Assignment05: H-Homework Streaming Large Text File

Due Saturday by 11:59pm | **Points** 20 | **Submitting** a file upload | **File Types** pdf, html, and txt

Do the exercise on the bottom of Homework Streaming Large Text File

---

- Calculate summary statistics from a data stream
- Use pipelines to process a file larger than memory

Announcements

- The question "How much longer does X take than Y" is usually most useful when answered in relative terms. For example, if the transfer took 1.9 seconds, and now it takes 79.2 seconds, then say that the second way is 79.2/1.9 = 42 times slower than the first.
- Check for comments / feedback on skills assignments.
- will post skills assignment solutions next week.

123 GO - what was the last kind of physical exercise you did?

Background

The GDELT Project is the Global Database of Events, Language, and Tone. It describes itself as:

> A Global Database of Society. Supported by Google Jigsaw, the GDELT Project monitors the world's broadcast, print, and web news from nearly every corner of every country in over 100 languages and identifies the people, locations, organizations, themes, sources, emotions, counts, quotes, images and events driving our global society every second of every day, creating a free open platform for computing on the entire world.

The events from 2018 are available in a single file at the S3 URI `s3://stat196k-data-examples/2018.csv.gz`. This file is 3.8 GB, compressed. Here are the column definitions.

Goldstein Score

> Each CAMEO event code is assigned a numeric score from -10 to +10, capturing the theoretical potential impact that type of event will have on the stability of a country. This is known as the Goldstein Scale. This field specifies the Goldstein score for each event type. **NOTE:** this score is based on the type of event, not the specifics of the actual event record being recorded, thus two riots, one with 10 people and one with 10,000, will both receive the same Goldstein score. This can be aggregated to various levels of time resolution to yield an approximation of the stability of a location over time.

I believe this is the CAMEO event code mapping to Goldstein scores.

Assignment

Turn in two files:

- A PDF or HTML document containing your answers to the following questions in a neatly organized report.

- A file with extension `.sh.txt` showing all the code necessary to reproduce your work. The `.sh` is for shell script, and this is normally the only extension you need. The `.txt` allows Canvas to render it as plain text in the web browser, so I can grade it.

I suggest you use markdown through something like pandoc, Rmarkdown, or Jupyter notebooks to create your report. The markdown source for the assignment is on Github, so you can copy and paste from there. MS Word and other GUI programs should work fine too.

Questions

1 - Subset

5 pts

Download a small subset of the data (100 rows is plenty) to your personal computer, and examine it using any software you like. Briefly describe this subset of the data by picking out a couple rows that look interesting to you.

1.1 How many columns are there?

Based on the column definitions there are 61 columns; in addition, you can run:

```
> **On ec2 instance:** $ ssh -i ~/.ssh/id_rsa ec2-user@ec2-100-27-27-57.compute
>
> > $ aws s3 cp s3://stat196k-data-examples/2018.csv.gz - --no-sign-request | g
> >
> > $ cat -T first100Rows.csv | head -1 | grep -o "\^I" | wc -l
```

and see that it outputs `60`. Ignoring the last column all tableIds are followed by a delimiter separating each data type to their corresponding column/tableIds; as a result, the number of delimiters counted in any given table or csv/tsv file is **n-1** of the number of columns listed in the table.

UNIX Commands

- **`cat`** - Concatenate files and print on the standard output
  - `-T, --show-tabs` - Display TAB characters as ^I
- **`grep`** - Searches for PATTERNS in each FILE. PATTERNS is one or more patterns separated by newline characters, and grep prints each line that matches a pattern. Typically PATTERNS should be quoted when grep is used in a shell command.A FILE of "-" stands for standard input. If no FILE is given, recursive searches examine the working directory, and non-recursive searches read standard input
  - `-o, --only-matching` - Print only the matched (non-empty) parts of a matching line, with each such part on a separate output line
- **`gunzip`** - Takes a list of files on its command line and replaces each file whose name ends with .gz, -gz, .z, -z, or _z (ignoring case) and which begins with the correct magic number with an uncompressed file without the original extension.
- **`head`** - Output the first part of files

    - with the leading `-`, print all but the last NUM lines of each file
        - As used above it outputs the first 100 lines of the file and the second time just the first line

- **`wc`** - Print newline, word, and byte counts for each file
    - **`-l, --lines`** - Print the newline counts

1.2 Do the data values in each column seem to match the column definitions?

Yes if you don't have the proper formatting/alignment; for, not all columns need to have a value and are left blank (tabbed over). So when read not aligned it may seem that a value is assigned to the second column, but is actually meant for the sixth column because values in columns three through five are left blank.

