DATA420-24S1 (C)

Assignment 1

GHCN Data Analysis using Spark

Due on Friday, April 26 by 5:00 PM.

If you have any questions about the assignment use the forum or send me an email.

LEARN

Help forum for Assignment 1
Report upload (pdf)
Supplementary material upload (zip, limited to 10 MB)

Instructions

- You are encouraged to work together to understand and solve each of the tasks, but you must submit your own work.
- Any assignments submitted after the deadline without obtaining an extension will receive a 50% penalty.
- The forum is a great place to ask questions about the assignment material. Any questions will be answered by the lecturer or tutors as soon as possible, students can also answer each other's questions, and you can all benefit from the answers and resulting discussion as well.
- All data under `hdfs:///data/` is read only. Please use your own home directory to store your outputs e.g. `hdfs:///user/abc123/outputs/`.
- I recommend that you use the pyspark notebook provided on LEARN as this will make it easier for you to develop code and check outputs interactively.
- Please be mindful that you are sharing cluster resources. Keep an eye on Spark using the web user interface mathmadslinux2p:8080, and don't leave a shell running for longer than necessary. Feel free to change the number of executors, cores, and memory you are using, but be prepared to scale down if others need to share those resources. If you need to ask someone to share their resources, email them using their user code (e.g. abc123@uclive.ac.nz). If they don't reply within a reasonable time, please email me and I will kill their shell.

Reports

- The report should be submitted as a single pdf file via LEARN. Any additional code, images, and supplementary material should be submitted as a single zip file via LEARN. You should **not** submit any outputs as part of your supplementary material, leave these in your home directory in HDFS.
- You should make sensible choices concerning margins, font size, spacing, and formatting. For example, margins between 0.5" and 1", a sans-serif font e.g. Arial with font size 11 or 12, line spacing 1 or 1.15, and sensible use of monospaced code blocks, tables, and images.
- The body of your report should be no more than 10 pages long, including figures and tables in the body but excluding a cover page, table of contents, references, appendices, and supplementary material. You need to be accurate and concise and you need to demonstrate depth of understanding.
- You should reference any external resources using a citation format such as APA or MLA, including any online resources which you used to obtain snippets of code or examples. You must reference any use of Grammerly, ChatGPT, or any other generative AI tools to **improve** the quality of your own original work.
- Your report should have the following sections
  - **Background**

    In this section you should give a brief overview of what you are doing in the assignment including any useful links or references to background material and a high level description of any difficulties that you had.
  - **Processing**

    *In this section you should describe the structure and content of the data in detail, answer the questions that have been asked, and provide any more details required for discussion in the analysis section below. You should describe the steps that you took and anything interesting that you discovered along the way, but you should **not** include any actual outputs as you create additional columns on stations step by step.*
  - **Analysis**

    *In this section you should answer the questions that have been asked **and** explain your methodology. You should **not** just list answers that you have not explained. You should include any visualizations that you have been asked to generate, or include an example and provide the rest in your supplementary material. You should only include snippets of code that you want to explain in detail, all other code should be in your supplementary material.*
  - **Conclusions**

    In this section you should give a high level summary of what you have done and any insights that you had. You should talk about any tasks that you were unable to complete and why.

DATA

In this assignment we will study some of the weather data contained in the Global Historical Climate Network (GHCN), an integrated database of climate summaries from land surface stations around the world. The data extends back over 250 years and is collected from more than 20 independent sources, each of which have been subjected to quality assurance reviews.

- Global Historical Climatology Network (GHCN)
- GHCN Daily

The daily climate summaries contain records from over 100,000 stations in 200 countries and territories around the world. There are several daily variables, including maximum and minimum temperature, total daily precipitation, snowfall, and snow depth; however, about half of the stations report precipitation only. The records vary by station and cover intervals ranging from less than a year to over 100 years in total.

