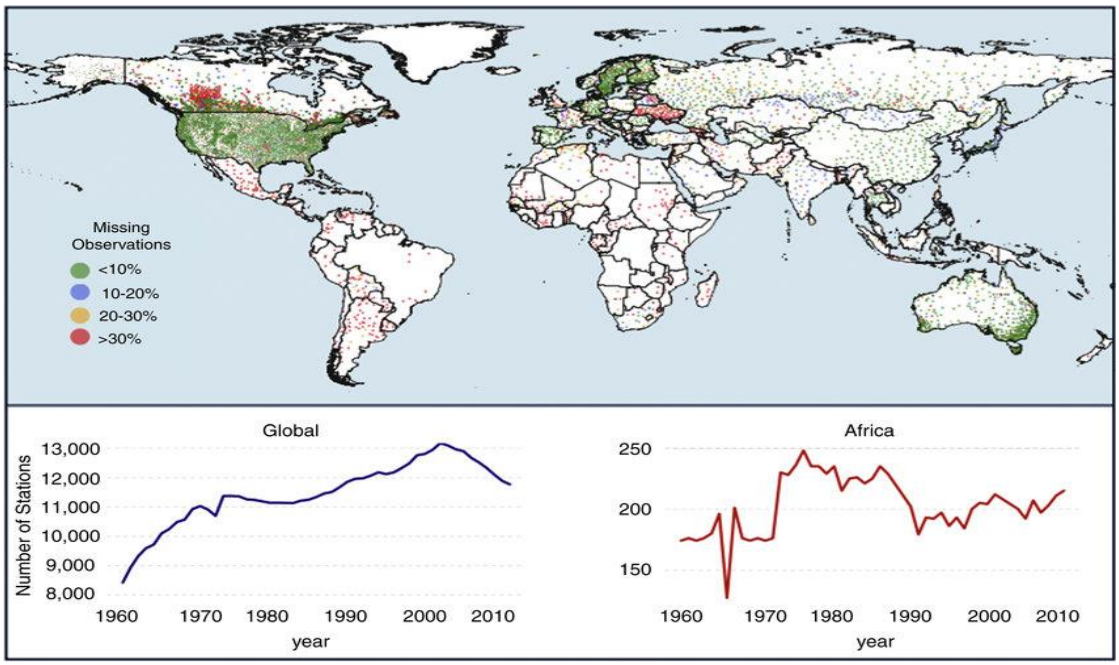
DATA420-24S1 (C)

Assignment 1

GHCN Data Analysis using Apache Spark



Chathurangi Godahewa Gamage

Table Of Content

Background

This assignment was directed to analyze the climate data collected from different global stations focusing on the Global Historical Climatology Network (GHCN) daily data set. This GHCN data set extends over 250 years, consisting of over 100,000 stations in 200 countries and territories. It contains different daily features like precipitation, minimum and maximum temperature, snowfall, and snow depth. This rich data set paves the way to learn about weather patterns, conditions, and trends.

The primary aim of this assignment is to get a better understanding of Apache Spark and its usage. Throughout the assignment, we can use Apache Spark techniques for the processing and analysis sections of the exercise using the GHCN dataset. Initially, GHCN daily documentation - Global Historical Climatology Network (GHCN) and GHCN Daily are immensely helpful to get a better understanding of the data set. Apart from that, Apache Spark Documentation: Apache Spark Official Documentation is very helpful to get a basic understanding of Apache Spark. After I started to do the Processing and Analysis parts of the assignment, Spark by Examples – Apache Spark Tutorial became the most useful resource to learn about the related examples for the questions and how they apply to the question.

Initially, the biggest challenge I faced was, that I didn't have a clear idea about what to do with this assignment. I was terrified after seeing how big the assignment was. Then with the help of lab sessions related to assignments, James' guidance with help sessions, and especially my friend's guidance I have got little by little understanding of the assignment step by step. I now mostly understand the concept and have an idea of what I'm doing. I also used chatGPT to understand about some questions. I think this whole assignment is a big challenge to me. However, I tried my best to face the challenge and complete the assignment.

Processing

Q1

How data structured

The GHCN (Global Historical Climatology Network data set is outlined with the "hdfs:///data/ghcnd" directory because we hosted data in HDFS (Hadoop Distributed File System). This data set consists of different types of information, including daily climate summaries, metadata about stations, countries, states, and inventory. When we consider the daily climate summary data, it along with based on the year. (e.g. '2023.csv.gz, 2022.csv.gz…). All the files are GZIP-compressed to save space.

The stations' directory, it includes a text file (ghcnd-stations.txt), which consists of metadata including ID, name, latitude, longitude, and so on. In states, it also contains a text file (ghcnd-states.txt), with code, and name as metadata which is like the countries text file (ghcnd-countries.txt). The inventory directory also includes a file as text (ghcnd-inventory.txt), which consists of different climate elements spanning different years.

