DATA420-25S2 (C)

Assignment 1

GHCN Data Analysis using Spark

Due on Friday, September 12, 2025 by 5:00 PM.

If you want to discuss the assignment material you can use the Discord server where the discussion will benefit all. If you have a question that requires an official answer you can use the forum on LEARN. If you have a more personal question you can email me or contact the class rep as needed.

A reminder that the Discord server is for discussion of concepts only, not for sharing code or answers to assignment questions.

Links

Report upload (pdf)
Supplementary material upload (zip, limited to 10 MB)

Discord server
Help forum for Assignment 1

Instructions

- Any assignments submitted after the deadline without obtaining an extension will receive a 20% penalty. You should aim to work on the assignment incrementally in your own time and in the labs each week.

- The forum is a great place to ask questions about the assignment! Any questions will be answered by a facilitator as soon as possible but students can also answer each other's questions so that you can all benefit from the answers.

- The data provided for the assignment is stored in Azure Blob Storage and outputs that you generate will be stored in Azure Blob Storage as well. Hadoop and Spark can both interact with Azure Blob Storage similar to how they interact with HDFS, but where the replication and distribution is handled by Azure instead. This makes it possible to read or write data in Azure over HTTPS where the path is prefixed by `wasbs://`. Follow the instructions in the notebook provided to load data from the data container and to save outputs that you generate to a separate user container where the path is prefixed by your username.

DATA

In this assignment we will study the weather data in the Global Historical Climatology Network (GHCN), a 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 passed quality assurance reviews.

- Global Historical Climatology Network (GHCN)

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 tables

The station, country, state, and variable inventory metadata tables 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).

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.

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

TASKS

The assignment is separated into a number of sections, each of which explore the data in increasing detail from processing to analysis to visualizations. You will need to plan your time carefully to complete the assignment on time..

These supplement the report requirements detailed on the cover page. In general, your write up should be accurate and concise and should demonstrate depth of understanding. The tasks are intentionally detailed to make it easier to work through the assignment step by step.

Processing

In this section you will explore datasets at a high level.

Q1 First you will investigate the `daily`, `stations`, `states`, `countries`, and `inventory` data provided in cloud storage in `wasbs://campus-data@madsstorage002.blob.core.windows.net/ghcnd/` using the `hdfs` command.

Follow the instructions in the notebook provided to explore each dataset using the `hdfs` command without loading any data into memory and answer the following questions:

(a) How is the data structured? Are any of the datasets compressed?

(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, and how much of that is `daily`?

Q2 You will now load each dataset to ensure the descriptions are accurate and that you can apply the schema either as the data is loaded or by casting columns as they are extracted by manually processing the text records.

Extend the code example in the notebook provided by following the steps below.

(a) Define a schema for `daily` based on the description above or in the GHCN Daily README, using the types defined in pyspark.sql. What do you think is the best way to load the DATE and OBSERVATION TIME columns?

(b) Modify the `spark.read.csv` command to load a subset of the most recent year of `daily` into Spark so that it uses the schema that you defined in step (a). Did anything go wrong when you tried to use the schema? What data types did you end up using and why?

(c) Load each of `stations`, `states`, `countries`, and `inventory` datasets into Spark and find a way to extract the columns and data types in the descriptions above. You will need to parse the fixed width text formatting by hand, as there is no method to load this format implemented in the standard `spark.read` library. You should use `pyspark.sql.functions.substring` to extract the columns based on their character range.

(d) How many rows are there in each of the metadata tables?

(e) How many rows are there 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`.

Q3 Next you will combine relevant information from the metadata tables by joining on station, state, and country to give an enriched `stations` table that we can use for filtering based on attributes at a station level.

(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 step (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 your output from step (c) and your output from step (d).

This enriched table will be useful. Save it to the user container prefixed by your username, `wasbs://campus-user@madsstorage002.blob.core.windows.net/abc123/`. You should think carefully about the file format e.g. `csv`, `csv.gz`, or `parquet` with respect to efficiency.

Q4 Next you will check for any missing stations in `daily`.

(a) LEFT JOIN a subset of `daily` and your `stations` table from Q3 step (e).

How expensive do you think it would be to join all of `daily` and `stations`? Can you think of an efficient way to check if there are any stations in `stations` that are not in `daily` at all without using LEFT JOIN?

(b) Based on step (a) count the total number of stations in `stations` that are not in `daily`.

You may need to stop and restart your Spark application to increase the resources you have allocated. You may increase your resources up to 4 executors, 2 cores per executor, 4 GB of executor memory, and 4 GB of master memory.

Analysis

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

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 2025?

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) 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?

(c) 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.

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 closest together in New Zealand?

Q3 Next we will study the `daily` climate summaries in more detail.

(a) Filter `daily` using the `where` command to obtain the subset of observations containing only 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?

(b) 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 TMAX do not have a corresponding observation of TMIN.

How many unique stations contributed to these observations?

Visualizations

In the section you will develop time series and geospatial visualizations of the `daily` climate summaries which will require you to collect or copy some of your outputs to the master node in order to visualize them together. **This should be fine provided you collect or copy outputs that have been filtered or aggregated at a reasonably coarse level.** Please be careful and do not accidentally collect or copy `daily` into memory or save a copy of `daily` back to cloud storage as the costs add up quickly.

Q1 First you will plot observations of TMIN and TMAX for stations in New Zealand.

(a) 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? How many years are covered by the observations?

Are there any gaps in the data where no observations were reported?

(b) Plot the time series for TMIN and TMAX for each station in New Zealand. You should use subplots to create separate axes for each station in a single figure and you should aggregate the observations to smooth the time series to an appropriate level of detail.

You should consider the following

- How to smooth the time series to an appropriate level of detail
- How to handle gaps in the data where no observations were reported
- How to line up the time series so they can be compared consistently

(c) Plot the average time series for TMIN and TMAX for the entire country in a separate figure, changing your approach as needed to be appropriate to the size of the figure which should be larger than the subplots above.

Q2 Next you will plot precipitation observations for stations around the world, grouping by year and by country to provide a sensible temporal and spatial resolution for our visualizations. You should look up sensible principles for visualization before continuing.

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

Generate descriptive statistics for the average rainfall.

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

Is this result sensible?

(b) Plot the average rainfall in 2024 for each country using a choropleth map. You should use an open source library such as plotly, geopandas, or folium for the visualization, and you should consider how the countries in this dataset line up with countries in the library.

Are there any countries that **were** matched where no observations were reported?

You should consider the following

- Are there any outliers that might be skewing your results
- How the countries in this dataset match with countries in the open source library
- How to handle countries where no observations were reported
- What map projection is the most suitable
- What color scale is the most suitable for the type of observation being visualized