DATA420-18S2 (C)

Assignment 1

GHCN Data Analysis using Spark

Due on Friday, September 14 by 5:00 PM, to be submitted as a zip file using LEARN.

If you have any questions please email me (james.williams@canterbury.ac.nz).

LEARN

https://learn.canterbury.ac.nz/mod/assign/view.php?id=706059

Instructions

- You are encouraged to work together to understand and solve each of the tasks, but you must submit your own work.
- You have the full resources of the internet at your disposal.
- All write up, code, and outputs should be submitted in a single zip file via LEARN.
- Any assignments submitted after the deadline will receive a 50% penalty.
- All data under `hdfs:///data` is read only. Please use your own home directory to store your outputs (e.g. `hdfs:///user/abc123/outputs`).
- Please be mindful that you are sharing cluster resources. Keep an eye on Spark using the web user interface (madslinux01:8080), and don't leave a shell running for longer than necessary. Feel free to change the number of executors, cores, and memory that 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 and their shell has been running for more than 24 hours, you can kill their shell by going to the web user interface and selecting **kill**.

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 covers the last 175 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 180 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 175 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 - 50	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 without waiting all day. 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 have learned so far to solve a challenge and write a brief report describing your method and the results that 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 Start pyspark shell 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 schemas for each of `daily`, `stations`, `states`, `countries`, and `inventory` based on the descriptions in this assignment and the GHCN Daily README. These should use the data types defined in `pyspark.sql`.

(b) Load 1000 rows of `hdfs:///data/ghcnd/daily/2017.csv.gz` into Spark using the `limit` command immediately after the `read` command.

Was the description of the data accurate? Was there anything unexpected?

(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.format('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 storage available in total, 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` command.

(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 only collection precipitation?

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 of the daily climate summaries, and you may want to increase the resources available to your Spark shell. 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 (madslinux01:8080), 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 in our database, before we study the actual daily climate summaries in more detail.

(a) How many stations are there in total? How many stations have been active in 2017?

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 store the output in `countries` using the `withColumnRenamed` command.

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 only?

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?

Q2 You can create user defined functions in Spark by taking native Python functions and wrapping them using `pyspark.sql.functions.udf` (which can take multiple columns as inputs). You will find this functionality extremely 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 the geographically closest in New Zealand?

Q3 Next you will start exploring all of the daily climate summaries in more detail. In order to know how efficiently you can load and apply transformations to `daily`, you need to first understand the level of parallelism that you can achieve for the specific 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 2017? What about the year 2010? What are the individual block sizes for the year 2010? You will need to use another `hdfs` command.

Based on these results, is it possible for Spark to load and apply transformations in parallel for the year 2017? What about the year 2010?

(b) Load and count the number of rows in `daily` for each of the years 2010 and 2017.

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 (madslinux01:8080). 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.

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

(c) Load and count the number of rows in `daily` from 2010 to 2015.

Note that you can use any regular expressions 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?

Find out how Spark partitions input files that are compressed.

(d) Based on parts (b) and (c), what level of parallelism can you achieve when loading and applying transformations to `daily`? Can you think of any way you could increase this level of parallelism either in Spark or by additional preprocessing?

Q4 All of the data stored in HDFS under `hdfs:///data/ghcnd` has a replication factor of 4 and is available locally on every node in our cluster. As such you will always be able to load and apply transformations to multiple years of `daily` in parallel.

Again, you may want to use only 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 never 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? Is this element consistent with Processing Q3?

(c) Many stations collect TMAX and TMIN, 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 different stations contributed to these observations? Do any belong to the GSN, HCN, or CRN?

(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 the time series of TMIN and TMAX on the same axis for each station in New Zealand using Python, R, or any other programming language you know well. Also, plot the average time series 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?

Find an elegant way to plot the cumulative rainfall for each country using Python, R, or any other programming language you know well. There are many ways to do this in Python and R specifically, such as using a choropleth to color a map according to average rainfall.

Challenges

These challenges are deliberately open ended, and you should answer them using the level of detail and complexity that you think is appropriate (which can be based on your level of interest as well). You may use anything at your disposal to solve these challenges, but remember that you are on a shared server and that there are limited resource overall.

Q1 Explore the `daily` data in more detail.

We have not considered any of the other elements that each station has collected, or the quality of the specific observations that each station has collected. Investigate the coverage of the other elements, both temporally and geographically.

Did you learn anything more about the quality flag or the core elements?

Q2 Investigate the quality flag in `daily`.

How many rows in `daily` failed each of the quality assurance checks? What is your overall assessment of quality of the dataset? Would you be confident making decisions that are based on the output of a statistical model that is trained on the dataset?