DATA420 Assignment 1

Jing Wu
29696576

Processing

Q1

There are two parts of the data. One of those is the daily climate record, which is stored in different csv files by year, and each csv file is compressed into gzip format. Another one refers to metadata, including stations, states, countries and inventory.

The data is structured according to the directory tree below

```
/ghcnd
    countries
    /daily
        1763.csv.gzip
        1764.csv.gzip
        1765.csv.gzip
        …
        2017.csv.gzip
    inventory
    states
    stations
```

In the ghcnd directory, there are four files and a directory. Stations, states, countries, and inventory are files and daily is a directory containing one file for each year, the number of files in the daily directory is 256.

The size of all the data is 13.4G. The size of each file is provided below

countries	3.6K
daily	13.4G
inventory	26.1M
states	1.1K
stations	8.5M

We can see that the size of the metadata is very small compared to daily. The size of metadata is just about 35M but the total daily size is 13.4G, which is almost the size of the entire data.

The file size in daily directory increases with the year. The file of 1763 is only 13.2k, and the 2016 file size is 741.5M. The file size of 2017 is only 477.8M, which is much smaller than the file size of 2016. This is because the 2017 file only contained records before September 10th, not the full year record.

Processing

Q2

After loading the 1000 rows daily data of 2017 and the metadata of stations, countries, states and inventory, it is basically confirmed that the description of the data provided in the assignment 1 instruction is accurate.

Table daily contains the column DATE and OBSERVATION TIME which cannot be loaded as DateType and TimestampType directly. I loaded these two columns as StringType, which could be converted to DateType and TimestampType according to demand.

Since the metadata of stations, countries, states and inventory are stored as the fixed width text formatting, it is required to extract each column data from the text by the fixed range and using the defined schema datatype to do the data type conversion.

The rows count of each matadata table is provided below.

Table Name	Rows count
stations	103656
countries	218
states	74
inventory	595699

There are 103656 records in the table stations, which represents that there are 103656 stations in total. There are 218 records in the table countries, representing that there are 218 countries around the world. There are 74 records in the table states, representing that there are 74 states around the world. There are 595699 records in the table states, representing that there are 595699 inventory records.

There are 95595 stations that do not have WMO ID.

Processing

Q3

Since Table stations does not contain the column for country directly and the first and second letters of the station ID are the code of the country in which the station is located, I need to extract the country code and save it in column COUNTRY_CODE in the Table stations. After getting the COUNTRY_CODE, I could left join Table stations and Table countries, which could make Table station human-readable.

Table stations directly has the column STATE which represents the state code, so I can directly left join Table stations and Table states.

In order to get the information about the first year and last year of each station, I grouped the station ID and get the minimum in first year and the maximum in last year, as well as counted how many element of each station ID.

I filtered the five core elements and count separately the number of core elements for each station ID by using the pivot function. For the “other” elements, I filtered

them and get the count of “other” elements of each station ID. Lastly, I joint the core elements count table and other elements table on station ID.

The joint Table stations was saved as parquet format. Considering that the loading of the csv.gz file is cumbersome in pyspark, I only tried to save and compare the two formats of csv and parquet. The result showed that, the size of csv file was 11097860K, and the size of parquet file was just 6279471K, which was almost half size of the csv file.

After joining the 1000 rows of 2017 daily records, it showed that there was no station which was in Table daily but not in Table stations.

There are 21904999 rows in the 2017.csv.gz, which is almost 21904 times of 1000 rows. So, the computation time should be 21904 times of the time of calculating 1000 rows.

Subtract function could help if we do not use left join to get the information about which stations are in Table daily but not in Table stations.

Analysis

Q1

There were 103656 stations in total and there were 459 stations that were active in 2017.

There are 991 stations that are in the GCOS Surface Network. I filtered the not null value in column GSN_FLAG and counted.

Since the HCN_CRN_FLAG column contains three values, which were HCN, CRN and null value, I grouped them and count each of them in the HCN_CRN_FLAG column. There are 1218 stations in the US Historical Climatology Network (HCN) and 230 stations in the US Climate Reference Network (CRN).

There are 14 stations in more than one of these networks. I filtered the not null value in both columns of GSN_FLAG and HCN_CRN_FLAG, and count the total number.