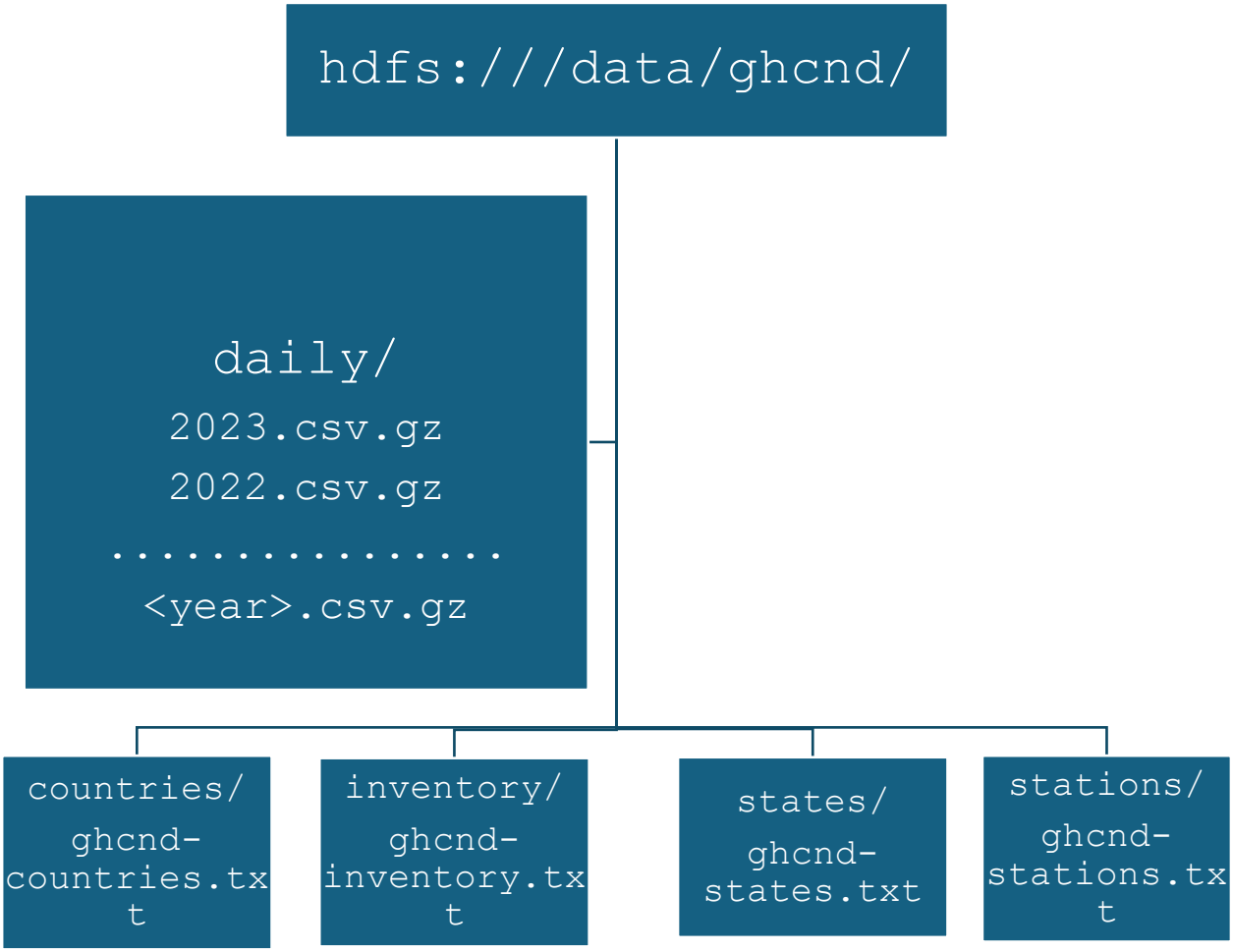


figure 1 – data structure in GHCND data

How many years are contained in daily, and how does the size of the data change?

The years from 1750 to 2024. That means we have 263 years of data daily.

Size change

In 1750, it started with 505.6KB (compressed file size) and it was reduced in the late 17's, and little by little it increased up to 999.5 KB in 1877 and 1878, the size change into 1.2 MB. The sizes change gradually between the late 1800s and early 1900s. The 20th century marked a period of data size increase. This reached more than 100MB in early 1900 and expanded in the next decades. During the late 20th century

and early 21st century, file size increased to gigabytes. In 2022, the file size in 1.3G and it was reduced by 0.1GB in 2023 which has 158.7 MB in compressed file. But in 2024, the file size shows 209.8MB (without compression), because it's still the first 4 months of the year. In conclusion, I can say the file size gradually increases when it comes to the modern era, because of the high level of analysis and research in climate change.

What is the total size of all the data? How much of that is daily?

File size (compressed)	File Size (without compress)	Data files /data/ghcnd/
12.4 G	98.9 G	/daily
3.6 K	28.6 K	/countries.txt
32.8 M	262.3 M	/inventory.txt
1.1 K	8.5 K	/states.txt
10.3 M	82.7 M	/stations.txt
Total = 12.44 G	Total = 99.26 G	

Table1: sizes of data

Daily file size (compressed) = 12.4/ 12.44

Daily file size (not compressed) = 98.9/ 99.26

Q2

Daily Schema consists of ID (station Identifier), DATE (date of the observation), ELEMENT (type of the observation – TMAX, TMIN...), VALUE (observation value – temperature, precipitation volume, ...), M_FLAG (measuring flag), Q_FLAG (quality flag), S_FLAG (source flag), and OBS_TIME (observation time).

