
UC Love Data Week - https://uc-love-data-week.github.io/

UC Love Data Workshop - Tips and Tools for Dealing with Large Datasets

Presenters:
Leigh Phan - Data Scientist - UCLA Data Science Center
Kristian Allen - Software Developer - UCLA Data Science Center
----

Stock tools to help with file introspection

Link to Zoom:
https://uclahs.zoom.us/w/99517728610?tk=3jHKqrLzoPVykqVVWxT6QWhFT2VjAXTVTA_uu5Iq8Vk.DQYAAAAXK7gHYhZnd1pDQ1RsNVIxQ0NySXhlZVNwQml3AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Link to slides:
https://docs.google.com/presentation/d/1nwA4uZpo74DoSazPSJGoEmJ26nZ8VHndpJzh3gaTJmg/edit#slide=id.g64ea3792a7_0_0

Link to full dataset:
https://drive.google.com/file/d/1oypkP37-2sy5ljrbUfwtOgPuVUMqIe8d/view?usp=sharing (3.6GB)

Link to subset of dataset (use this for demo, first 10k rows):
https://drive.google.com/file/d/1X1KDlYiaJo_Sdppr8nJSgz3YDPomTxdV/view?usp=sharing

Tools are available in the terminal:
- Linux terminal
- **MacOS terminal**

- (Steps for opening the terminal: search for terminal in finder or select from Applications->Utilities->Terminal)
- **Check that X-Code is installed and up-to-date**
    - xcode-select --install
- **Other options for MacOS:**
    - https://brew.sh/

- **Windows** - Some twists....

- Recommended: For this workshop... gitbash is the quickest... but is not exactly the same as the other 2 above
- - https://git-scm.com/downloads
- Longer term solution, install Linux subsystem
- - https://learn.microsoft.com/en-us/windows/wsl/install

File Format Examples:
* CSV: comma-separated values
* JSON: JavaScript Object Notation
* XML: Extensible Markup Language

Lets take a look at what we have...

exploring our data in the command line
Navigate to the directory containing the data

ls = List
gr

```
> ls
parking_snippet.csv
```

```
> ls -l
total 7126752
-rw-r--r--  1 leigh  staff    1489661 Feb  8 11:59 parking_snippet.csv
```

```
> ls -lh
total 7126752
-rw-r--r--  1 leigh  staff   1.4M Feb  8 11:59 parking_snippet.csv
```

wc = Word Count

```
> wc -l parking_snippet.csv
10000 parking_snippet.csv
```

We know we have a 1.4MB gig file with 10000 rows.

Lets poke around before we decide on next steps...

head and tail commands give us bookends of a file, 10 lines by default, or can give argument for different amount

```
> head parking_snippet.csv

ticket_number,issue_date,issue_time,meter_id,marked_time,rp_state_plate,plate_expiry_date,vin,make,body_style,color,location,route,agency,violation_code,violation_description,fine_amount,agency_desc,color_desc,body_style_desc,loc_lat,loc_long
1000448820,12/13/2022 12:00:00 AM,1620,,0000,CA,202205,,JEEP,PA,WH,13922 OSBORNE
```

```
ST,19SL0,1,4000A1,,50,WESTERN,,PASSENGER CAR,34.2344923,-118.4360021
1000448831,12/13/2022 12:00:00 AM,1625,,0000,CA,202210,,MAZD,PA,BK,13922 OSBORNE
ST,19SL0,1,4000A1,,50,WESTERN,BLACK,PASSENGER CAR,34.2344923,-118.4360021
1000448842,12/16/2022 12:00:00 AM,1210,,0000,CA,202205,,HOND,PA,BR,9540 VAN NUYS
BLVD,19SL0,1,4000A1,,50,WESTERN,,PASSENGER CAR,34.24279,-118.450141
1000448853,12/19/2022 12:00:00 AM,845,,0000,CA,202209,,LEXU,PA,BK,14650 NOVICE
ST,19SL0,1,4000A1,,50,WESTERN,BLACK,PASSENGER CAR,34.243561,-118.450389
1000448864,12/19/2022 12:00:00 AM,855,,0000,CA,202109,,TOYO,PA,GY,14650 NOVICE
ST,19SL0,1,4000A1,,50,WESTERN,GREY,PASSENGER CAR,34.243561,-118.450389
1000448875,12/19/2022 12:00:00 AM,900,,0000,CA,202209,,DODG,TR,GY,14662 NOVICE
ST,19SL0,1,4000A1,,50,WESTERN,GREY,TRAILER,34.243561,-118.450389
1000448886,12/19/2022 12:00:00 AM,905,,0000,CA,202203,,CHEV,SU,RE,9703 CEDROS
AVE,19SL0,1,4000A1,,50,WESTERN,,,34.246246,-118.452149
1000448890,12/19/2022 12:00:00 AM,910,,0000,CA,202105,,JEEP,SU,GY,VESPER/VAN NUYS
BL,19SL0,1,4000A1,,50,WESTERN,GREY,,34.2450868,-118.4502511
1000448901,12/19/2022 12:00:00 AM,915,,0000,CA,202211,,BMW,PA,BK,VAN
NUYS/VESPER,19SL0,1,4000A1,,50,WESTERN,BLACK,PASSENGER CAR,34.1996715,-
118.4487824
```

```
tail -5 parking_snippet.csv
4578225982,01/08/2024 12:00:00 AM,2303,,0000,CA,202408,,TOYT,PA,GN,1414 CATALINA ST
N,0403A,54,80.58L,PREFERENTIAL PARKING,68,54 - DOT -
HOLLYWOOD,GREEN,PASSENGER CAR,34.0965474,-118.2951415
4578236040,01/08/2024 12:00:00 AM,2306,,0000,CA,0,,FORD,PA,BK,2352 RIPPLE
ST,,56,5200,DISPLAY OF PLATES,25,56 - DOT - CENTRAL,BLACK,PASSENGER
CAR,34.1070656,-118.2532953
4578551493,01/08/2024 12:00:00 AM,2316,,0000,CA,202411,,HYUN,PA,WT,7000 BROADWAY
S,,55,80.56E4+,RED ZONE,93,55 - DOT - SOUTHERN,WHITE,PASSENGER CAR,33.975921,-
118.2784538
4577559070,01/08/2024 12:00:00 AM,2300,,0000,CA,202310,,HOND,PA,GY,5327 VALLEY
BLVD,,56,22514,FIRE HYDRANT,68,56 - DOT - CENTRAL,GREY,PASSENGER CAR,34.0728495,-
118.1705965
4577694984,01/08/2024 12:00:00 AM,2322,,0000,CA,202204,,NISS,PA,BL,250 89TH ST
W,,55,80.56E4+,RED ZONE,93,55 - DOT - SOUTHERN,BLUE,PASSENGER CAR,33.9556935,-
118.2781085
```