1.3 What character delimits the records?

Escape character for tab `\t`
OR

> cat -T first100Rows.csv

will output `^I` as representation of the tab delimiter. For example:

```
719024869 ^I20170101 ^I201701 ^I2017 ^I2017.0027 ^IAGR ^IFARMER ^I ^I ^I ^I
```

1.4 What is the CAMEO event code, what event does this correspond to, and what is the Goldstein score?

The CAMEO event code is a there level hierarchal codification that articulates the action that Actor1 performed upon Actor2 in events around the world. Form left to right, the first number (can be zero leading: 01 vs 12) sets the Superordinate (**ex. 02: APPEAL**), appended to that sets the Basic level (**ex. 025: Appeal to yield**), and depending on the situation the last number to be appended is the Subordinate level (**ex. 0251: Appeal for easing of administrative sanctions**). In use these raw CAMEO action code describes the action that Actor1 performed upon Actor2; for example, *Human Rights Watch also called on Yemen, Algeria and Malaysia to immediately lift bans on newspapers closed in recent days for printing the caricatures*.

The Goldstein score is based on CAMEO event code each is assigned a numeric score from -10 to +10, where a positive number denotes that the region is doing well/contributed to good versus a negative number denotes that the region isn't doing well/is unstable, capturing the theoretical potential impact that type of event will have on the stability of a country; for example, a Goldstein score of -7.6 as a result of Armed force mobilization **VS** a Goldstein score of 7.4 as a result of Extend economic aid.

1.5 Are the URL's to the news articles still live, and do they match the CAMEO event code?

Most are still live, but others like the article to After Trump criticism China denies selling oil illicitly to North Korea; nevertheless, given the short descriptions of the articles they do seem to match the CAMEO event code and fits the Goldstein score as well.

1.6 Does the Goldstein score appear to be doing what it was designed to do?

Yes although relatively arbitrary in what sets a type of event to be assigned a lower score than another; for example, a "Non-military destruction/injury" is given a Goldstein score of -8.7 a versus a "Non-injury destructive action" is given a Goldstein score of -8.3 are both terrible events it should be kept in mind for the general public that the "score is based on the type of event, not the specifics of the actual event record being recorded – thus two riots, one with 10 people and one with 10,000, will both receive the same Goldstein score. This can be aggregated to various levels of time resolution to yield an approximation of the stability of a location over time"

2 - Histogram

10 pts

Create a histogram of the Goldstein scores for all of 2018, using the integers as bin endpoints for the histogram. It's possible to do this in less than 10 minutes using a single shell pipeline on a t2 micro instance with 1 vCPU, 1 GiB memory, and 8 GiB storage.

2.1 How long does your program take to run?

It took 6 minutes 58 seconds.

```
> **On ec2 instance:** $ ssh -i ~/.ssh/id_rsa ec2-user@PublicDNS-hostname.com
>
> > $ time aws s3 cp s3://stat196k-data-examples/2018.csv.gz - --no-sign-reques

   Outs all values as int (no decimals/fractional values)
   13 -10
   1 -9
   1 -8
   .
   .
   .
   real    6m58.523s
   user    6m39.716s
   sys     0m18.233s
```

2.2 Explain in detail what each command in the pipeline does and how they work together

This command measures the time it takes to copy the 2018.csv.gz from the s3 bucket, unzip its contents, pulls all values in the 31st column of the table, sorts it in ascending order, counts the repeated values, and outputs the results into file named "allGoldsteinScoresCounted.txt"

```
> **On ec2 instance:** $ ssh -i ~/.ssh/id_rsa ec2-user@PublicDNS-hostname.com
>
> > $ time aws s3 cp s3://stat196k-data-examples/2018.csv.gz - --no-sign-reques
```

1. `time` - Run programs and summarize system resource usage
2. aws - The AWS Command Line Interface is a unified tool to manage your AWS services
3. s3 - An Amazon S3 bucket is a public cloud storage resource available in Amazon Web Services' (AWS) Simple Storage Service (S3), an object storage offering. Amazon S3 buckets,

which are similar to file folders, store objects, which consist of data and its descriptive metadata.