What data types did you end up using for the schema and why?

Daily Schema consists of String type and float type.

ID – String Type: This denotes station Identifier, and it's a mix of letters and numbers, which is suitable as strings.

DATE – String Type: This can be easier if it has different data formats.

ELEMENT – String Type: This is alphanumerical and categorical which gives types of observations and is suitable as a string.

VALUE – Float Type: This allows fractions to be represented accurately. Normally, observation values can have decimal numbers. Therefore, float type provides more accuracy for the value than using Integer Type.

M_FLAG, Q_FLAG, and S_FLAG – String Type: This illustrates different flags in the daily schema, they can contain alphanumerical characters of observation data. String Type is more suitable because they are categorical, not quantitative.

OBS_TIME – String Type: Observation time representing format can be varied. Therefore, String Type can be more flexible and convenient.

How many rows are in each metadata table? How many stations do not have a WMO ID?

File Name	Number of Rows
stations	125983
countries	219
states	74
inventory	747382

Table2: No. of rows in metadata table

Number of stations do not have WMO ID = 118023

Q3

In this section firstly, I have extracted two characters ‘country code’ from each station code using the ‘withColumn’ method. Then I saved the output. After that, using the output that I have got as station data, left join with my countries data frame. Before that, I renamed the ‘name’ column in my countries data frame into ‘country name’. Thereafter, I joined, the new data frame with the state data frame. In that process I Joined two data frames by matching ‘states code’ in my states df and ‘state’ column in my new df. Then I dropped the ‘state’ column to avoid confusion between similar columns.

Then I left joined the station data frame and states data frame using country code ‘US’ and saved the new output.

what was the first and last year that each station was active and collected any element at all? -> appendix C

In this question, I have used my inventory data frame and used groupBy function, I have grouped it using ‘ID’ and used the aggregate function to get min and max years in each station.

How many different elements have each station collected overall? -> appendix C

To make a new data frame I first used my Inventory data and grouped it by ‘ID’. Then I used the countDistinct function in my ‘element’ column to count the number of different elements in each station. After that, I made a new_inventory_data data

frame to join the above two data frames (years and element count) using ‘ID’. Then save the output.

Thereafter using 5 core elements, 'PRCP', 'TMAX', 'TMIN', 'SNOW', and 'SNWD', I made a list and using that list I created two different data frames with the count of Core elements and other elements by filtering ‘element column in inventory data frame and group by ‘ID’. After that, both outputs are joined to the new_inventory_data data frame separately and saved.

How many stations collect all five core elements? 20467

Here I have filtered my core elements data frame above, using the core_element column which is equal to 5 and then got the count.

How many collect only precipitation and no other elements? 35662

Using the filter function, I have filtered the ‘element‘ column using the ‘PCRP’ type.

Then I made another data frame to collect elements. To get the results, I have grouped the rows in inventory data using ‘ID’. Then I used to collect.set (pyspark SQL) function which I used to collect all unique elements in each group as a set (appendix). Then I joined this data frame to my new_inventory_data df and saved it. Thereafter, I made an enriched station data frame using the new_inventory_data and country_state_data data frames and saved the output. Further, I have left joined 1000 rows subset of daily data with enriched station df.

Determining Missing Stations Using and not using LEFT JOIN

With using LEFT JOIN and without using LEFT JOIN there are 7 stations found which is in the subset of daily, but not in stations.

How expensive do you think it would be to LEFT JOIN all daily and stations?

I think it’s an expensive and time-consuming procedure because, on the one hand, the GHCN daily data set is significantly large, and records time going over centuries with more than 100 000 stations all over the world. Although the stations’ data set is smaller, the processing of total volume might need computational resources to do the LEFT JOIN. On the other hand, this LEFT JOIN process needs to compare each row in the daily data set with stations to find matching “ID” s. For this kind of large dataset, it requires a long time and power to process.

Analysis

Q1

How many stations are there in total?

To find the total number of stations I have used the enriched_stations data frame, which I have made in the processing section, collecting other data frames. Also, I have used ‘ID’ to select stations and I got **125983** stations in total.

How many stations were active so far in 2024?

Here I have applied the filter option to inventory data and filtered the number of stations which is greater than or equal to the 2024 in first year of inventory data and

less than or equal to 2024 in the last year of inventory data. As a result, it gives **31839** stations**.**

Counting stations in each network.

I have used my enriched data (which I have made in the processing section) to filter each station separately column-wise. Then I applied the. count() option to count the number of stations in each network separately.

Stations in GSN	991
Stations in HCN	1218
Stations in CRN	234

Table3: No. of stations in each network

Number of stations in more than one of these networks?

I have filtered stations that are in GSN and marked as either HCN or CRN. After that, I got the whole count by applying the. count() option to the variable (stations_in_multiple_network), and finally, I got 15 as the count.

Thereafter I counted the total number of stations in each country by applying the groupBy option to the stations_countries_df variable's (which I got in the processing section) country name and made another column as num_stations to put the aggregated count of the stations. Thereafter I have again joined those countries_with_station_counts variables into my countries_data data frame using left join. Finally, I have saved the output using the Parquet method.

Likewise, I have counted the total number of stations in each state. As same as the above procedure I have used station_states_df (data frame from the processing section) to the group and count the number of stations. Then I joined the result into my states_data data frame and finally saved the output.