What about a slice in the middle?

```
> awk 'NR >= 5 && NR <= 10' parking_snippet.csv
# the above returns record numbers between 5 and 10; to escape, press 'CTRL + C'

1000448853,12/19/2022 12:00:00 AM,845,,0000,CA,202209,,LEXU,PA,BK,14650 NOVICE
ST,19SL0,1,4000A1,,50,WESTERN,BLACK,PASSENGER CAR,34.243561,-118.450389
1000448864,12/19/2022 12:00:00 AM,855,,0000,CA,202109,,TOYO,PA,GY,14650 NOVICE
ST,19SL0,1,4000A1,,50,WESTERN,GREY,PASSENGER CAR,34.243561,-118.450389
1000448875,12/19/2022 12:00:00 AM,900,,0000,CA,202209,,DODG,TR,GY,14662 NOVICE
ST,19SL0,1,4000A1,,50,WESTERN,GREY,TRAILER,34.243561,-118.450389
1000448886,12/19/2022 12:00:00 AM,905,,0000,CA,202203,,CHEV,SU,RE,9703 CEDROS
AVE,19SL0,1,4000A1,,50,WESTERN,,,34.246246,-118.452149
```

```
1000448890,12/19/2022 12:00:00 AM,910,,0000,CA,202105,,JEEP,SU,GY,VESPER/VAN NUYS BL,19SL0,1,4000A1,,50,WESTERN,GREY,,34.2450868,-118.4502511
1000448901,12/19/2022 12:00:00 AM,915,,0000,CA,202211,,BMW,PA,BK,VAN NUYS/VESPER,19SL0,1,4000A1,,50,WESTERN,BLACK,PASSENGER CAR,34.1996715,-118.4487824
```

What if we are only interested in rows that contain a specific value?

```
> grep LEXU parking_snippet.csv
```

```
1000448853,12/19/2022 12:00:00 AM,845,,0000,CA,202209,,LEXU,PA,BK,14650 NOVICE ST,19SL0,1,4000A1,,50,WESTERN,BLACK,PASSENGER CAR,34.243561,-118.450389
1086691701,12/27/2022 12:00:00 AM,820,,0000,CA,202312,,LEXU,SU,GY,14720 ROSCOE BL,26TL1,1,8070,,163,WESTERN,GREY,,34.2209006,-118.4533903
1112867195,03/19/2023 12:00:00 AM,1152,41,0000,CA,202104,,LEXU,PA,BK,160 S FULLER AVE,,54,8058L,,68,54 - DOT - HOLLYWOOD,BLACK,PASSENGER CAR,34.0734064,-118.3494597
1112867324,03/28/2023 12:00:00 AM,828,,0000,CA,202307,,LEXU,PA,WH,130 N CROFT AVE,,54,8069BS,,73,54 - DOT - HOLLYWOOD,,PASSENGER CAR,34.07453,-118.3742332
1113070571,01/24/2023 12:00:00 AM,828,,0000,TX,202307,,LEXU,PA,WH,621 S CLOVERDALE,,54,8069BS,,73,54 - DOT - HOLLYWOOD,,PASSENGER CAR,34.0646238,-118.3458781
1113070932,01/22/2023 12:00:00 AM,927,,0000,CA,202309,,LEXU,PA,BL,7463 FRANKLIN AV,,54,8058L,,68,54 - DOT - HOLLYWOOD,BLUE,PASSENGER CAR,34.103362,-118.350399
1113072310,01/23/2023 12:00:00 AM,950,,0000,CA,202310,,LEXU,PA,WH,5230 CLINTON ST,,54,8069BS,,73,54 - DOT - HOLLYWOOD,,PASSENGER CAR,34.081579,-118.315873
```

