [4].

1.3 Future considerations

Since aggregations were done on an annual basis, including future data in the current pipeline wouldn't be a challenge and wouldn't require recalculating past data. Including future weather stations also wouldn't present a challenge beyond requiring that the appropriate geo-spatial information is updated.

There is however significant opportunity to improve the performance of the mapper code by including country and hemisphere data in the pre-processing of csv's. I suspect that the loading of this data into the mapper process slowed the process (perhaps imperceptibly individually, but it quickly accumulates over a large dataset). Finally, the default configuration of Hadoop seems to have spawned 1 mapper process per file (~27000 in all), despite each mapper being quite capable of processing multiple files. There is therefore opportunity to explore tuning the splitting settings.

2 Visualisation

Both the code as well as the HTML source files for the visuals discussed below as well as some additional visuals are available on my GitHub repository:

- https://github.com/jpienaar-tuks/MIT805

2.1 World summer temperatures

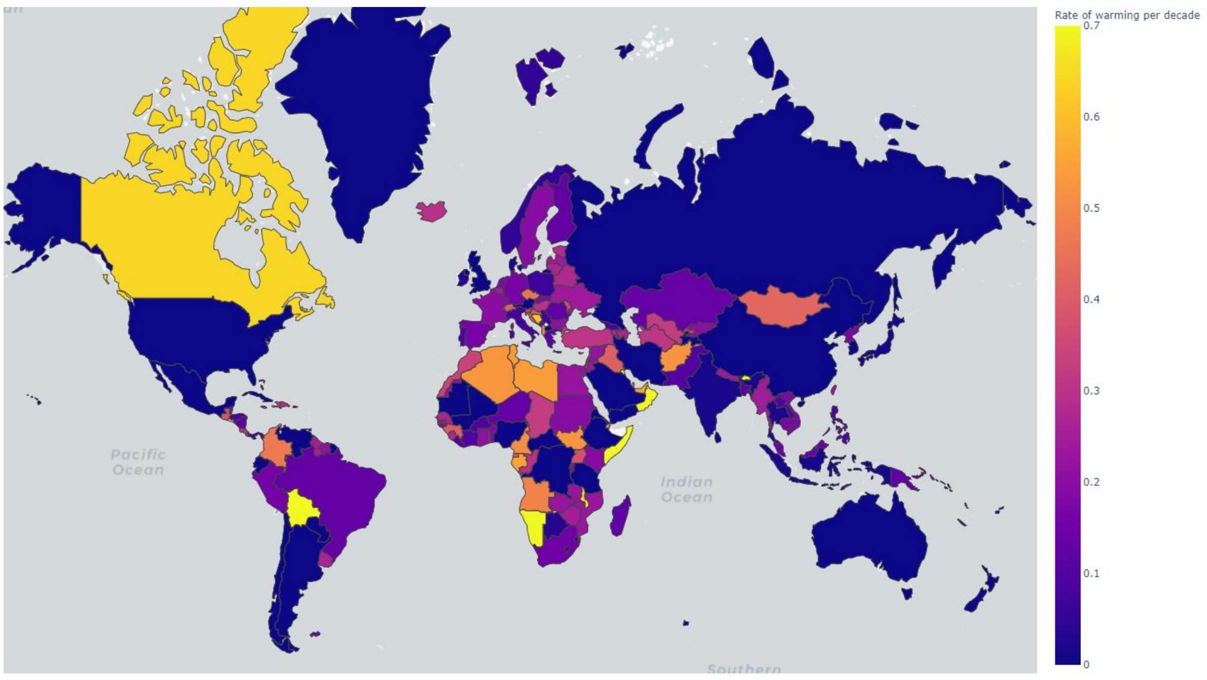

Table 2 Rate of warming in summer mean temperatures

Although at first glance it may appear as though much of the world is not warming at any significant rate, it should be noted that the p-statistic for most of those areas indicate that the result is not significant (i.e. Russia has a p-statistic of 0.699 and China 0.1251). Also note that the map projection used by the plotly library inflates the size of objects near the poles such as Greenland and Russia.