How many stations are there in the Southern Hemisphere?

To find the answer I have used my enriched_stations_df data frame and filtered the column latitude values less than 0 because latitude values are negative in the Southern Hemisphere. To get the number of stations there, I have used the. count() option and I have got **25316** as a result.

How many stations are there in total in the territories of the United States around the world, excluding the United States itself?

In this case, I have used "Country name" contains as ["United States"] to filter my stations_countries_df data frame. To get the count of US territories I have used 'ID' to select. Finally, I have got the result as **383.**

Q2

In this section, first I have defined and registered a User Define function (UDF) in PySpark to calculate the distance between two geographical points using the Haversine formula. After that I used cross join to calculate the distance between all the pairs of stations which I have taken a subset of the enriched_station_df dataset using previously defined UDF using the haversine formula.

After that I filtered New Zealand stations using my enriched_station_df using "Country_Code == "NZ". Then I used the nz_stations data frame to find the number of stations in New Zealand selecting "ID" and got 15 stations as a result.

Then I cross-joined all the possible pairs of stations in the subset of New Zealand stations. Then I calculated the pairwise distances by adding the distance variable as a new column.

What two stations are geographically the closest together in New Zealand?

I have applied the filter function to the nz_distances data frame using IDs' in station1 and station2. Then I made the distance_km in ascending order and limited it to 1. I have got results as ID - NZ000093417 (PARAPARAUMU AWS.) and ID-NZM00093439 (WELLINGTON AERO A..) are the closest stations geographically and the distance between the closest stations in New Zealand is 50.53km.

Q3

default block size of HDFS: 134217728

Compressed daily file size in 2023 is 158.7 MB and uncompressed file is 1.2 GB. In 2024, the compressed file size is 26.2 MB, and the uncompressed file size is 209.8 MB. (It's still small in file size because it is nearly the first trimester of the year)

How many blocks are required for the daily climate summaries for the year 2024?

1 block

/data/ghcnd/daily/2024.csv.gz 27492832 bytes, replicated: replication=8, 1 block(s)

How many blocks are required for the daily climate summaries for the year 2023?

2 blocks

/data/ghcnd/daily/2023.csv.gz 166367488 bytes, replicated: replication=8, 2 block(s)

What are the individual block sizes for the year 2023?

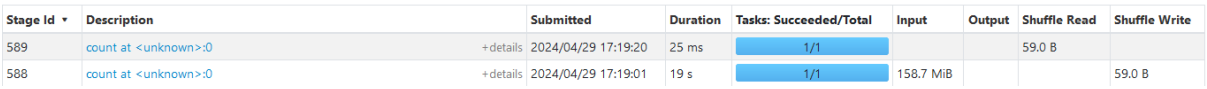
In 2023, the file size is 166367488 bytes, which is approximately 158.66MB (1MB = 1024 *1024 Bytes). It has 2 blocks which are divided in 128MB in one block and the rest of the 30.66MB in the other block.

Using the daily.csv files related to 2023 and 2024, found the no. of observations.

Number of observations in 2023 – 37395852

Number of observations in 2023 – 6061827

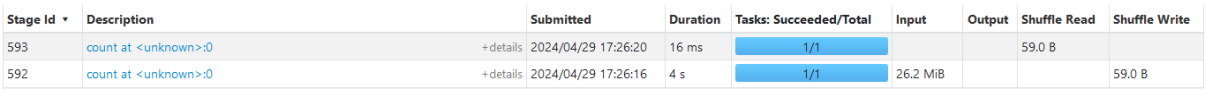
How many tasks were executed by each stage of each job?



Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
589	count at <unknown>:0 +details	2024/04/29 17:19:20	25 ms	1/1			59.0 B	
588	count at <unknown>:0 +details	2024/04/29 17:19:01	19 s	1/1	158.7 MiB			59.0 B

figure 2 – task executed in daily 2023.

In the 2023 daily CSV file, when loading it executed 1 task in the first stage, and when counting the number of observations in the second stage, it also executed 1 task. Altogether, 2 tasks were executed in 2 stages.



Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
593	count at <unknown>:0 +details	2024/04/29 17:26:20	16 ms	1/1			59.0 B	
592	count at <unknown>:0 +details	2024/04/29 17:26:16	4 s	1/1	26.2 MiB			59.0 B

Figure 3– task executed in daily 2024.

In the 2024 daily CSV file, when loading it executed 1 task in the first stage, and when counting the number of observations in the second stage, it also executed 1 task. Altogether, 2 tasks were executed in 2 stages.

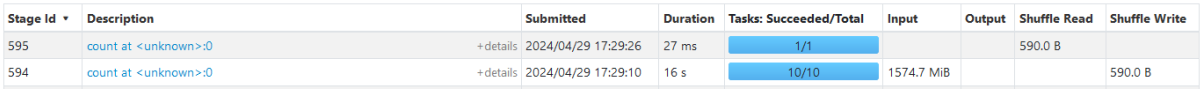
Did the number of tasks executed correspond to the number of blocks in each input?