In order to get the number of stations in each country, I selected the station ID and the COUNTRY_CODE from Table stations, and then I grouped the COUNTRY_CODE in and counted the station ID with each COUNTRY_CODE. Then I joint the counted result with the Table countries on COUNTRY_CODE. I also saved the Table countries in the parquet format on hdfs.

I also used the same way to count the stations in each state.

There are 25337 stations in the Southern Hemisphere. I filtered the latitudes of stations which were less than 0.

There were 57227 stations in total in the territories of the United States around the world. By using the contains () function to filter the COUNTRY_NAME involving the string of “United States”, I got all the stations in United States.

Analysis

Q2

I defined the function compute_distance(long_1, lat_1, long_2, lat_2) in Python to compute the distance between two stations by using their latitudes and longitudes. And then I used pyspark.sql.functions.udf function to wrap the native Python function in Spark.

As I needed to apply this function to compute the pairwise distances between all stations in New Zealand, I filtered all the stations in New Zealand to create a new table called NZ_stations, and CROSS JOIN the Table NZ_stations to get the pairwise stations in New Zealand. Then, I applied the wrapped Python function on the pairwise stations to compute the distance and stored in the column DISTANCE.

By ordering the result, I can know that WELLINGTON AERO AWS and PARAPARAUMU AWS were the geographically closest in New Zealand.

Analysis

Q3

We could get the size of each file under the daily directory by using the –du command. The size of 2017.csv.gz is 125257391 bytes and the size of 2010.csv.gz is 207181730 bytes. The default blocksize of HDFS is 134217728 bytes. Based on the file size of 2017.csv.gz and 2010.csv.gz, we could know that only one block was required for the daily climate summaries for the year 2017, but there should be two blocks for the daily climate summaries for the year 2010.

A stage with dependencies must be executed after the dependent stage execution is complete to execute the next stage. Since applying transformations on the data of 2017 is dependent on loading the data of 2017 and the file of 2017.csv.gz was compressed, it is impossible for Spark to load and apply transformations in parallel for the year 2017. The same is true for the data of 2010.

I checked the web user interface (mathmadslinux1p:8080), there were 2 stages in the job of counting for the year 2010, for each stage there was only one task. It was the same for the year 2017. The number of tasks executed does not correspond to the number of blocks in each input. The input of 2010 was 197.6 MB which required 2 blocks and the input of 2017 was 119.5 MB which required 1 blocks, both of the number of task was 1.

I used regular expressions in the path argument of the read command to load the data from 2010 to 2015. There were 2 stages in the job of counting for the year from 2010 to 2015, there were 6 tasks in the first stage and only 1 task in the second stage. The input of the first stage was 1130.2 MB contained 6 files, which corresponded to the tasks number of 6.

There were 6 compressed files needed to be loaded and spark set 6 tasks to load these 6 compressed files separately. Spark did not partition the single compressed input file.

We have 255 files need to be loaded when we want to deal with the entire daily files. Since Spark does not partition the single compressed input file, we could only apply transformations after the data loaded. There will be 255 tasks in the first stage.

In order to increase this level of parallelism, we could increase the number of cores in the cluster to speed up. We could also decompress the compressed file before loading it with pyspark.

Analysis

Q4

As I needed to count the number of rows for the entire daily files, I increased my resources up to 4 executors, 2 cores per executor, 4 GB of executor memory, and 4 GB of master memory when I started the pyspark shell. The total daily count was 2624027105.

I filtered all observations of five core elements in daily and counted for each element. Most observations were the records of PRCP, which was consistent with Processing Q3. The result for each element count for daily and inventory is provided below.

ELEMENT	ELEMENT_COUNT(daily)	ELEMENT_COUNT(inventory)
PRCP	918490401	101614
TMAX	362528096	34109
TMIN	360656728	34012
SNOW	322003304	61756
SNWD	261455306	53307

There are 7528188 observations of TMIN do not have a corresponding observation of TMAX.

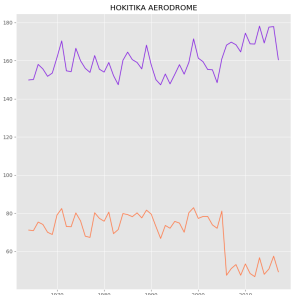
Firstly, I created a pivot table for TMIN and TMAX which is grouped by station ID and DATE and counted each of TMIN and TMAX. Then I filtered the rows with null value of TMAX and not null value of TMIN.