2.2 World precipitation

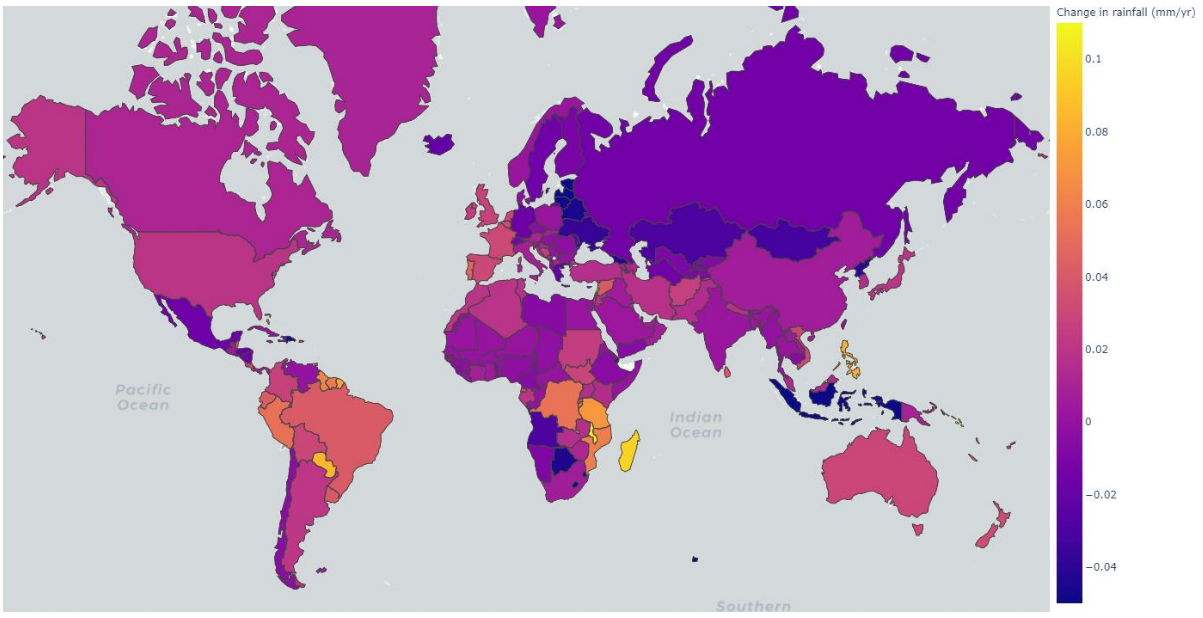

Table 3 Rate of change in rainfall (January)

In the above image we can see the impact of climate change in the significant increase in rainfall that countries in the south west Indian ocean receive in January due to changes in cyclone frequency/intensity.