4. `cp` - Copies a local file or S3 object to another location locally or in S3.
    1. Using the s3 protocol go to s3://stat196k-data-examples
5. `--no-sign-request` - (boolean) Do not sign requests. Credentials will not be loaded if this argument is provided.
6. `gunzip` - Takes a list of files on its command line and replaces each file whose name ends with .gz, -gz, .z, -z, or _z (ignoring case) and which begins with the correct magic number with an uncompressed file without the original extension.
7. `cut` - Remove sections from each line of files
    1. `-f 31` - Remove the 31st section from each line of files
8. `sort` - Sort lines of text files
    1. `-n` - Compare according to string numerical value
9. `uniq` - Report or omit repeated lines
    1. `-c` - Prefix lines by the number of occurrences
10. `>` - Redirects output to a file, overwriting the file.
    1. Writes output into file named allGoldsteinScoresCounted.txt

---

This command measures the time it takes to copy the 2018.csv.gz from the s3 bucket, unzip its contents, pulls all values in the 31st column of the table, converts all numbers into integer (non-decimal/fractional values), sorts it in ascending order, counts the repeated values, and outputs the results into file named "allIntegerGoldsteinScoresCounted.txt"

```
> **On ec2 instance:** $ ssh -i ~/.ssh/id_rsa ec2-user@PublicDNS-hostname.com
>
> > $ time aws s3 cp s3://stat196k-data-examples/2018.csv.gz - --no-sign-reques
```

1. `time` - Run programs and summarize system resource usage
2. `aws` - The AWS Command Line Interface is a unified tool to manage your AWS services
3. `s3` - An Amazon S3 bucket is a public cloud storage resource available in Amazon Web Services' (AWS) Simple Storage Service (S3), an object storage offering. Amazon S3 buckets, which are similar to file folders, store objects, which consist of data and its descriptive metadata.
4. `cp` - Copies a local file or S3 object to another location locally or in S3.
    1. Using the s3 protocol go to s3://stat196k-data-examples
5. `--no-sign-request` - (boolean) Do not sign requests. Credentials will not be loaded if this argument is provided.
6. `gunzip` - Takes a list of files on its command line and replaces each file whose name ends with .gz, -gz, .z, -z, or _z (ignoring case) and which begins with the correct magic number with an uncompressed file without the original extension.
7. `cut` - Remove sections from each line of files

    1. `-f 31` - Remove the 31st section from each line of files

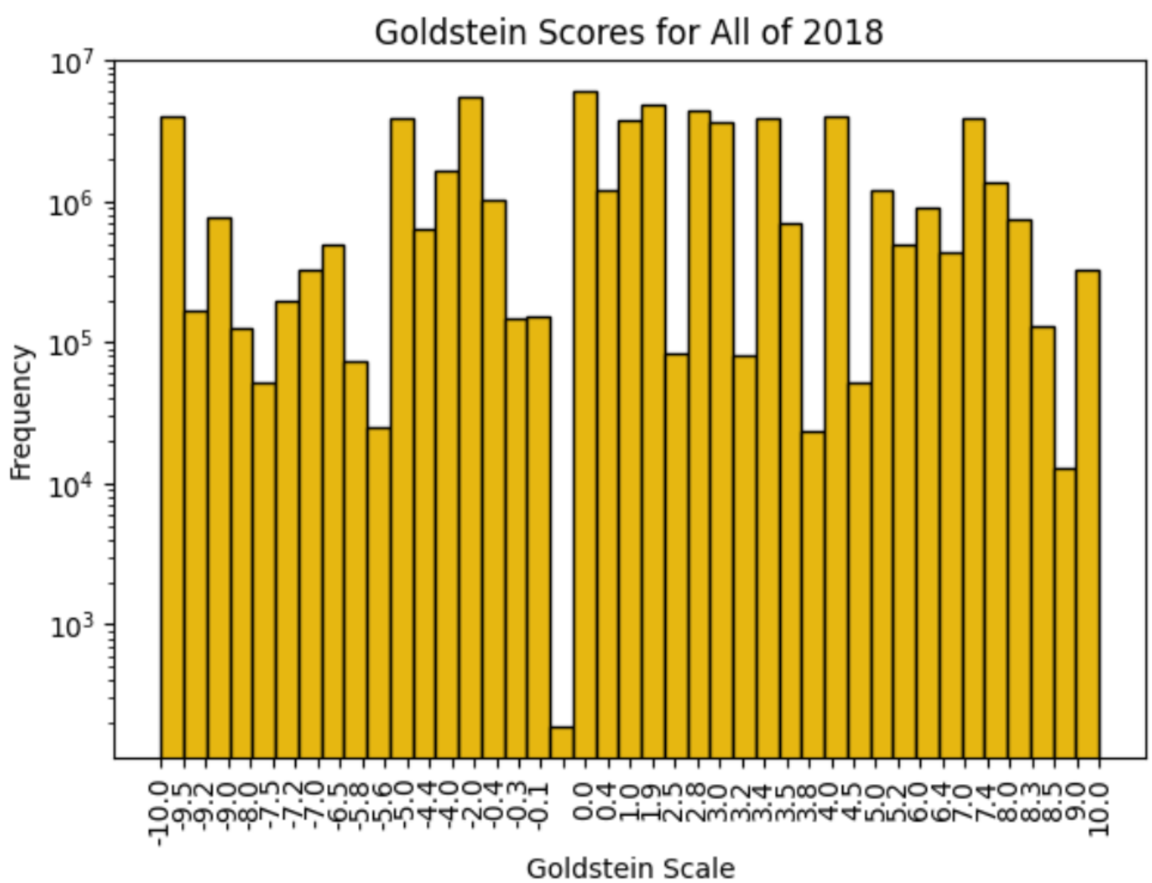
8. `awk '{print int($1)}'` - For every line print the contents of the first field
9. `sort` - Sort lines of text files
    1. `-n` - Compare according to string numerical value