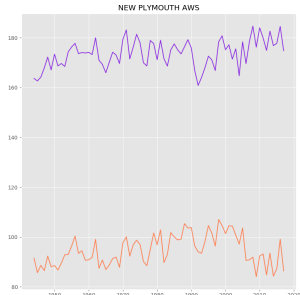
There are 26625 stations contribute to these observations, 910 of which are GSN, 1181 of which are HCN and 23 of which are CRN.

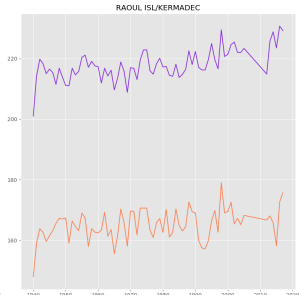
There are 447017 observations in daily are the records for TMIN and TMAX in New Zealand. These observations cover the records of 78 years, from 1940 to 2017.

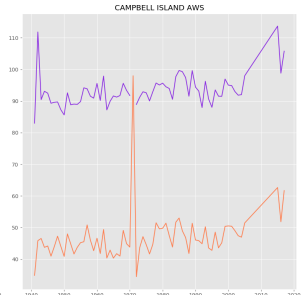
I used Python to plot the time series of TMIN and TMAX on the same axis for each station in New Zealand. The plots are provided in the next page (Figure 1).

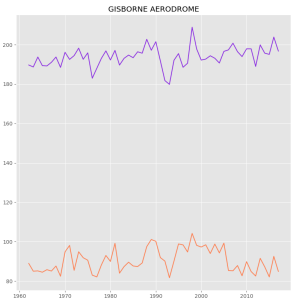
Figure 1. Time series of TMIN and TMAX of each stations in New Zealand