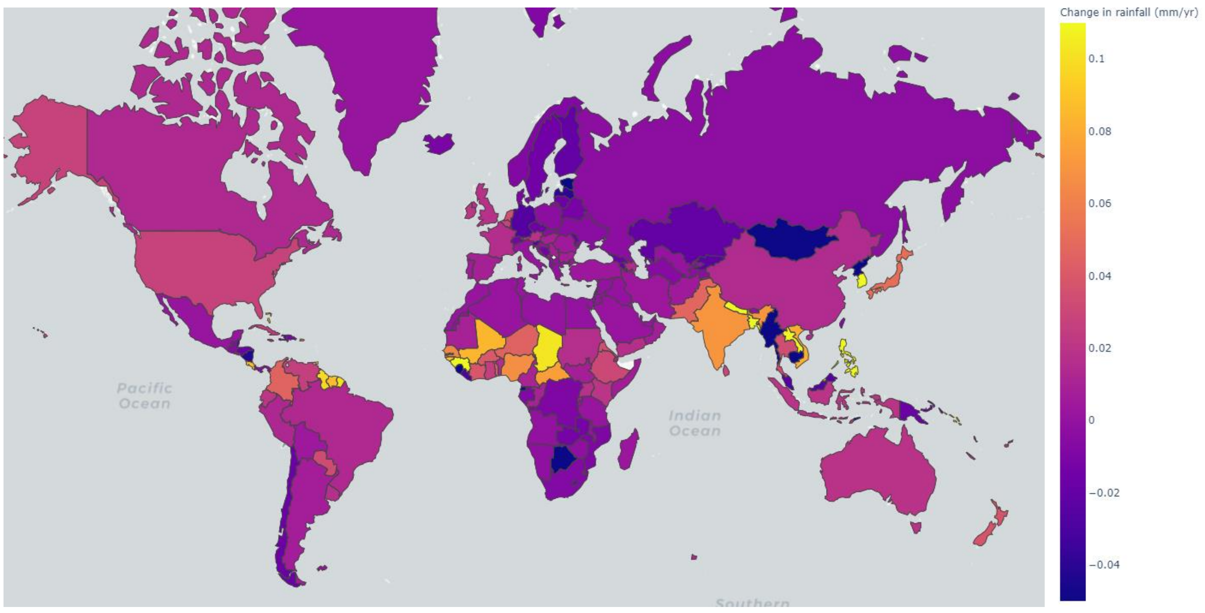

Table 4 Rate of change in rainfall (July)

In the above image we can see the increases in rainfall received by countries such as India, Pakistan and Nepal as a result of global warming. Also visible in both images is increases in rainfall received by the Philippines (becoming a much wetter country year-round) as well as decreases in rainfall in Mongolia (becoming much drier year-round).

