Project 12 Answer Key and Grading Guide

https://datamine.purdue.edu/seminars/fall2019/stat19000project12.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 (1 pts)

> Among these five words, which word appears most often as the first word of the "name" column: "Beautiful", "Charming", "Cozy", "Modern", or "Private"?

NOTE: There is more than one way to go about this problem. Grant full points for final answers as long as student arrive at their answers methodically provide comments explaining their process. The final numbers should not differ too much.

The Sane Solution

```
# Use grep() with a regex pattern to match each word specified by the problem:
#   ^beautiful  matches the character sequence 'beautiful' if it occurs at the
#               beginning of a line.

beautiful = grep("^beautiful", airbnb$name, ignore.case=TRUE)
charming = grep("^charming", airbnb$name, ignore.case=TRUE)
cozy = grep("^cozy", airbnb$name, ignore.case=TRUE)
modern = grep("^modern", airbnb$name, ignore.case=TRUE)
private = grep("^private", airbnb$name, ignore.case=TRUE)

# Use length to check the number of matches per specified pattern
length(beautiful)
length(charming)
length(cozy)
length(modern)
length(private)
```

```
>>>
> length(beautiful)
[1] 1269
> length(beautiful)
[1] 1269
> length(charming)
[1] 984
> length(cozy)
[1] 1835
> length(modern)
[1] 972
> length(private)
[1] 2573
```

The Slightly Overkill Solution

It is possible to check the frequencies of all the words in one go. This method also ignores leading non-alphanumeric characters. None of this is necessary at all.

```
# Use read.csv() to read in the AirBnB data
airbnb = read.csv("/scratch/scholar/park831/listings.csv")

# Use sub() and a regex pattern to extract the first word out of each listing
# name in the AirBnB data:
#   \\W*    matches 0 or more non-alphanumeric characters.
#   (\\w+)  is a capture group that matches for sequential alphanumeric characters.
#   .*      matches 0 or more of any character.
# The replacement parameter "\\1" will replace the entire line with the string
# captured by the (\\w+) capture group:
#   ex. airbnb$name             -> firstwords
```

```
#       "beautiful location"    -> "BEAUTIFUL"
#       "Cozy space near town"  -> "COZY"
# User toupper() to convert the output to upper case.
firstwords = toupper(sub(pattern="\\W*(\\w+).*", replacement="\\1", airbnb$name))

# Define a regex pattern for lines that contain words specified by the problem:
#   ^                                           matches the beginning of the line.
#   (beautiful|charming|cozy|modern|private)    is a capture group for any of the
#                                               words contained inside.
#   $                                           matches the end of the line.
matches1a = grep('^(beautiful|charming|cozy|modern|private)$',
                 firstwords, ignore.case=TRUE, value=TRUE)

# Use table() to get the frequencies of each word
# Use sort() with decreasing=TRUE to order the frequencies from greatest to
# least for each word.
sort(table(matches1a), decreasing=TRUE)
```

```
>>>
    PRIVATE      COZY BEAUTIFUL    MODERN  CHARMING
       2597      1860      1257      1007       994
```

Question 1b (1 pts)

> Among these five words, which word appears most often as the last word of the "name" column: "Apartment", "Beach", "Hollywood", "Home", or "House"?

NOTE: There is more than one way to go about this problem. Grant full points for final answers as long as student arrive at their answers methodically provide comments explaining their process. The final numbers should not differ too much.

The Sane Solution

```
# Use grep() with a regex pattern to match each word specified by the problem:
#   apartment$  matches the character sequence 'apartment' if it occurs at the
#               end of a line.

apartment = grep("apartment$", airbnb$name, ignore.case=TRUE)
beach = grep("beach$", airbnb$name, ignore.case=TRUE)
hollywood = grep("hollywood$", airbnb$name, ignore.case=TRUE)
home = grep("home$", airbnb$name, ignore.case=TRUE)
house = grep("house$", airbnb$name, ignore.case=TRUE)

# Use length to check the number of matches per specified pattern
length(apartment)
length(beach)
length(hollywood)
length(home)
length(house)
```

```
>>>
> length(apartment)
[1] 950
> length(beach)
[1] 1253
> length(hollywood)
[1] 1112
> length(home)
[1] 1304
> length(house)
[1] 1583
```

The Slightly Overkill Solution

```
# Use sub() and a regex pattern to extract the first word out of each listing
# name in the AirBnB data:
#   .*      matches 0 or more of any character.
#   \\s     matches a space character (tabs, spaces).
#   (\\w+)  is a capture group that matches for sequential alphanumeric characters.
#   \\W*    matches 0 or more non-alphanumeric characters.
# The replacement parameter "\\1" will replace the entire line with the string
# captured by the (\\w+) capture group:
#   ex. airbnb$name             -> firstwords
#       "beautiful location"    -> "BEAUTIFUL"
#       "Cozy space near town"  -> "COZY"
# User toupper() to convert the output to upper case.
lastwords = toupper(sub(".*\\s(\\w+)\\W*", "\\1", airbnb$name))

# Define a regex pattern for lines that contain words specified by the problem:
#   ^                                       matches the beginning of the line.
```

```
#   (apartment|beach|hollywood|home|house)  is a capture group for any of the
#                                           words contained inside.
#   $                                       matches the end of the line.
matches1b = grep('^(apartment|beach|hollywood|home|house)$',
                 lastwords, ignore.case=TRUE, value=TRUE)

# Use table() to get the frequencies of each word
# Use sort() with decreasing=TRUE to order the frequencies from greatest to
# least for each word.
sort(table(matches1b), decreasing=TRUE)
```

```
>>>
    BEACH      HOME     HOUSE HOLLYWOOD APARTMENT
     1611      1429      1317      1274      1050
```