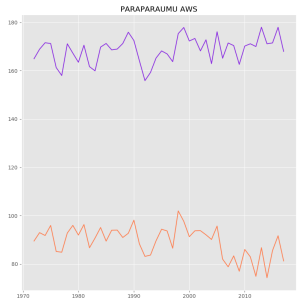

HOKITIKA AERODROME

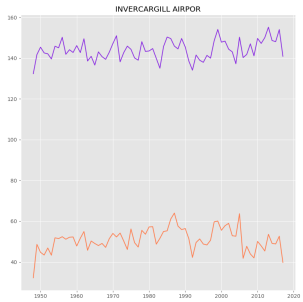

NEW PLYMOUTH AWS

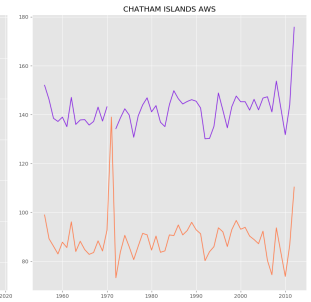

RAOUL ISL/KERMADEC

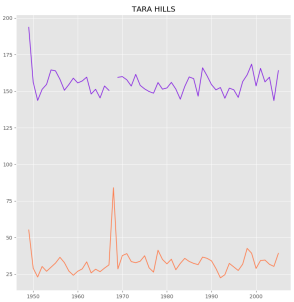

CAMPBELL ISLAND AWS

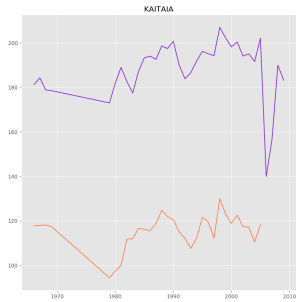

GISBORNE AERODROME

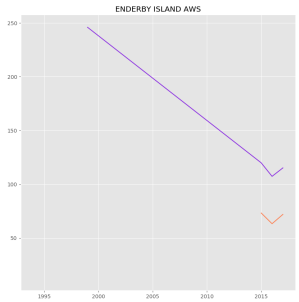

PARAPARAUMU AWS

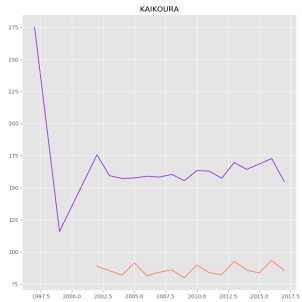

INVERCARGILL AIRPOR

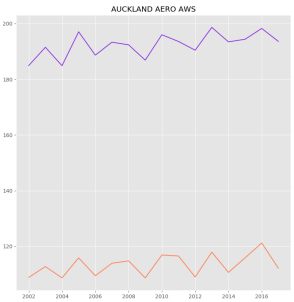

CHATHAM ISLANDS AWS

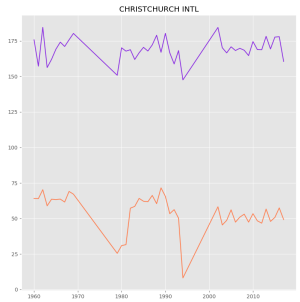

TARA HILLS

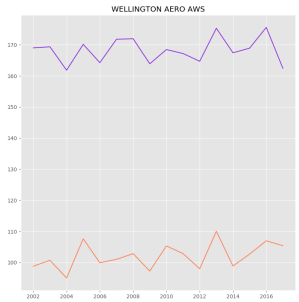

KAITAIA


ENDERBY ISLAND AWS

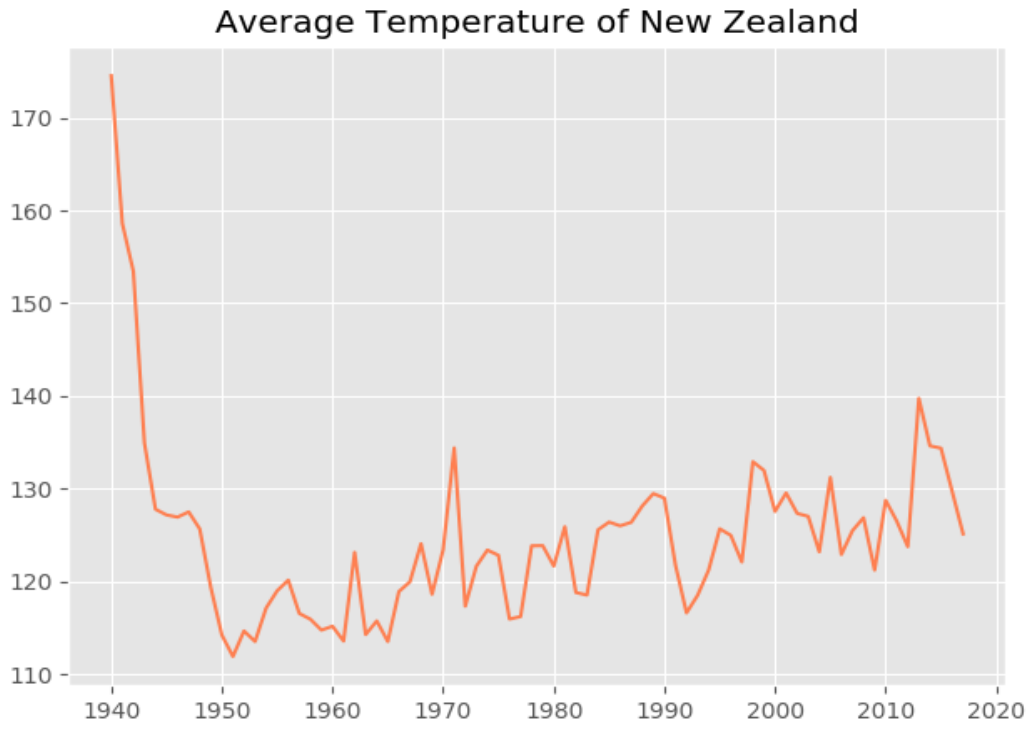

KAIKOURA


AUCKLAND AERO AWS

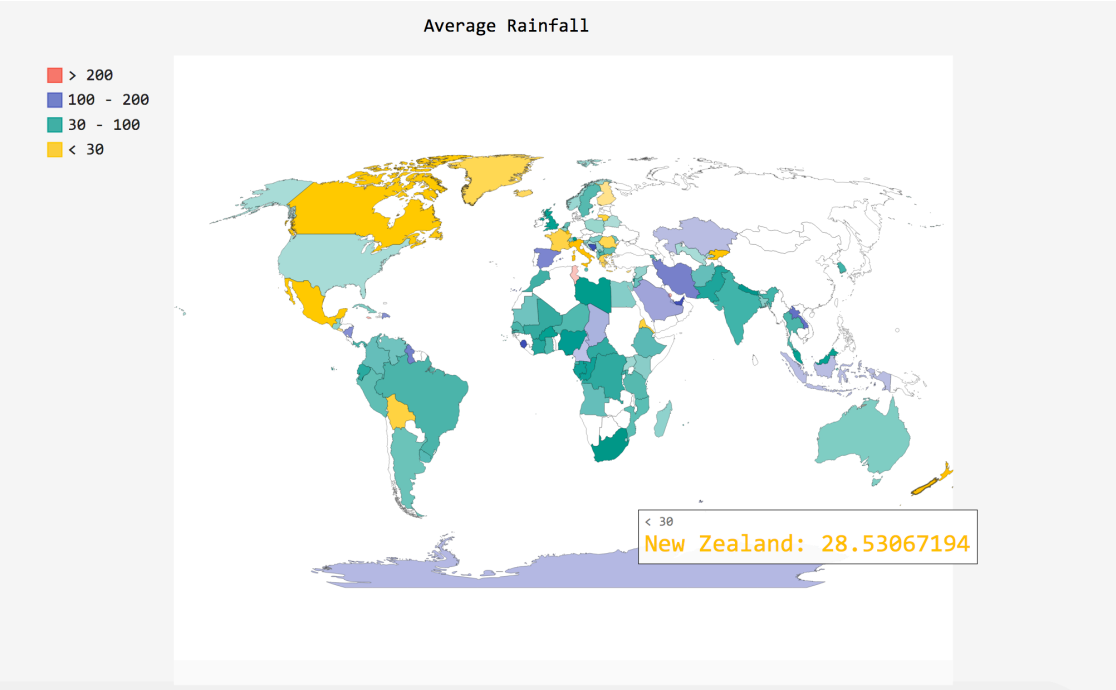

CHRISTCHURCH INTL


WELLINGTON AERO AWS

The average temperature time series for the entire country is provided below.



I grouped the observations by year and country and save the result in pyspark. Based on the result, I used Python pygal library to plot a choropleth to color a map according to average rainfall for each country. The plot is provided below.



Challenges

Q1

In order to investigate the coverage of the other elements temporally, I selected the DATE and ELEMENT columns in the Table all_daily, filtered all the other elements and extracted the year from the DATE column. Then I separately get the earliest year and latest year of each other elements, and compute the length of years they covered.