2.3 SA temperatures

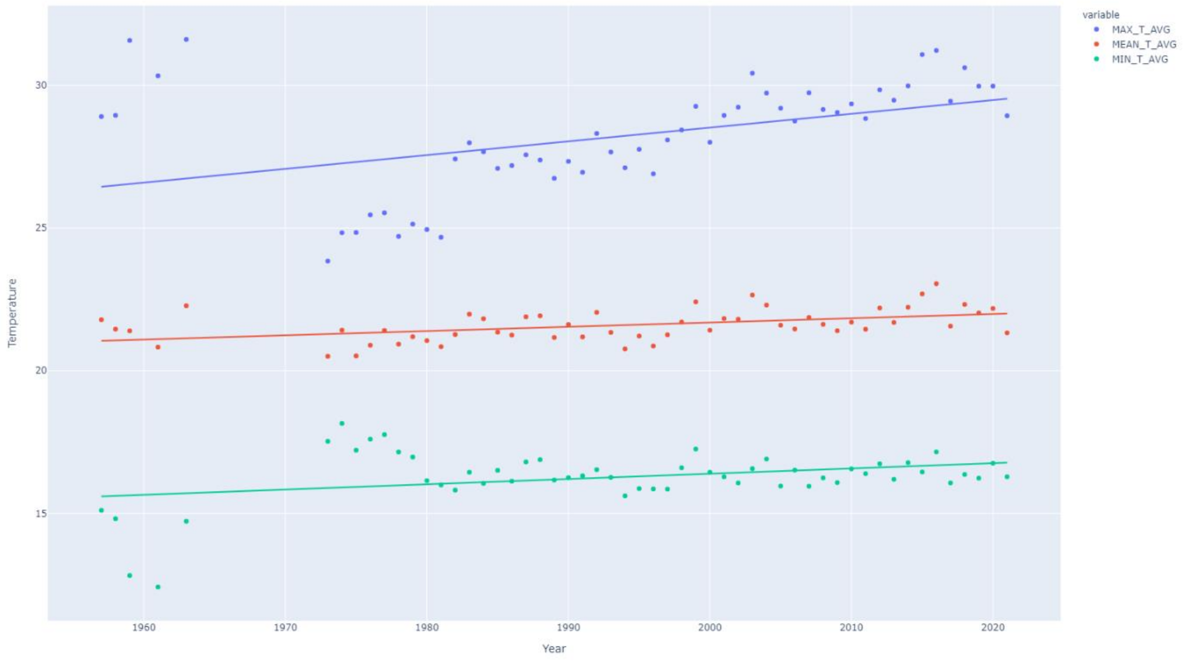

Table 5 SA Summer temperatures

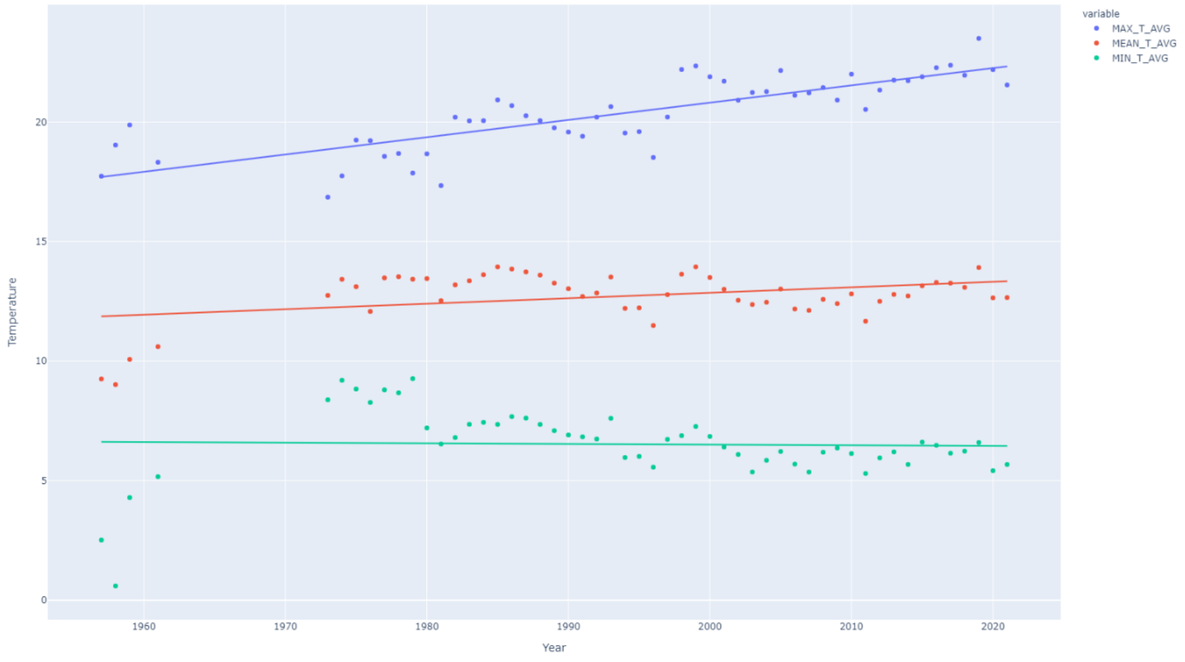

Table 6 SA winter temperatures

Country	Season	Variable	Rate of warming (°C/decade)	p
SF	WINTER	MIN_T_AVG	-0.03	0.82
SF	WINTER	MEAN_T_AVG	0.23	0.01
SF	WINTER	MAX_T_AVG	0.72	0.00
SF	SUMMER	MIN_T_AVG	0.18	0.01
SF	SUMMER	MEAN_T_AVG	0.15	0.00
SF	SUMMER	MAX_T_AVG	0.48	0.00

Table 7 SA temperatures regression results

From the two charts as well as the table above, we can see that the South African summer is warming at 0.15 °C/decade, while the winter is warming at 0.23 °C/decade, indicating that while our summers are becoming warmer, our winters are warming at a faster rate which may have implications for diseases such as malaria if mosquitos are able to survive the winter in larger parts of the country. We can also see that the midday maximum temperatures are warning faster than the morning minimums, indicating that the diurnal temperature range is becoming larger. Finally, all the regression results achieved a significant p-statistic, except for the winter morning minimum temperature.