10. `uniq` - Report or omit repeated lines
    1. `-c` - Prefix lines by the number of occurrences

11. > - Redirects output to a file, overwriting the file.
    1. Writes output into file named allIntegerGoldsteinScoresCounted.txt

2.3 Plot and interpret the histogram. You'll probably want to download the summary statistics (around 20 numbers) to your personal computer to plot the histogram. Do you notice anything strange?

Yes there are 191 entries without an assigned Goldstien Score.



note: the frquency with a blank interval is a result that they were not assigned a goldstein score.

```
> **On local machine**
>
> > $ scp -i ~/.ssh/id_rsa ec2-user@ec2-100-27-27-57.compute-1.amazonaws.com:~/
> >
> > $ cat data/allGoldsteinScoresCounted.txt | head -20 > data/firstTwentyGolds
```

2.4 Exactly how many events (rows) are in this data?

There are 43 events (rows) in this data.

```
> $ cat -n data/allGoldsteinScoresCounted.txt
```

3 - Performance

5 pts

Print and interpret the output of `top` while your program is running.

```
top - 12:33:25 up 1 day, 15:45,  2 users,  load average: 1.64, 1.28, 0.67
Tasks:  97 total,   3 running,  58 sleeping,   0 stopped,   0 zombie
%Cpu(s): 95.3 us,  3.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  1.7 si,  0.0 st
KiB Mem :  1006900 total,    70880 free,   489592 used,   446428 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   378864 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
21170 ec2-user  20   0    4628    832    768 R 41.5  0.1   1:43.99 gzip
21172 ec2-user  20   0  125044   8316   2032 S 29.9  0.8   1:18.25 sort
21171 ec2-user  20   0  116524    816    752 S 23.3  0.1   0:57.17 cut
21174 ec2-user  20   0 1067208 366252  10660 S  4.3 36.4   0:14.62 aws
14432 ec2-user  20   0  154008   3672    908 S  0.3  0.4   0:04.33 sshd
```

```
> Using pipeline command
>
> > $ time aws s3 cp s3://stat196k-data-examples/2018.csv.gz - --no-sign-reques
```

Performance Validation

In testing I will be adhearing to the following:

- **Standard Deviation (SD, s):** amount of variability in a data set.
  - How tightly packed around mean are the data?
- **Standard Error (SE, sM):** how far is the sampled average likely to be from the true average.
  - SD normalized to number of samples.
- **Relative Standard Error (RSE):** Standard error as a percentage of the mean.
  - SE normalized across multiple types of measurements.

test	coldRun	run01	run02	run03	AVG	SD	SE	RSE
test01	22	23	20	22	21.67	1.528	0.882	4.07%
test02	52	53	50	52	51.67	1.528	0.882	1.71%
test03	50	55	49	47	50.33	4.163	2.403	4.78%

test01 and **test02** have the same SD and SE because the datapoints are equally distributed around the AVG and they have the same number of runs (3).

test02 has a lower RSE than **test01** because a SE of 0.882 is less significant to an AVG of 51.67 than it is to an AVG of 21.67.

test03 has a higher RSE than **test02** because **test03**’s SE is significantly higher than **test02**’s while their AVG is about the same.

test	coldRun	run01	run02	run03	run04	run05	AVG	SD	SE	RSE
test04	50	55	49	47	54	47	50.4	3.847	1.720	3.41%

test04 has almost identical AVG as **test03**, however SD is marginally lower and SE is significantly lower. Because **test04** has 5 warm runs instead of 3, the SE is lower even with a similar SD. The RSE drops significantly as runs increase.