The daily climate summaries are supplemented by metadata further identifying the stations, countries, states, and elements inventory specific to each station and time period. These provide human readable names, geographical coordinates, elevations, and date ranges for each station variable in the inventory.

Daily

The daily climate summaries are comma separated, where each field is separated by a comma (,) and where null fields are empty. A single row of data contains an observation for a specific station and day, and each variable collected by the station is on a separate row.

The following information defines each field in a single row of data covering one station day. Each field described below is separated by a comma (,) and follows the order below from left to right in each row.

Name	Type	Summary
ID	Character	Station code
DATE	Date	Observation date formatted as YYYYMMDD
ELEMENT	Character	Element type indicator
VALUE	Real	Data value for ELEMENT
MEASUREMENT FLAG	Character	Measurement Flag
QUALITY FLAG	Character	Quality Flag
SOURCE FLAG	Character	Source Flag
OBSERVATION TIME	Time	Observation time formatted as HHMM

The specific ELEMENT codes and their units are explained in Section III of the GHCN Daily README, along with the MEASUREMENT FLAG, QUALITY FLAG, and SOURCE FLAG. The OBSERVATION TIME field is populated using the NOAA / NCDC Multinetwork Metadata System (MMS).

Metadata

The station, country, state, and variable inventory metadata files are fixed width text formatted, where each column has a fixed width specified by a character range and where null fields are represented by whitespace instead.

Stations

The stations table contains geographical coordinates, elevation, country code, state code, station name, and columns indicating if the station is part of the GCOS Surface Network (GSN), the US Historical Climatology Network (HCN), or the US Climate Reference Network (CRN). The first two characters of the station code denote the country code (FIPS).

Name	Range	Type
ID	1 - 11	Character
LATITUDE	13 - 20	Real
LONGITUDE	22 - 30	Real
ELEVATION	32 - 37	Real
STATE	39 - 40	Character
NAME	42 - 71	Character
GSN FLAG	73 - 75	Character
HCN/CRN FLAG	77 - 79	Character
WMO ID	81 - 85	Character

Countries

The countries table contains country name only.

Name	Range	Type
CODE	1 - 2	Character
NAME	4 - 64	Character

States

The states table contains state name only.

Name	Range	Type
CODE	1 - 2	Character
NAME	4 - 50	Character

Inventory

The inventory table contains the set of elements recorded by each station, along with the time period each element was recorded. The specific ELEMENT codes and their units are explained in Section III of the GHCN Daily README.

Name	Range	Type
ID	1 - 11	Character
LATITUDE	13 - 20	Real
LONGITUDE	22 - 30	Real
ELEMENT	32 - 35	Character
FIRSTYEAR	37 - 40	Integer
LASTYEAR	42 - 45	Integer

Note that the latitude and longitude correspond to the latitude and longitude in the stations table exactly.

TASKS

The assignment tasks are separated into three sections, each of which explore the data in increasing detail. In the first section you will explore the metadata tables and part of the daily climate summaries to help develop your code interactively. In the second section you will answer specific questions about the data using the code you have developed. In the third section you will apply everything you that have learned to solve a challenge and write a report describing your method and the results you obtained.

Processing

In this section you will combine the daily climate summaries with the metadata tables, joining on station, state, and country. This will ensure that you have all of the relevant metadata at your disposal for every row of data in the daily climate summaries, and that you can sort and group the data as desired.

Q1 Define the separate data sources as `daily`, `stations`, `states`, `countries`, and `inventory` respectively. All of the data is stored in HDFS under `hdfs:///data/ghcnd` and is read only. **Do not copy any of the data to your home directory.**

Use the `hdfs` command to explore `hdfs:///data/ghcnd` without actually loading any data into memory.

(a) How is the data structured? Draw a directory tree to represent this in a sensible way.

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

(c) What is the total size of all of the data? How much of that is `daily`?