2.4 SA precipitation

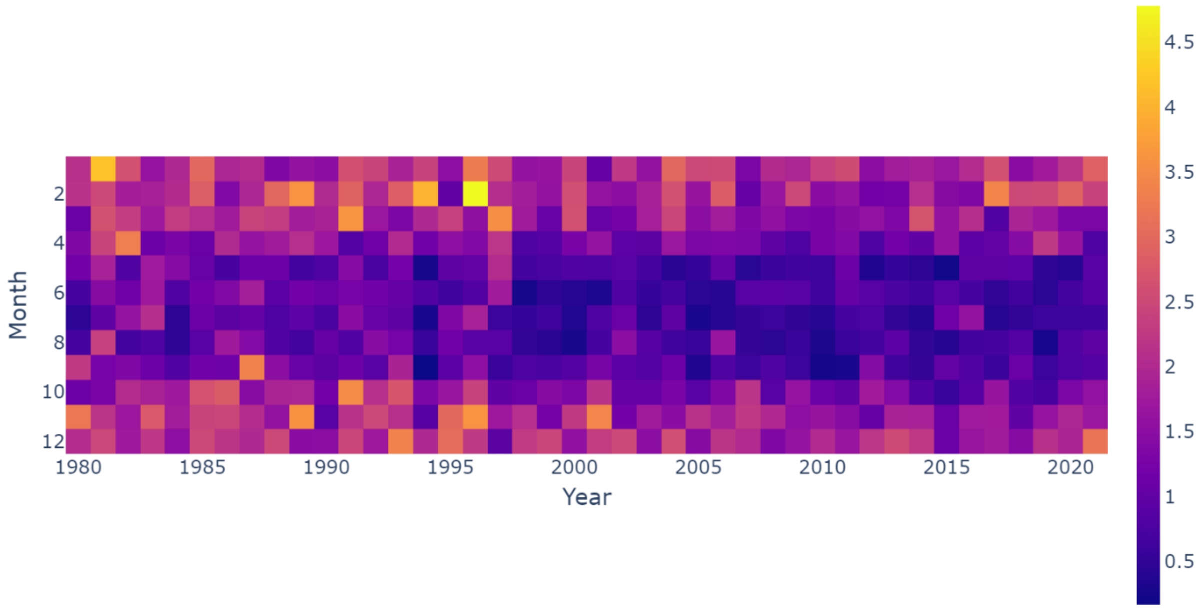

Table 8 SA precipitation since 1980

From the image above it seems that South Africa has been becoming a drier country since the mid-late 90’s especially around the period April-August. While the regressions seem to confirm this result, the p-statistic fails to indicate a significant relationship except for February, May and September.

Country	Month	Rate of change (mm/decade)	p
SF	1	0.07	0.37
SF	**2**	**0.15**	**0.05**
SF	3	0.01	0.84
SF	4	-0.07	0.29
SF	**5**	**-0.17**	**0.00**
SF	6	-0.07	0.08
SF	7	-0.07	0.13
SF	8	-0.10	0.08
SF	**9**	**-0.14**	**0.01**
SF	10	-0.10	0.11
SF	11	-0.10	0.13
SF	12	0.08	0.16

Table 9 Regression results for precipitation in SA

3 References

[1] Apache, “Apache Hadoop MapReduce Streaming,” [Online]. Available: https://hadoop.apache.org/docs/r3.3.4/hadoop-streaming/HadoopStreaming.html. [Accessed 10 10 2022].

[2] Plotly, “Plotly Python Graphing Library,” [Online]. Available: https://plotly.com/python/. [Accessed 10 10 2022].

[3] J. Sundström, “GitHub,” [Online]. Available: https://raw.githubusercontent.com/johan/world.geo.json/master/countries.geo.json. [Accessed 10 10 2022].

[4] D. Hernan, “GitHub,” [Online]. Available: https://github.com/dieghernan/Country-Codes-and-International-Organizations/blob/master/outputs/Countrycodes.csv.