The number of tasks executed can correspond to the number of blocks in each file if the input file is not compressed, and if it's stored in HDFS because HDFS divides large files into blocks and distributes them across the filesystem. But compress files like GZIP('.gz'), which is not divided, the complete file is read as a single block by a task, without considering the file size. As a result, for compressed files like '.gz', the number of tasks, basically corresponds to the number of files rather than the number of blocks.

Total number of observations from 2014 to 2023:

First, I have listed the daily path using for loop, range in Python. Then use the count () function to get the number of observations from 2014 to 2023 as **369419065.**

Number of partitions (and likely number of tasks for reading):



Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
595	count at <unknown>:0 +details	2024/04/29 17:29:26	27 ms	1/1			590.0 B	
594	count at <unknown>:0 +details	2024/04/29 17:29:10	16 s	10/10	1574.7 MiB			590.0 B

Figure 4 – tasks executed in the year 2014 to 2023.

To find the no. of partitions in the data frame of daily (2014-2023) I used. rdd.getNumPartitions() and got the result as 10. Here, the number of partitions equal to number of tasks executed. Therefore, when the file loads from 2014 to 2023 (10 years) it is partitioned into 10 tasks in the first stage, and in the second stage it executes one task to get the count. Altogether, number of tasks executed are 11.

Explain how Spark partitions input files that are compressed.

Compressed files are not like uncompressed files. They cannot be split, especially now here we are using a common file compress method as GZIP ('.gz'). This means that for parallel processing, Spark cannot split a single compressed file over multiple partitions. Therefore, every compressed file turns into one partition. Also, to process the data, Spark allocates one task per partition. This is the reason for parallelism and the performance because the concurrency level becomes shorter by the quantity (number) of compressed files than the complete size of the cluster's capacity.

Based on parts (b) and (c), how many tasks do you think would run in parallel when loading and applying transformations to all daily? Can you think of any practical way you could increase this either in Spark or changing how the data is stored in HDFS?

Hence daily data is stored as compressed files (GZip, .gz), loaded as single partitions. That means the number of compressed files of daily data related to parts (b), and (d), have a similar number of tasks running parallel during the load phase, which is forced by the available cores and executors. Cluster configuration in spark also being a reason to limit the parallelism. To increase parallelism, we can enhance the performance of loading and processing daily. We can use a splittable compression format (to split a single large file into numerous partitions) like BZIP or LZO to increase the performance. Also, we can adjust the cluster configuration in Spark by using more executors with more cores. Furthermore, we can repartition the data after loading.

Q4

The number of rows in daily?

First, I loaded all the daily data in the CSV file and changed the schema route, and after that using the count() function have counted the number of rows in the daily data frame as 3103954141.

How many observations are there for each of the five core elements?

I first listed 5 core elements and then filtered my new daily data frame using the 'Element' column and my list of core elements.

Element	Count
PCRP	1069105193
TMAX	455946095
TMIN	454759421
SNOW	353904309
SNWD	297846434

Table4: Observations in 5 elements

Which element has the most observations? PCRP

To get the answer as **PCRP** I have sorted the element_count data frame using the orderBy() function and made the count in descending order to get the most viewed element first.

Determine how many observations of TMIN do not have a corresponding observation of TMAX. 9322723

First, I have filtered TMIN and TMAX data using the 'element' column in my new daily data frame. Then I limited the filtered data frame because it took so much time to load the data frame. Then using that limited data frame I made a new df by

grouping ‘ID’ and “DATE”. Thereafter to find the missing TMAX stations, used new df and filtered TMIN which is greater than 0 and TMAX == 0.

How many unique stations contributed to these observations? 27927

To find the unique stations, I have selected the missing TMAX data frame using ‘ID’, and then count ().

How many observations are in NZ = 485520

Before finding the number of observations in NZ, I joined the new daily data frame and enriched_stations_df (which was made in the processing section), using ‘ID’ and left join, and made a complete data frame. Then I filtered my complete data frame according to the column element equal to TMIN and TMAX. Using the filtered data frame, I have filtered New Zealand related to the country code. Then I saved the result. Using that I have used the count () function to get the observations in NZ.

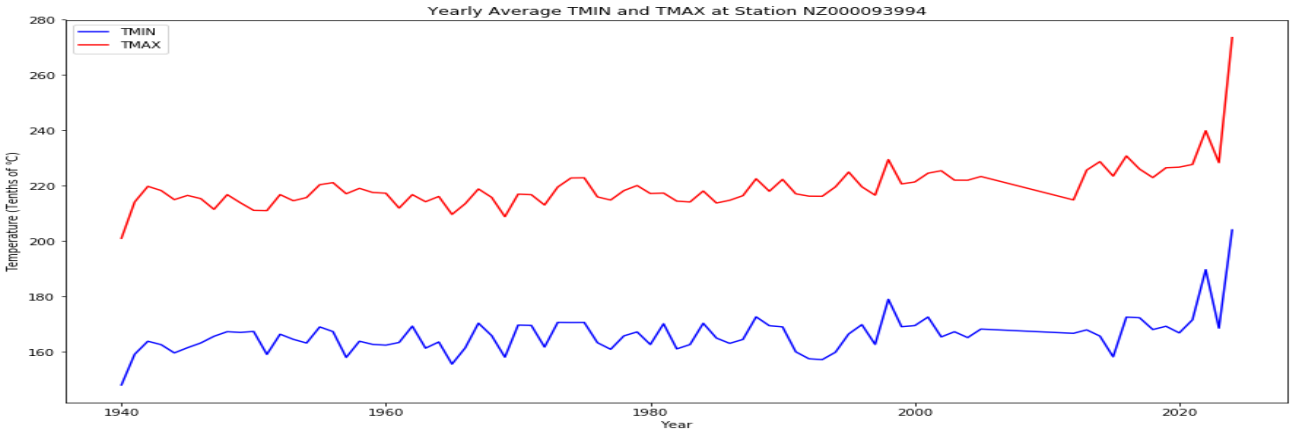
How many years are covered by the observations? 1940 to 2024