Performance Validation Conclusion

RSE is a useful statistic for analyzing the variance present in benchmarking datasets.

Proposal: ensure data falls within 5% RSE threshold unless there is a technically valid explanation for the variance.

- If data does not meet this RSE threshold, increase the number of test runs (in increments of 2) until it does.

3.1 What are the bottlenecks?

At a glance the CPU load roughly evenly distributed; however, we see that `gzip` uses almost half of the CPU power, so our bottle neck is `gzip`.

3.2 Run and time your program on an EC2 instance with more vCPU's and a faster network and show the results of `top` once more. Is the program faster on the more expensive instance?

No; however, during my test I found that **t2.nano** runs faster and cheaper than our default **t2.micro** instance.

price	0.0116	0.0058	
0.0058	t2.micro	t2.nano	delta Δ
	407	23	1769.57
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	1669.57	% faster	
Multiplier (old/new)	17.70	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	94.35	% less time	

price	0.0116	0.096	
-0.0844	t2.micro	m5.large	delta Δ
	407	296	137.50
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	37.50	% faster	
Multiplier (old/new)	1.38	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	27.27	% less time	

price	0.0116	0.192	
-0.1804	t2.micro	m5.xlarge	delta Δ
	407	260	156.54
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	56.54	% faster	
Multiplier (old/new)	1.57	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	36.12	% less time	

price	0.0116	0.023	
-0.0114	t2.micro	t2.small	delta Δ
	407	408	99.75
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	-0.25	% faster	
Multiplier (old/new)	1.00	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	-0.25	% less time	

price	0.0116	0.0928	
-0.0812	t2.micro	t2.large	delta Δ
	407	292	139.38
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	39.38	% faster	
Multiplier (old/new)	1.39	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	28.26	% less time	

price	0.0116	0.1856	
-0.174	t2.micro	t2.xlarge	delta Δ
	407	277	146.93
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	46.93	% faster	
Multiplier (old/new)	1.47	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	31.94	% less time	

price	0.0116	0.023	
-0.0114	t2.micro	c4.large	delta Δ
	407	299	136.12
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	36.12	% faster	
Multiplier (old/new)	1.36	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	26.54	% less time	

price	0.0116	0.0464	
-0.0348	t2.micro	t2.medium	delta Δ
	407	295	137.97
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	37.97	% faster	
Multiplier (old/new)	1.38	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	27.52	% less time	

price	0.0116	0.199	
-0.1874	t2.micro	c4.xlarge	delta Δ
	407	250	162.80
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	62.80	% faster	
Multiplier (old/new)	1.63	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	38.57	% less time	

price	0.0116	0.085	
-0.0734	t2.micro	c5.large	delta Δ
	407	271	150.18
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	50.18	% faster	
Multiplier (old/new)	1.50	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	33.42	% less time	

price	0.0116	0.17	
-0.1584	t2.micro	c5.xlarge	delta Δ
	407	225	180.89
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	80.89	% faster	
Multiplier (old/new)	1.81	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	44.72	% less time	

price	0.0116	0.384	
#VALUE!	t2.micro	m5.2xlarge	delta Δ
	407	238	171.01
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	71.01	% faster	
Multiplier (old/new)	1.71	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	41.52	% less time	

price	0.0116	0.10	
-0.0884	t2.micro	m4.large	delta Δ
	407	355	114.65
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	14.65	% faster	
Multiplier (old/new)	1.15	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	12.78	% less time	

price	0.0116	0.20	
-0.1884	t2.micro	m4.xlarge	delta Δ
	407	293	138.91
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	38.91	% faster	
Multiplier (old/new)	1.39	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	28.01	% less time	

price	0.0116	0.3712	
-0.3596	t2.micro	t2.2xlarge	delta Δ
	407	296	137.50
Speed comparison: NEW is … as/than OLD			
Percentage (((old/new)*100)-100)	37.50	% faster	
Multiplier (old/new)	1.38	X as fast	
Time comparison: NEW completes task in … as/than OLD			
Percentage (100-((new/old)*100))	27.27	% less time	

3.3 Are you benefitting from pipeline parallelism?

Generally yes; howver, we do have outliers that perform super duper amazing (t2.nano instance) and a disappointing results produced from the (t2.small instance). Beyond that a majority of the instance have performed around and/or around 50 percent.

