Project 06 Answer Key and Grading Guide

https://datamine.purdue.edu/seminars/fall2019/stat19000project8.html

General guidelines

Generally we don't want to penalize incorrect answers too heavily. What's important is that the student makes an honest attempt at a solution and provides rationale for their methods. Remember, it's all about the learning.

- Each assignment is worth 10 points

Accepted file formats

To receive full credit, students must use the provided project template.

- If a solution's formatting deviates significantly from that of the template, deduct 0.5 points.

Adding comments to student assignments

Create a text file called `grader_notes.txt` in each student's project folder. Put any comments or corrections in there.

Project-specific guidelines

For any given problem…

- deduct 0.5 points for missing code (if code is required to solve this problem)
- deduct 0.5 points for missing output (if output is required to solve this problem)
- deduct 0.5 points for missing comments
- deduct 0.5 points for incorrect solutions

… for a minimum score of 0 on the individual problem.

Question 1a (2 pts)

> Use a pipeline in the terminal to solve Project 3, Question 1a, namely: Make a table of the number of transactions that occur in each of the four stores in the 84.51 data set called `5000_transactions.csv`, found in the folder `The_Complete_Journey_2_Master`.

```
# Use cd to navigate to the transactions data directory
cd /class/datamine/data/8451/The_Complete_Journey_2_Master/
# Use cat to list the contents of 5000_transactions.csv
cat 5000_transactions.csv | cut -d, -f7 | sort | uniq -c
```

```
>>>
2463343 CENTRAL
3263360 EAST
2221500 SOUTH
      1 STORE_R
2677350 WEST
```

Question 1b (2 pts)

> Use wget to download the file: `http://stat-computing.org/dataexpo/2009/airports.csv` Then use a pipeline in the terminal to solve Project 4, Question 1c, namely: Find the 4 cities that have the most airports.

```
# Use wget to download airports.csv
wget http://stat-computing.org/dataexpo/2009/airports.csv
# Use cat to list the contents of airports.csv
# Use cut with -d, to specify a comma delimiter and -f3,4 to get fields 3 and 4
# Use sort on the output to prepare it for the uniq command
# Use uniq with -c to get unique counts
# Use sort with -n to sort numerically
# Use tail with -n5 to get the last five entries
cat airports.csv | cut -d, -f3,4 | sort | uniq -c | sort -n | tail -n5
```

```
>>>
      6 "Indianapolis","IN"
      6 "Miami","FL"
      6 "New York","NY"
      8 "Houston","TX"
     12 NA,NA
```

Question 2a (2 pts)

Use a pipeline in the terminal to solve Project 4, Question 2c, namely: Which pickup location ID was the most popular for yellow taxi cab rides in June 2019?

```
# Use cd to navigate to the taxi data directory
cd /class/datamine/data/taxi/yellow/
# Use cat to list the contents of the June 2019 taxi data
# Use cut with -d, to specify a comma delimiter and -f8 to get the 8th field
# Use sort on the output to prepare it for the uniq command
# Use uniq with -c to get counts for each unique locationID
# Use use sort with -n to sort the resulting frequencies numerically
# Use tail with -n5 to get the last 5 entries in the sorted output
cat yellow_tripdata_2019-06.csv | cut -d, -f8 | sort | uniq -c | sort -n | tail -n5
```

```
>>>
 242355 186
 246880 162
 261418 236
 276362 161
 295057 237
```

Question 2b (2 pts)

Use a pipeline in the terminal to solve Project 4, Question 3c, namely: Which city has the largest number of donations so far, in the 2020 election season?

```
# Use cd to navigate to the election data directory
cd /class/datamine/data/election/
# Use cat to list the contents of itcont20202.txt
# Use cut with -d\| to specify a '|' delimiter and -f9,10 to get fields 9 and 10
# Use sort to prepare the output for the uniq command
# Use uniq with -c to get frequencies for each unique city|state
# Use sort with -n to sort the resulting frequencies
# Use tail with -n5 to get the last 5 entries of the sorted output
cat itcont2020.txt | cut -d\| -f9,10 | sort | uniq -c | sort -n | tail -n5
```

```
>>>
  28878 CHICAGO|IL
  30339 HOUSTON|TX
  35916 PHILADELPHIA|PA
  61126 WASHINGTON|DC
  77999 NEW YORK|NY
```

Question 3a (2 pts)

> The question is: How long does the code for these for solutions take to run (altogether) in R?

```r
# We'll use R's time functionality to time the execution time of our code
start_time = Sys.time()

# PROJECT 3, QUESTION 1a
transactions = read.csv('/class/datamine/data/8451/The_Complete_Journey_2_Master/5000_transactions.c
table(transactions$STORE_R)

# Project 4, QUESTION 1c
airports = read.csv('http://stat-computing.org/dataexpo/2009/airports.csv')
city_and_state = paste(airports$city, airports$state)
unique_cities = city_and_state[city_and_state != 'NA NA']
city_freq = sort(table(unique_cities))
tail(city_freq, n=4)

# PROJECT 4, QUESTION 2c
taxi = read.csv('/class/datamine/data/taxi/yellow/yellow_tripdata_2019-06.csv')
PU_location_freq = sort(table(taxi$PULocationID), decreasing=TRUE)
head(PU_location_freq, n=1)
pickup_locations = read.csv('https://s3.amazonaws.com/nyc-tlc/misc/taxi+_zone_lookup.csv')
pickup_locations[pickup_locations$LocationID == 237, ]

# PROBLEM 4, QUESTION 3c
elections = read.csv('/class/datamine/data/election/itcont2020.txt', sep='|')
elections$location = paste(elections$CITY, elections$STATE, sep=', ')
donation_freq = sort(table(elections$location), decreasing=TRUE)
head(donation_freq, n=1)


end_time = Sys.time()
end_time - start_time
```

```
>>>
Time difference of 4.254714 mins
```

NOTE: your time will vary based on server load and your solution methods.