4 Appendix

4.1 Video report

https://drive.google.com/file/d/1IlQ6U9F55awkpoqvC_FXNSMV2vNXZjgJ/view?usp=share_li
nk

4.2 Mapper code

```
#!/usr/bin/python3
import sys
import pickle
from collections import namedtuple

def mapper():
    for line in sys.stdin:
        row=Line._make(line.split(', '))
        if row.STN=='STN':
            continue # Continue to next row if current row is a file header
        try:
            country, hemisphere = station_dict[f'{row.STN}-{row.WBAN}'].split(', ')
        except KeyError:
            continue # Continue to next row if current station didn't have geo-
spatial info

        # For temperatures I'm mostly interested in the winter/summer changes,
        # since autumn, spring should fall inbetween by default. I'll also
        # consider the equinox months to mark the start of the season. Therefore:
        # if the station is in the northern hemisphere, then I'll consider the
        # months 6,7,8 to be summer and 1,2,12 to be winter. This is reversed
        # in the southern hemisphere. For changes in precipitation I'm interested
        # in year round changes. Finally, I anticipate using the built-in
aggregation
        # functions provided by the hadoop streaming API, but will need to preserve
        # some data count information, so that averages can be calculated
afterwards

        #Let's start with precipitation:
        try:
            float(row.PRCP)

print(f'DoubleValueSum:{country}.{row.YEAR}.{row.MONTH}.PRCP\t{row.PRCP.strip()}')
            print(f'DoubleValueSum:{country}.{row.YEAR}.{row.MONTH}.PRCP_C\t1')
        except ValueError:
            pass

        #Let's move on to temperatures:
        if int(row.MONTH) in [1,2,6,7,8,12]:
            season=season_dict[f'{hemisphere}-{int(row.MONTH)}']
            for variable, variable_text in zip([row.MAX_TEMP, row.MEAN_TEMP,
row.MIN_TEMP],
                                               ['MAX_T','MEAN_T','MIN_T']):
                writeline(variable, variable_text, country, row.YEAR, season)


def writeline(variable, variable_text, country, year, season):
    try:
        float(variable)

print(f'DoubleValueSum:{country}.{year}.{season}.{variable_text}\t{variable}')
        print(f'DoubleValueSum:{country}.{year}.{season}.{variable_text}_C\t1')
    except ValueError:
        pass


if __name__ == "__main__":
    with open('station-dict.pickle','rb') as f:
        station_dict=pickle.load(f)
    Line = namedtuple('Line','STN, WBAN, YEAR, MONTH, DAY, MEAN_TEMP, MAX_TEMP,
MIN_TEMP, PRCP')
    season_dict={}
    for hemisphere in ['N','S']:
        for month in [1,2,6,7,8,12]:
            if hemisphere=='N':
```

```
            if month in [6,7,8]:
                season_dict[f'{hemisphere}-{month}']='SUMMER'
            else:
                season_dict[f'{hemisphere}-{month}']='WINTER'
        else:
            if month in [6,7,8]:
                season_dict[f'{hemisphere}-{month}']='WINTER'
            else:
                season_dict[f'{hemisphere}-{month}']='SUMMER'
mapper()
```

4.3 Post-processing and regressions

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 12 10:14:57 2022

@author: Johann
"""
import pandas as pd
import os
from scipy.stats import linregress

precip_data=[]
temperature_data=[]

with open(os.path.join('..','hadoop output','part-00000'),'rt') as f:
    for line in f.readlines():
        prefix, value = line.split('\t')
        fields = prefix.split('.')
        if 'PRCP' in line:
            precip_data.append(fields+[value])
        else:
            temperature_data.append(fields+[value])
df_temp = pd.DataFrame(temperature_data,
columns=['Country','Year','Season','Variable','Value'])
df_temp['Year']=df_temp['Year'].astype('int32')
df_temp['Value']=df_temp['Value'].astype('float')