The elements “DWPR”, “MDPR” and “DAPR” are the longest recorded items with 186 years coverage and all started being recorded in 1832.

I selected the columns ELEMENT and LATITUDE in the Table all_daily and added a column to group each record depending on their latitudes into 6 zones, they are ARCTIC CIRCLE, NORTHERN TEMPERATE ZONE, NORTHERN TROPICS, SOUTHERN TROPICS, SOUTHERN TEMPERATE ZONE and ANTARCTIC CIRCLE.

There are 60 other elements being captured in ARCTIC CIRCLE and the top five elements with the most records were “TAVG”, “TOBS”, “PGTM”, “WSFG” and “WDFG”. On the contrary, there are only 29 other elements being captured in ANTARCTIC CIRCLE and the top five elements with the most records were “TAVG”, “WSFG”, “PGTM”, “WDFG” and “WT18”. Both ARCTIC CIRCLE and ANTARCTIC CIRCLE have the most records of “TAVG”.

“TAVG” and “PGTM” are the top two elements that these six zones all recorded frequently.

NORTHERN TEMPERATE ZONE capture 127 elements but SOUTHERN TEMPERATE ZONE only capture 8 elements, which are “DWPR”, “MDTX”, “DATX”, “DATN”, “MDPR”, “MDTN”, “TACG” and “DAPR”.

Challenges

Q2

There are total 7757511 observations failed quality assurance checks. The most failed quality checked type is “I”, which means the observation failed internal consistency check. Through the Analysis Q4 we could know that there were 2624027105 observations of the whole daily table, the failed observations only accounted for 3%. The quality of the dataset is good.