To obtain the answer first I have changed the type of date into date type because it already had string type and had yyyyMMdd format in my filtered New Zealand data frame. Then I extracted the date using the year() function in PySparkSQL in the date column. After that, I have used the aggregate function to find the minimum year and maximum year of the observations.

There I have copied my NZ data frame into Local and counted the number of rows in the part files using the wc -l bash command.

Plot time series for TMIN and TMAX together for each station in New Zealand

First, I have imported matplotlib and pandas into my notebook. Then converted the spark data frame into the pandas’ data frame. After, convert ‘DATE’ into a datetime object, and then extract the year. Thereafter, two data frames have tmin and tmax. Then grouped the data frames by year and calculated the average temperature. Further, combined two data frames using ‘ID’ and ‘YEAR’. After that using Python, I plotted the time series for tmin and tmax together for each NZ station.

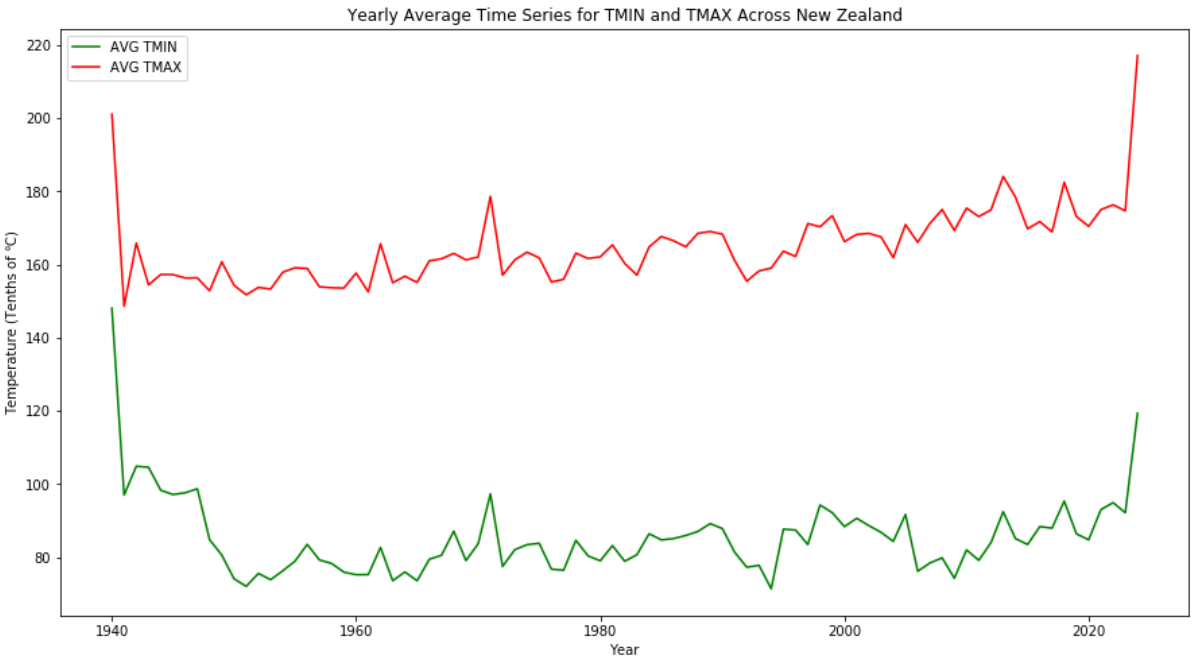


Plot1 – Time series for TMIN and TMAX for each station

We have 15 New Zealand stations, and the rest of the 14 plots are included in supplementary materials.

Plot the average time series for TMIN and TMAX together for the entire country.

Using the combined data frame that I used above to plot the time series, I grouped by “YEAR” and calculated the countrywide average for tmin and tmax. Then using Python made the plot.



Plot 2 – Average time series for TMIN and TMAX together in all countries.

Group the precipitation observations by year and country. Compute the average rainfall in each year for each country. -> appendix D

To find out the average rainfall in each country each year I have used my previous complete data frame (combination of enrich df and daily data frame) and using this I have created a new data frame in which I filtered the ‘ELEMENT’ column into “PRCP”. Then the created data frame is grouped by ‘country name and ‘date’ and using the aggregate function sums up the total rainfall in each country related to the date and summed up a unique station count. Thereafter after diving total rainfall by unique station count, have found the average rainfall. Then I created another column called ‘YEAR’ by filtering the ‘date’ column in the data frame. Finally, to calculate the exact average rainfall, I have grouped the above data frame into ‘country name’ and, ‘year’, and aggregated the sum of average rainfall related to year.