df_temp_pivot = df_temp.pivot(index=['Country','Year','Season'],
columns='Variable',values='Value')
df_temp_pivot['MAX_T_AVG'] = df_temp_pivot['MAX_T']/df_temp_pivot['MAX_T_C']
df_temp_pivot['MEAN_T_AVG'] = df_temp_pivot['MEAN_T']/df_temp_pivot['MEAN_T_C']
df_temp_pivot['MIN_T_AVG'] = df_temp_pivot['MIN_T']/df_temp_pivot['MIN_T_C']
df_temp_pivot =
df_temp_pivot.drop(['MAX_T','MAX_T_C','MEAN_T','MEAN_T_C','MIN_T','MIN_T_C'],axis=1
).unstack()
df_temp_pivot.stack().to_csv('temp.csv')

df_precip = pd.DataFrame(precip_data,
columns=['Country','Year','Month','Variable','Value'])
df_precip['Year']=df_precip['Year'].astype('int32')
df_precip['Month']=df_precip['Month'].astype('int32')
df_precip['Value']=df_precip['Value'].astype('float')

df_precip_pivot =
df_precip.pivot(index=['Country','Year','Month'],columns='Variable',values='Value')
df_precip_pivot['PRCP_AVG'] = df_precip_pivot['PRCP']/df_precip_pivot['PRCP_C']
df_precip_pivot = df_precip_pivot.drop(['PRCP','PRCP_C'], axis=1).unstack()
df_precip_pivot.stack().to_csv('precip.csv')

# Precip regressions
precip_regressions=[]
for country in df_precip_pivot.index.levels[0]:
    for month in range(1,13):
        x = df_precip_pivot.loc[country].index.values
        y = df_precip_pivot.loc[country,('PRCP_AVG',month)].values
        m, c, r, p, *rest = linregress(pd.DataFrame(zip(x,y)).dropna())
        precip_regressions.append([country, month, m, r**2, p])
pd.DataFrame(precip_regressions,columns=['Country','Month','Slope','R2','p']).to_cs
v('Precipitation regressions.csv',index=False)