3.4 What's the bottleneck now?

Still `gzip` and then `uniq`, but only because it's the only program to run last in my pipe.

3.5 Compare and comment on the financial cost of using a more expensive instance versus the t2.micro

Is it worth it? Yes only for the t2.nano instance.

Remember to terminate these more expensive machines immediately after you use them!

Otherwise, you may quickly run through your $50 credit and have to spend your own money. AWS Services Supported says that our Educate accounts can only use these kinds of instances: "t2.small", "t2.micro", "t2.nano", "m4.large", "c4.large", "c5.large", "m5.large", "t2.medium", "m4.xlarge", "c4.xlarge", "c5.xlarge", "t2.2xlarge", "m5.2xlarge", "t2.large", "t2.xlarge", "m5.xlarge".

4 - Extra Credit Challenge

0 pts, optional

Starting with the same 3.8 GB file on S3, calculate the summary statistics necessary for the histogram as fast as possible. You can use the shell or any other programming language together with any EC2 instance available through your AWS Educate account. Hint: look into software like GNU parallel and pigz. Turn in any extra code you write. The student with the fastest program gets a minimal amount of extra credit and a maximal amount of glory.

Resources

Blog post streaming with S3

- 1 Subset

  Download a small subset of the data (100 rows is plenty) to your personal computer, and examine it using any software you like. Briefly describe this subset of the data by picking out a couple rows that look interesting to you.

  - 1.1 How many columns are there?
    - How to count number of tabs in each line using shell script?
    - How to count number of columns in CSV file using bash shell
  - 1.2 Do the data values in each column seem to match the column definitions?
  - 1.3 What character delimits the records?
  - 1.4 What is the CAMEO event code, what event does this correspond to, and what is the Goldstein score?
    - GDELT-Data_Format_Codebook.pdf
    - THE GDELT EVENT DATABASE DATA FORMAT CODEBOOK V2.0
    - Goldstein Scale for WEIS Data
  - 1.5 Are the URL's to the news articles still live, and do they match the CAMEO event code?
  - 1.6 Does the Goldstein score appear to be doing what it was designed to do?
- 2 Histogram

  Create a histogram of the Goldstein scores for all of 2018, using the integers as bin endpoints for the histogram. It's possible to do this in less than 10 minutes using a single shell pipeline on a t2 micro instance with 1 vCPU, 1 GiB memory, and 8 GiB storage.

  - 2.1 How long does your program take to run?
    - Pipes as input/output files
    - VIDEO - Cambridge Missing Semester Lecture 4: Data Wrangling (2020)
    - Data Wrangling Lecture Notes

    - 2.2 Explain in detail what each command in the pipeline does and how they work together
        - aws options
    - 2.3 Plot and interpret the histogram. You'll probably want to download the summary statistics (around 20 numbers) to your personal computer to plot the histogram. Do you notice anything strange?
        - How to Determine Histogram Bin Width and Bin Intervals
        - How to plot a histogram using Matplotlib in Python with a list of data?
        - SCP first line of a file to another system
        - How to copy a file from a remote server to a local machine?
        - How do you read from stdin?
        - Difference between input() and sys.stdin.readline()
        - Python | Pandas DataFrame
        - Python Histogram Plotting: NumPy, Matplotlib, Pandas & Seaborn
        - ~~Plotting in Golang – Histogram, BarPlot, BoxPlot~~
    - 2.4 Exactly how many events (rows) are in this data?
- 3 Performance
    - 3.1 What are the bottlenecks?
        - Tmux Cheat Sheet & Quick Reference
    - 3.2 Run and time your program on an EC2 instance with more vCPU's and a faster network and show the results of `top` once more. Is the program faster on the more expensive instance?
    - 3.3 Are you benefitting from pipeline parallelism?
    - 3.4 What's the bottleneck now?
    - 3.5 Compare and comment on the financial cost of using a more expensive instance versus the t2.micro.