Which country has the highest average rainfall in a single year across the entire dataset?

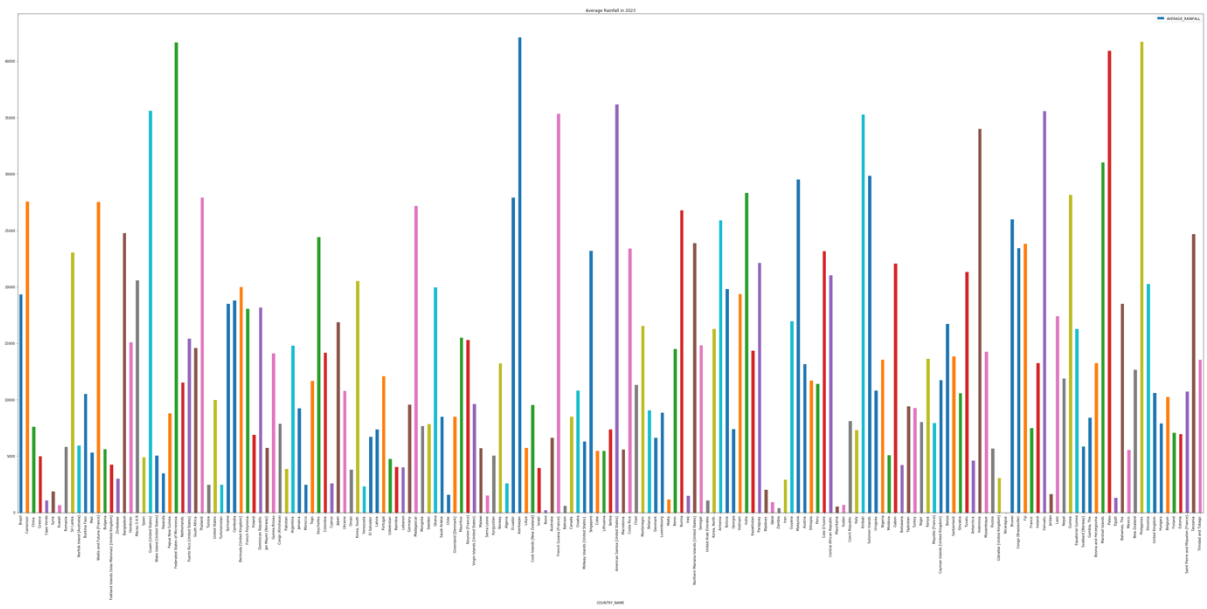
The country with the highest average rainfall is Caledonia [France] in the year 1999 with an average rainfall of 176744.0mm.

Is this result sensible? Is this result consistent with your previous analysis?

New Caledonia in France has a tropical climate which includes periods of intense rainfalls. However, the results I have received as average rainfall is comparatively high because on Earth typically annual rainfall in wet countries is around 10000mm. Therefore, I don’t think my results are very sensible and consistent with the previous results.

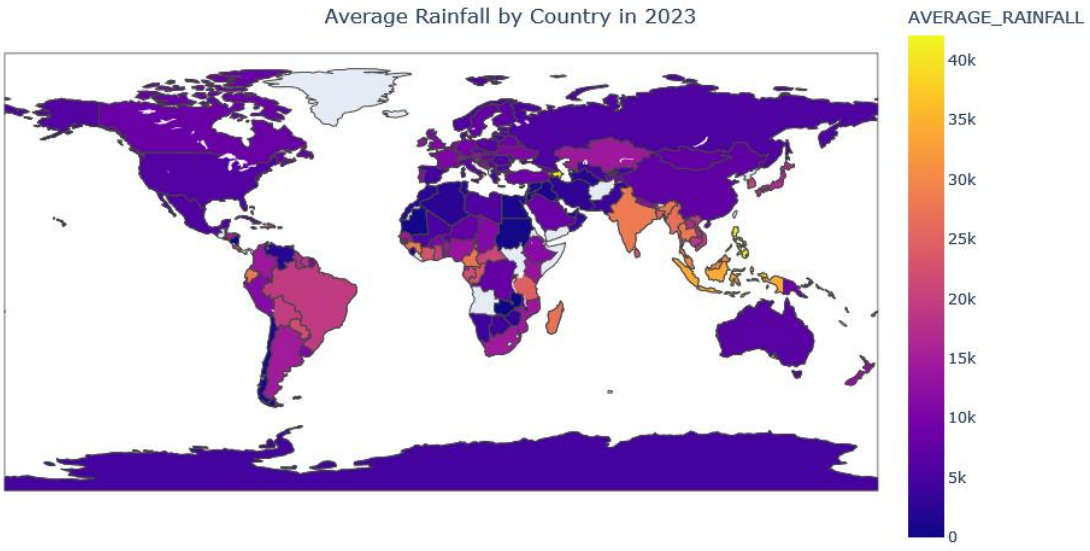
Filter average rainfall 2023

Here I have filtered my average rainfall data frame using the 'YEAR' column which is equal to '2023'.



Plot3 – Average Rainfall in 2023

Find an elegant way to plot the average rainfall in 2023 for each country.