# Temperature regressions
temp_regressions=[]
for country in df_temp_pivot.index.levels[0]:
    for season in ['WINTER','SUMMER']:
        for variable in ['MIN_T_AVG','MEAN_T_AVG','MAX_T_AVG']:
            x = df_temp_pivot.loc[country].index.values
            y = df_temp_pivot.loc[country,(variable,season)].values
            m, c, r, p, *rest = linregress(pd.DataFrame(zip(x,y)).dropna())
            temp_regressions.append([country, season, variable, m, r**2, p])
pd.DataFrame(temp_regressions,columns=['Country','Season','Variable','Slope','R2','
p']).to_csv('Temperature regressions.csv',index=False)
```

4.4 Visualisations

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 13 10:44:05 2022

@author: Johann
"""

import json
import pandas as pd
import plotly.express as px
import os

with open('countries.geo.json','rt') as f:
    geojson = json.load(f)

df_temp = pd.read_csv('Temperature regressions.csv')
country_code_lookups = pd.read_csv('Countrycodes.csv',
usecols=['FIPS_GEC','ISO_3166_3'])
country_code_lookups.columns=['ISO3','FIPS']

country_lookups_dict = {row.FIPS: row.ISO3 for i, row in
country_code_lookups.iterrows()}


for season in ['WINTER','SUMMER']:
    for variable in ['MIN_T_AVG','MEAN_T_AVG','MAX_T_AVG']:
        temp_visual=[]
        for i, row in df_temp.loc[(df_temp.Season==season) &
(df_temp.Variable==variable)].iterrows():
            try:

iso3=country_code_lookups.loc[country_code_lookups.FIPS==row.Country,
'ISO3'].values[0]
            except IndexError:
                continue
            temp_visual.append([iso3, row.Slope*10, row.p])
        df_visual = pd.DataFrame(temp_visual, columns=['ISO3','slope','p'])

        fig = px.choropleth_mapbox(df_visual, geojson=geojson,
                                   locations='ISO3', color='slope',
                                   range_color=[-0,0.7],
                                   mapbox_style="carto-positron",
                                   hover_data={'slope':':.2f',
                                               'p':':.4f'},
                                   labels={'slope': 'Rate of warming per decade',
                                           'p': 'p statistic'},
                                   zoom=2)

        with open(os.path.join('..','visuals',f'{season}-{variable}.html'),'wt') as
f:
            f.write(fig.to_html(include_plotlyjs='cdn'))


df_precip = pd.read_csv('Precipitation regressions.csv')

precip_visual=[]
month_dict={1:'Jan',
            2:'Feb',
            3:'Mar',
            4:'Apr',
            5:'May',
            6:'Jun',
            7:'Jul',
            8:'Aug',
            9:'Sep',
            10:'Oct',
            11:'Nov',
            12:'Dec'}
for i, row in df_precip.iterrows():
    try:
        iso3=country_code_lookups.loc[country_code_lookups.FIPS==row.Country,
'ISO3'].values[0]
    except IndexError:
        continue
    precip_visual.append([iso3, row.Month, row.Slope, row.p])
df_visual = pd.DataFrame(precip_visual, columns=['ISO3', 'month','slope','p'])
df_visual['month_text'] = df_visual.month.map(month_dict)

fig = px.choropleth_mapbox(df_visual, geojson=geojson,
                           locations='ISO3', color='slope',
                           animation_frame='month',
                           range_color=[-0.05,0.11],
                           mapbox_style="carto-positron",
                           hover_data={'slope':':.3f',
                                       'p':':.4f',
                                       'month_text':True,
                                       'ISO3':True},
                           labels={'slope': 'Change in rainfall (mm/yr)',
                                   'p': 'p statistic',
                                   'month_text':'Month',
                                   'ISO3':'Country'},
                           zoom=2)
fig["layout"].pop("updatemenus")
with open(os.path.join('..','visuals','Precipitation.html'),'wt') as f:
    f.write(fig.to_html(include_plotlyjs='cdn'))

df_world_temp=pd.read_csv('World temperatures.csv')
df_world_temp['ISO3']=df_world_temp['Country'].map(country_lookups_dict)
fig = px.choropleth_mapbox(df_world_temp.loc[(df_world_temp.Season=='SUMMER') &
(df_world_temp.Year>1980)].dropna(),
                           geojson=geojson,
                           locations='ISO3', color='MEAN_T_AVG',
                           animation_frame='Year',
                           range_color=[0,45],
                           mapbox_style="carto-positron",
                           #hover_data={'slope':':.3f',
                           #             'p':':.4f',
                           #             'month_text':True,
                           #             'ISO3':True},
                           #labels={'slope': 'Change in rainfall (mm/yr)',
                           #        'p': 'p statistic',
                           #        'month_text':'Month',
                           #        'ISO3':'Country'},
                           zoom=2)
fig["layout"].pop("updatemenus")
with open(os.path.join('..','visuals','World temperatures.html'),'wt') as f:
    f.write(fig.to_html(include_plotlyjs='cdn'))

df_SA_precip = pd.read_csv('precip.csv')
df_SA_precip = df_SA_precip.loc[df_SA_precip.Country=='SF']
df_SA_precip = df_SA_precip.pivot(index='Year', columns='Month',values='PRCP_AVG')
fig = px.imshow(df_SA_precip.loc[df_SA_precip.index>=1980].transpose())
fig.update_layout(font=dict(size=24))

with open(os.path.join('..','visuals','SA precipitation.html'),'wt') as f:
    f.write(fig.to_html(include_plotlyjs='cdn'))

for season in ['WINTER','SUMMER']:
    df_SA_temp = df_world_temp.loc[(df_world_temp.Country=='SF') &
(df_world_temp.Season==season)].melt(id_vars=['Country','Year','Season','ISO3'])
    df_SA_temp.rename(columns={'value':'Temperature'}, inplace=True)
    fig = px.scatter(df_SA_temp, x='Year', y='Temperature',
color='variable',trendline='ols')
```

```
    with open(os.path.join('..','visuals',f'SA {season} temperatures.html'),'wt')
as f:
        f.write(fig.to_html(include_plotlyjs='cdn'))
    fig.write_image(os.path.join('..','visuals',f'SA {season} temperatures.jpg'),
width=400, height=300, engine='kaleido')
```