Question 2 (5 pts)

How many 5-character tailnums are there, with the form: N, followed by 3 digits, followed by 1 alphanumeric character?

```
# Use read.csv() to read in the 2019 flight data
flights = read.csv("/class/datamine/data/flights/2019.csv")
# Store the tail nums in a variable
tailnums = flights$TAIL_NUM

# Use grep() with a regex pattern:
#   ^N          matches 'N' if it occurs at the beginning of the line.
#   [0-9]{3}    matches any three numeric characters from 0-9
#   \\w$        matches a single alphanumeric character that occurs at the end
#               of a line.
N_3digit_1alpha = grep("^N[0-9]{3}\\w$", tailnums, value=TRUE)
# Use length() to get a total number of matches
length(N_3digit_1alpha)
```

```
>>>
[1] 7344
```

How many 6-character tailnums are there, with the form: N, followed by 3 digits, followed by 2 alphanumeric characters?

```
# Use grep() with a regex pattern:
#   ^N          matches 'N' if it occurs at the beginning of the line.
#   [0-9]{2}    matches any two numeric characters from 0-9
#   \\w{2}$     matches any two alphanumeric characters that occur at the end
#               of a line.
N_3digit_2alpha = grep("^N[0-9]{3}\\w{2}$", tailnums, value=TRUE)
# Use length() to get a total number of matches
length(N_3digit_2alpha)
```

```
>>>
[1] 2936537
```

How many 5-character tailnums are there, with the form: 3 digits, followed by NV?

```
# Use grep() with a regex pattern:
#   ^[0-9]{3}   matches any two numeric characters from 0-9 that occur at the
#               beginning of the line.
#   NV$         matches "NV" as it occurs at the end of a line.
threedigit_NV = grep("^[0-9]{3}NV$", tailnums, value=TRUE)
# Use length() to get a total number of matches
length(threedigit_NV)
```

```
>>>
[1] 42415
```

How many tailnums a there, with the form: ALL?

```
# Use grep() with a regex pattern:
#   ^ALL$   matches "ALL" as it occurs simultaneously at the beginning and end
#           of a line.
all_tailnums = grep("^ALL$", tailnums, value=TRUE)
# Use length() to get a total number of matches
length(all_tailnums)
```

```
>>>
[1] 64
```

How many tailnums are blank?

```
# Use grep() with a regex pattern:
#   ^$   matches "" as it occurs simultaneously at the beginning and end of a
#                  line.
blank_tailnums = grep("^$", tailnums, value=TRUE)
# Use length() to get a total number of matches
length(blank_tailnums)
```

```
>>>
[1] 11287
```

Question 3 (3 pts)

Answer a fun question (of your choice) about a musical artist who you like, using the file of more than 4.5 million Amazon music reviews in the file:

Submissions for this problem will undoubtedly vary. Some examples are included below.

Example 1

Many recognizable musical acts have cited Fugazi as a musical influece: Nirvana, the Red Hot Chili Peppers, Lorde, and Tool to name a few. For reviews that mention Fugazi, what proportion mentions influence?

```
music = read.delim("/class/datamine/data/amazon/music.txt",
                   quote="", header=FALSE)
fugazi = grep("fugazi", music$V1, ignore.case=TRUE, value=TRUE)
fugazi_influence = grep("influen(ce|ces|ced|tial)",
                        fugazi, ignore.case=TRUE, value=TRUE)
length(fugazi_influence) / length(fugazi)
```

```
>>>
[1] 0.13879
```

Example 2

Britpop heavyweights Oasis and Blur had a famous rivalry in the mid 90s. How many reviews contain the word 'oasis'? 'blur'? How do average character counts for these reviews compare?

```
oasis = grep("oasis", music$V1, ignore.case=TRUE, value=TRUE)
blur = grep("blur", music$V1, ignore.case=TRUE, value=TRUE)

length(oasis)
length(blur)

mean_review_len_oasis = mean(nchar(oasis))
mean_review_len_blur = mean(nchar(blur))
```

```
>>>
> length(oasis)
[1] 4354
> length(blur)
[1] 7838

> mean_review_len_blur
[1] 1887.874
> mean_review_len_oasis
[1] 1438.59
```

Example 3

> Despite their American origin, the Minnesota-based alternative rock group Hüsker Dü are named after a Danish board game titled "Hūsker Dū?". What proportion of reviews that mention Hüsker Dü spell the band's name with full diacritics?

```
husker_du = grep("(H[ūüu]sker D[ūüu])", music$V1, ignore.case=TRUE, value=TRUE)
husker_du_diacritic = ("Hüsker Dü", husker_du, ignore.case=TRUE, value=TRUE)

length(husker_du)
length(husker_du_diacritic)
length(husker_du_diacritic) / length(husker_du)
```

```
>>>
> length(husker_du)
[1] 644
> length(husker_du_diacritic)
[1] 0
> length(husker_du_diacritic) / length(husker_du_diacritic)
[1] NaN
```