Project 11 Answer Key and Grading Guide

https://datamine.purdue.edu/seminars/fall2019/stat19000project11.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)

In the 2020 election data (only considering the 9th field, not the 10th field) how many donations are from cities whose name ends in "burg"? How about "boro"? "shire"? "ton"? "town"? "ville"?

```r
# Use read.csv() to read in the 2020 election data
election = read.csv("/class/datamine/data/election/itcont2020.txt", sep="|")
# Use grep() to filter city names by several regular expressions corresponding
# to the given city name endings.
length(grep("BURG$", election$CITY))
length(grep("BORO$", election$CITY))
length(grep("SHIRE$", election$CITY))
length(grep("TON$", election$CITY))
length(grep("TOWN$", election$CITY))
length(grep("VILLE$", election$CITY))
```

```
>>>
> length(grep("BURG$", election$CITY, value=TRUE))
[1] 23621
> length(grep("BORO$", election$CITY, value=TRUE))
[1] 10618
> length(grep("SHIRE$", election$CITY, value=TRUE))
[1] 474
> length(grep("TON$", election$CITY, value=TRUE))
[1] 235188
> length(grep("TOWN$", election$CITY, value=TRUE))
[1] 23323
> length(grep("VILLE$", election$CITY, value=TRUE))
[1] 107955
```

Question 2a (1 pt)

How many donations in the 2020 election data have a consecutive, repeated vowel in the (personal) name of the donor? In other words, how many donations are from a donor with AA or EE or II or OO or UU in the donor's name?

```
# Use grep() to filter the NAME field
# The regex call looks for any of the double vowel patterns.
length(grep("(AA|EE|II|OO|UU)", election$NAME, value=TRUE))
```

```
>>>
[1] 158280
```

Question 2b (1 pt)

Which donor(s) has/have the longest name(s) in the 2020 election data, in terms of character length?

```
# Use as.character() to convert the NAME field type to character.
# Use nchar() to get the character count of this newly-converted field.
name_lengths = nchar(as.character(election$NAME))
# Use data.frame() to create a new data frame containing only names and their
# lengths.
names_and_lengths = data.frame(name_lengths, election$NAME)
# Use which.max() on the name lengths to find the index at which the max
# name length occurs.
max_index = which.max(names_and_lengths$name_lengths)
# Subset the newly-created data frame to get the value at the max index.
names_and_lengths[max_index, ]
```

```
>>>
                name_lengths
336019                   118

336019          election.NAME
                CAPITAN GRANDE BAND OF DIEGUENO MISSION INDIANS OF CALIFORNIA
                (BARONA GROUP OF CAPITAN GRANDE BAND OF MISSION INDIANS)
```

Question 2c

How many donations in the 2020 election have donors with the same last name as yours? (For instance, Dr Ward would look for people whose name starts with Ward. You want to make sure to check the beginning of the name, since the last names come first.)

```
# Use grep() to find names that begin with PARK.
# Use length() to find the number of occurences.
length(grep("^PARK,", election$NAME, value=TRUE))
```

```
>>>
[1] 711
```

Question 3a (2 pts)

> Use the method you learned in Project 9, Question 1, to cut only the 9th field from the data for all election years, and save the result in a file in your home directory called: myelectiontowns.txt

```
# Use tail with -n+2 to list the contents of all files, excluding the headers.
# Use cut with -d\| to specify a '|' delimeter and -f9 to isolate the 9th
# field.
tail -n+2 /class/datamine/data/election/*.txt | cut -d\| -f9 |
# Use grep with -v to remove headers created when listing the contents of all
# files. These headers will all contain "==>".
# Pipe the output to a file called myelectiontowns.txt
grep -v "==>" > myelectiontowns.txt
```

Question 3b (2 pts)

> How many donations come from cities whose names end in the phrase "ton", across all election years?

```
# Use read.delim() to read in the previously-created file.
electiontowns = read.delim("myelectiontowns.txt", quote="", header=FALSE)
# Use grep() to find towns that end in "TON".
towns_with_ton = grep("TON$", electiontowns$V1, value=TRUE)
# Use length() to find the number of matches.
length(towns_with_ton)
```

```
>>>
[1] 6253577
```

Question 3c (1 pt)

> How many unique city names are there in question 3b? (For this question, it is safe to only consider the city name and to ignore the State name.)

```
# Use unique() to get a vector of unique towns listed in the result of the
# previous question.
uniq_towns_with_ton = unique(towns_with_ton)
# Use length() to find the number of towns that end in "TON"
length(uniq_towns_with_ton)
```

```
>>>
[1] 3256
```