We can use wc (word count) to list the number of results for our previous command:

```
grep LEXU parking_snippet.csv | wc -l
129
```

If I have a lot of files, how can I find potential problem files?
Give me everything over 2GB

```
>find . -size +2G -ls
```

Combining multiple commands, we can do more advanced operations by using pipes to string commands together.

For example:
Find files that are larger than 2 GB, then grab the first 10 rows, then grab the first 3 columns

```
find . -size +2G -exec awk -F ',' '{print $1 "," $2 "," $3}' {} \; | head
```

More info on **find -exec** and **awk**

- https://www.redhat.com/sysadmin/linux-find-command
- https://www.digitalocean.com/community/tutorials/how-to-use-the-awk-language-to-manipulate-text-in-linux
- https://www.digitalocean.com/community/tutorials/how-to-use-find-and-locate-to-search-for-

files-on-linux#executing-commands-on-find-results

So you can contruct mini-programs to do some complex operations, but can get kind of confusing or hard to remember the syntax, as a result there are some helper tools like csvkit

csvkit
csvkit is next tool to reach for:
https://csvkit.readthedocs.io/

Prereq -

- OSX - brew - https://brew.sh/
- Will be prompted to install xtools

To install, in terminal:

- - MacOS: brew install csvkit
- - Win/Linux: sudo pip install csvkit

relevent examples to parking data set
https://gist.github.com/swayson/6aa54c9d7f01190292c0
Adding some other helpful commands from: https://csvkit.readthedocs.io/en/latest/cli.html

List of all csvkit subtools and some bonus points (command line graphs and more!)
https://csvkit.readthedocs.io/en/latest/cli.html

# Don't need to do this, only an example of slicing a smaller dataset
Lets bite off a smaller section of our data to work with
head -10000 parking_Citations_20240111.csv > parking_snippet.csv

Some csvkit basics
* NOTE: in the following steps, since we are exploring the first 10,000 rows of the dataset, summary statistics will reflect this subset of the data only.

To view all columns in our file
cs parking_Citations_20240111.csv

```
  1: ticket_number
  2: issue_date
  3: issue_time
  4: meter_id
  5: marked_time
  6: rp_state_plate
  7: plate_expiry_date
  8: vin
  9: make
 10: body_style
 11: color
 12: location
 13: route
 14: agency
 15: violation_code
 16: violation_description
 17: fine_amount
 18: agency_desc
 19: color_desc
 20: body_style_desc
 21: loc_lat
 22: loc_long
```

Lets get some basic statistics about our values (type of data, # of unique values, etc..

```
csvstat parking_subset.csv
```

Example, looks like Tennessee has most tickets outside of California (at least in first 10k rows...)

```
...
  6. "rp_state_plate"

        Type of data:          Text
        Contains null values:  False
        Non-null values:       9999
        Unique values:         50
        Longest value:         2 characters
        Most common values:    CA (9309x)
                               TN (90x)
                               ME (86x)
                               TX (64x)
                               AZ (62x)
...
```

Of this subset of parking violation tickets, the most common car make is Toyota.

```
  9. "make"

    Type of data:          Text
    Contains null values:  True (excluded from calculations)
    Non-null values:       9650
    Unique values:         308
    Longest value:         5 characters
    Most common values:    TOYO (1617x)
                           HOND (1047x)
                           CHEV (824x)
                           FORD (767x)
```

NISS (672x)

-- Ditch Excel (for real)
# e.g. Say you saved the data file as a .xls file. You could use csvkit to quickly convert it to .csv format.

```
in2csv parking_Citations_20240111.xls > parking_Citations_20240111.csv
```

-- Conquer fixed-width formats

```
in2csv -f fixed -s parking_Citations_20240111.csv data.fixed > data.csv
```

-- Generate summary statistics for any CSV file

```
csvstat parking_snippet.csv
```