Q2 Open a notebook on the master node using `start_pyspark_notebook` and run `start_spark` with 2 executors, 1 core per executor, 1 GB of executor memory, and 1 GB of master memory. You will now explore each data source briefly to ensure that the descriptions are accurate and that the data is as expected.

(a) Define a schema for `daily` based on the description in this assignment or in the GHCN Daily README. This schema should use the data types defined in pyspark.sql.

(b) Load 1000 rows of the `hdfs:///data/ghcnd/daily/2023.csv.gz` file into Spark by using the `limit` command immediately after the `read` command.

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

(c) Load each of `stations`, `states`, `countries`, and `inventory` into Spark as well. You will need to find a way to parse the fixed width text formatting, as this format is not included in the standard `spark.read` library. You could try using `spark.read.text` and `pyspark.sql.functions.substring` or finding an existing open source library.

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

Q3 Next you will combine the daily climate summaries with the metadata tables, joining on station, state, and country. Note that joining the daily climate summaries and metadata into a single table is not efficient for a database of this size, but joining the metadata into a single table is very convenient for filtering and sorting based on attributes at a station level.

You will need to start saving some intermediate outputs along the way. Create an output directory in your home directory e.g. `hdfs:///user/abc123/outputs/ghcnd/`. Note that we only have 400GB of distributed storage available so think carefully before you write output to HDFS.

(a) Extract the two character country code from each station code in `stations` and store the output as a new column using the `withColumn` method.

(b) LEFT JOIN `stations` with `countries` using your output from part (a).

(c) LEFT JOIN `stations` and `states`, allowing for the fact that state codes are only provided for stations in the US.

(d) Based on `inventory`, what was the first and last year that each station was active and collected any element at all?

How many different elements has each station collected overall?

Further, count separately the number of core elements and the number of "other" elements that each station has collected overall.

How many stations collect all five core elements? How many collect **only** precipitation and no other elements?

Note that we could also determine the set of elements that each station has collected and store this output as a new column using `pyspark.sql.functions.collect_set` but it will be more efficient to first filter `inventory` by element type using the element column and then to join against that output as necessary.

(e) LEFT JOIN `stations` and your output from part (d).

This enriched `stations` table will be useful. Save it to your output directory. Think carefully about the file format that you use (e.g. `csv`, `csv.gz`, `parquet`) with respect to consistency and efficiency. From now on assume that `stations` refers to this enriched table with all the new columns included.

(f) LEFT JOIN your 1000 rows subset of `daily` and your output from part (e). Are there any stations in your subset of `daily` that are not in `stations` at all?

How expensive do you think it would be to LEFT JOIN all of `daily` and `stations`? Could you determine if there are any stations in `daily` that are not in `stations` without using LEFT JOIN?

Analysis

In the section you will answer specific questions about the data using the code that you have developed.

For the some of the tasks in this section you will need to process all the `daily` climate summaries, and you may want to increase the resources allocated to your notebook. You may increase your resources up to 4 executors, 2 cores per executor, 4 GB of executor memory, and 4 GB of master memory.

Don't forget that you are sharing cluster resources. Keep an eye on Spark using the web user interface mathmadslinux2p:8080, and don't leave a shell running for longer than necessary. If you are asked to share your resources, please respond quickly to let them know how much more time you need.

Q1 First it will be helpful to know more about the `stations` themselves before we study the `daily` climate summaries in more detail.

(a) How many stations are there in total? How many stations were active so far in 2024?

How many stations are in each of the GCOS Surface Network (GSN), the US Historical Climatology Network (HCN), and the US Climate Reference Network (CRN)? Are there any stations that are in more than one of these networks?

(b) Count the total number of stations in each country, and join these counts onto `countries` so that we can use these counts later if desired.

Do the same for `states` and save a copy of each table to your output directory.

(c) How many stations are there in the Southern Hemisphere?

Some of the countries in the database are territories of the United States as indicated by the name of the country. How many stations are there in total in the territories of the United States around the world, excluding the United States itself?