Plot4 – Average Rainfall in 2023 (ELEGANT WAY)

According to my plot, there are some countries with missing values including Greenland, Iceland, Somalia, Sudan, Afghanistan, etc.

Conclusion

This assignment consists of two major sections including processing and analysis. Majorly, this report gives a comprehensive analysis of the GHCN data set using Apache Spark. In the processing part, first I got to know about data sets using the commands in Windows PowerShell and Pyspark notebook. It provided information about how data size varied in compressed and uncompressed phases, and in which manner (CSV or txt). Then eventually using daily data sets and metadata, we have created several different data frames and using the 'left join' function we have combined lots of data frames considering comment features in each data frame (e.g. "ID"). In this Processing phase, I gained much knowledge about how to join data frames and how to save them in CSV form or parquet form. I have used the data frame that I made as an enriched data frame (by joining lots of different data frames) to answer several questions in the analysis. In the analysis section, I have done several analyses based on the output data frames I have made in the processing section.

In the analysis part, I have gained knowledge about cross-joining, mostly filtering, and got a better understanding of the structure and format of daily. Using Apache Spark, got more knowledge about executors and, the number of tasks and jobs related to the analysis. As well as learnt how to partition in RDD and how to get to know whether the files we use are healthy or not related to its detailed information. In the final section of the analysis part, I got to know about how to deal with large data sets how to extract columns to make different data frames how to use them and how to make different plots according to the questions asked.

When I was doing Q4 in the analysis section I was depressed and couldn't be able to overcome the trauma. That time my friend Uditha guided me to solve the questions. In the same part, I have faced the issue of the time it took to load data frames. I felt that's because I have done most of the processing parts and analysis parts in the same notebook. To overcome that problem, I have done the Q4 part in a separate notebook. When I copied the output from HDFS to my local directory and found the total number of rows in the part file. However, I didn't get the same total number of observations I got using Spark.

Overall, this assignment enhanced my ability to manage large data sets using Apache Spark and got a good understanding of climate data analysis.

References

1. *Global Historical Climatology Network daily (GHCNd)*. (2024, April 17). National Centers for Environmental Information (NCEI). https://www.ncei.noaa.gov/products/land-based-station/global-historical-climatology-network-daily
2. *Examples | Apache Spark*. (n.d.). https://spark.apache.org/examples.html

3. GeeksforGeeks. (2022, September 5). *Haversine formula to find distance between two points on a sphere*. GeeksforGeeks. https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/
4. Dutheil, C., Menkès, C. E., Lengaigne, M., Vialard, J., Peltier, A., Bador, M., & Petit, X. (2020). Fine-scale rainfall over New Caledonia under climate change. *Climate Dynamics, 56*(1–2), 87–108. https://doi.org/10.1007/s00382-020-05467-0
5. *Apache Spark™ - Unified Engine for large-scale data analytics*. (n.d.). https://spark.apache.org/
6. Adler, R. F., & Gu, G. (2024). Global precipitation for the year 2023 and how it relates to longer term variations and trends. *MDPI*. https://doi.org/10.3390/atmos15050535
7. *chatGPT4*. (n.d.). chatGPT. https://chat.openai.com
8. *Grammarly: free AI writing assistance*. (n.d.). https://www.grammarly.com/

APPENDICES

Appendix A: Data

Data - hdfs:///data/ghcnd

Data is collected from the Global Historical Climate Network (GHCN) and GHCN Daily, and data is stored in HDFS. Daily climate data is stored in csv.gz format and other metadata including stations, countries, inventory, and states are stored in txt format.

Appendix B: Data Processing – Notebook 1 (Processing Q1)

Appendix B is Notebook 1 consists of Q1 in the processing section. It consists of how data is structured, how data size changes daily and how the sizes are distributed in each daily and metadata. Everything is done using HDFS commands.

Appendix C: Data Processing and Analysis- Notebook 2 (Processing Q2 to Analysis Q3)

Appendix C is Notebook 2 consists of codes related to the rest of the processing questions (Q2 and Q3) which combined the daily climate data with metadata tables by joining, filtering, and group functions.

ID	FIRST_ACTIVE_YEAR	LAST_ACTIVE_YEAR
AGE00147719	1888	2024
ALE00100939	1940	2000
AQC00914873	1955	1967

Table1 – First and Last year of activity of each station

ID	NUM_ELEMENTS
US1COEP0384	5
US1COEP0420	4
US1COJF0361	6

Table 2 – Count of different elements collected by each station.

Appendix D – Data Analysis – Notebook 3 (Analysis Q4)

Appendix D in Notebook 3 consists of the rest of the (final) analysis part. It has plots and graphs related to rainfalls station-wise and country-wise.

COUNTRY_NAME	YEAR	AVERAGE_RAINFALL
Albania	1983	11308.833333333334
Algeria	1942	1433.3333333333
Algeria	1963	2425.0333333333338
Algeria	1968	1872.4999999999995
Algeria	1987	3412.3343925805984
Algeria	2005	3122.132728313118
American Samoa [U...	1974	30159.666666666668
Argentina	1977	8897.641975313432
Argentina	2009	19133.697777898822

Table 3 – Average rainfall

Appendix E – Supplementary Graphs of station

Appendix E in pdf document consists of the rest of the 14 station plots related to tmin and tmax as visual presentation.