-- Find cells matching a regular expression,

- -c The column you want to look in
- -m The pattern you want to match, can use regular expressions as well
- csvgrep -c make -m JEEP parking_subset.csv

```
   csvgrep -c location -r "\d{3}" parking_subset.csv > tickets-given-at-3-number-address.csv
```

-- Execute a SQL query directly on a CSV file (show example query, make sure in first 10k rows)
Lets see what make of car got the most tickets...

```
csvsql --query "select make, count(make) as num_tix from parking_subset group by make order by
num_tix  desc" parking_subset.csv
```

Can also redirect into file:

```
csvsql --query "select make, count(make) as num_tix from parking_subset group by make order by
num_tix desc" parking_subset.csv > tickets-by-make.txt
```

Or clean up using csvlook

```
csvsql --query "select make, count(make) as num_tix from parking_subset group by make order by
num_tix  desc" parking_subset.csv | csvlook

| make  | num_tix |
| ----- | ------- |
| TOYO  |   1,617 |
| HOND  |   1,047 |
| CHEV  |     824 |
| FORD  |     767 |
| NISS  |     672 |
| DODG  |     376 |
| BMW   |     356 |
| KIA   |     310 |
```

-- Turn your data into a JSON lookup table

```
   csvjson --key slug --indent 4 geo.csv
```

-- Turn a CSV with latitude and longitude columns into GeoJSON

```
csvjson --lat loc_lat --lon loc_long --crs EPSG:4269 --indent 4 parking_subset.csv > parking_geo.json
```

-- Would need postgresql or other database installed to perform these oprations
-- Automatically create a SQL table and import a CSV into a database

```
csvsql --db postgresql:///my_parking_db --insert parking_snippet.csv
   psql -q my_parking_db -c "\d parking_snippet"
```

-- Extract a table from a SQL database into a CSV

```
sql2csv --db postgresql:///my_parking_db --query "select * from parking_subset" > extract.csv
```

-- Turn your Github issues into a CSV

```
   curl https://api.github.com/repos/onyxfish/csvkit/issues?state=open | in2csv -f json > issues.csv
```

Further exploration --

csvkit has handy suggestion page for text related tools including graphing and data transformation:
https://csvkit.readthedocs.io/en/latest/cli.html

Text based plotting and charts - https://github.com/sgreben/jp
https://github.com/red-data-tools/YouPlot

csvkit is ultimately python under the covers, and can be subject to some of same challenges, starts to bog down after a few million rows

xsv is a Rust based tool that addresses this issue and can handle 10s of millions of rows in acceptable time
https://github.com/BurntSushi/xsv

Example:

```
xsv stats parking_Citations_20240111.csv | xsv table
```

Takes about 20-30 seconds to compile stats for 20 million rows, csvkit takes 10-15 minutes

Option to squeeze out more performance from csvkit, some operations can be performed with streaming data and some need buffers
https://csvkit.readthedocs.io/en/latest/contributing.html#streaming-versus-buffering

You may be able to bump up the X Million row limit depending if the command is streaming or buffering

Additional options:

OpenRefine - https://openrefine.org/
Visual tool to explore data -- "Excel on steroids"

Additional questions and types of data you work with, please add below, it can help dial in future workshops or find a tool/solution for your needs

ie I work with large images

I work with large genomic datasets primarily on R so I would love to see some more tips on how to navigate them (while minimizing RAM usage hopefully?)/subset them for specific graphs/etc. on any data tool/platform

From Derek D. at UCM Strongly endorse RipGrep, a very fast text searcipgr
https://www.geeksforgeeks.org

Also if you would like to review the recording, please visit our Youtube channel. We should have the talk up before the end of the week. https://www.youtube.com/playlist?list=PLPtVwXLu7HGUzcBla2eHY7tBTsOgwU5OT

Tim - I like this book: https://jeroenjanssens.com/dsatcl/
Data Science at the CLI

Survey:
https://uclahs.zoom.us/survey/tJ0odu6vrDMsG9Sa4CEwIM8Vg3oo8qWg0a1y/postsurvey#/preview?fromModule=meeting&meetingNumber=tJ0odu6vrDMsG9Sa4CEwIM8Vg3oo8qWg0a1y&meetingType=meeting&tk=LoColnXRwMs9BRULheNqlinOKgnIdgSpmxAWSqldcuk.DQYAAAAXK7gHYhZnd1pDQ1RsNVIxQ0NySXhlZVNwQml3GGthbGxlbjJAbGlicmFyeS51Y2xhLmVkdQAAAAAAAAAAAAAAAAAAAAAAAAAAYZ0Fia2F2T3dTUlNsWGtPZC9oS1FQdz09AAAAAAAAHktyaXN0aWFuIEFsbGVuIC0gaGUvaGltIC0gVUNMQQ