Q2 You can create user defined functions in Spark by taking native Python functions and wrapping them with `pyspark.sql.functions.udf` which allows you to apply a function to each row using columns as inputs. You may find this functionality useful.

(a) Write a Spark function that computes the geographical distance between two stations using their latitude and longitude as arguments. You can test this function by using CROSS JOIN on a small subset of `stations` to generate a table with two stations in each row.

Note that there is more than one way to compute geographical distance, choose a method that at least takes into account that the earth is spherical.

(b) Apply this function to compute the pairwise distances between all stations in New Zealand, and save the result to your output directory.

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

Q3 Next you will start exploring all the `daily` climate summaries in more detail. In order to know how efficiently you can load and apply transformations to `daily`, you first need to understand the structure and format of `daily`.

(a) Recall the `hdfs` commands that you used to explore the data in Processing Q1. You would have used

```
hdfs dfs -ls [path]
hdfs dfs -du [path]
```

to determine the size of files under a specific path in HDFS.

Use the following command

```
hdfs getconf -confKey "dfs.blocksize"
```

to determine the default blocksize of HDFS. How many blocks are required for the `daily` climate summaries for the year 2024? What about the year 2023? What are the individual block sizes for the year 2023? You will need to use another `hdfs` command.

(b) Load and count the number of observations in 2023 and then separately in 2024.

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

You can check this by using your application console. which you can find in the web user interface mathmadslinux2p:8080. You can either determine the number of tasks executed by each stage of each job or determine the total number of tasks completed by each executor after the job has completed.

You can also use `getNumPartitions` to determine the number of partitions in any RDD, which determines the number of tasks it would take to apply a transformation to that RDD.

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

(c) Load and count the total number of observations in the years from 2014 to 2023 (inclusive).

Note that you can use glob patterns in the path argument of the `read` command.

Now how many tasks were executed by each stage, and how does this number correspond to your input?

Explain how Spark partitions input files that are compressed.

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

Q4 The data stored in HDFS under `hdfs:///data/` has a replication factor of 8 and is distributed evenly across the worker nodes. You should always be able to load and apply transformations to multiple years of `daily` in parallel.

Again, you should only use part of the `daily` climate summaries while you are still developing your code so that you can use fewer resources to get preliminary results without waiting all day.

(a) Count the number of rows in `daily`.

Note that this will take a while if you are only using 2 executors and 1 core per executor, and that the amount of driver and executor memory should not matter unless you actually try to cache or collect all of `daily`. You should **not** try to cache or collect all of `daily`.

(b) Filter `daily` using the `filter` command to obtain the subset of observations containing the five core elements described in `inventory`.

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

Which element has the most observations?

(c) Many stations collect TMIN and TMAX, but do not necessarily report them simultaneously due to issues with data collection or coverage. Determine how many observations of TMIN do not have a corresponding observation of TMAX.

How many unique stations contributed to these observations?

(d) Filter `daily` to obtain all observations of TMIN and TMAX for all stations in New Zealand, and save the result to your output directory.

How many observations are there, and how many years are covered by the observations?

Use `hdfs dfs -copyToLocal` to copy the output from HDFS to your local home directory, and count the number of rows in the part files using the `wc -l` bash command. This should match the number of observations that you counted using Spark.

Plot time series for TMIN and TMAX together for each station in New Zealand using Python, R, or any other programming language or data visualization tool that you know well. Also, plot the average time series for TMIN and TMAX together for the entire country.

(e) Group the precipitation observations by year and country. Compute the **average** rainfall in each year for each country, and save this result to your output directory.

Which country has the highest average rainfall in a single year across the entire dataset? Is this result sensible? Is this result consistent with your previous analysis?

Find an elegant way to plot the average rainfall in 2023 for each country. There are many ways that you could do this, such as using a choropleth to color a map based on average rainfall. Are there any gaps or missing values in your plot?