8 Processing Text Homework

7-24-20

There are six exercises below. You are required to provide solutions for at least four of the six. You are required to solve at least one exercise in R, and at least one in SAS. You are required to provide five solutions, each solution will be worth 10 points. Thus, you may choose to provide both R and SAS solutions for a single exercise, or you may solve five of the sixth problems, mixing the languages as you wish.

If you choose SAS for an exercise, you may use `IML`, `DATA` operations or `PROC SQL` at your discretion.

Warning I will continue restricting the use of external libraries in R, particularly `tidyverse` libraries. You may choose to use `ggplot2`, but take care that the plots you produce are at least as readable as the equivalent plots in base R. You will be allowed to use whatever libraries tickle your fancy in the midterm and final projects.

Reuse

For many of these exercises, you may be able to reuse functions written in prior homework. Define those functions here.

Exercise 1.

Write a loop or a function to convert a matrix to a `CSV` compatible string. Given a matrix of the form

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

produce a string of the form

`1,2,3\n4,5,6\n7,8,9`

where `\n` is the newline character. Use the matrix below as a test case.

```
Wansink <- matrix(c(268.1, 271.1, 280.9, 294.7, 285.6, 288.6, 384.4,
                    124.8, 124.2, 116.2, 117.7, 118.3, 122.0, 168.3,
                    18, 18, 18, 18, 18, 18, 18),ncol=3)

Ex_1.func <- function(x){
  paste(x[,1],x[,2],x[,3],sep=',',collapse='\n')
}
```

```
Ex_1 <- Ex_1.func(Wansink)
cat(Ex_1)

## 268.1,124.8,18
## 271.1,124.2,18
## 280.9,116.2,18
## 294.7,117.7,18
## 285.6,118.3,18
## 288.6,122,18
## 384.4,168.3,18
```

If you choose SAS, I've include `Wansink` as a data table and framework code for IML in the template. I used the `CATX` function in IML. I found I could do this in one line in R, with judicious use of `apply`, but I haven't found the equivalent in IML. Instead, I used a pair of nested loops to accumulate an increasingly longer string.

Exercise 2.

Calculate MSW, MSB, F and p for the data from Wansink Table 1 (Homework 4, Exercise 5) where

$$MSB = \frac{\sum_i n_i \, (x_i - \overline{x})^2}{k - 1}$$

$$MSW = \frac{\sum_i (n_i - 1) s_i^2}{N - k}$$

and $F = MSB/MSW$.

Start with the string:

```
WansinkString <-
"268.1,271.1,280.9,294.7,285.6,288.6,384.4\n124.8,124.2,116.2,117.7,118.3,122
.0,168.3\n18,18,18,18,18,18,18"

Ex_2a.func <- function(x,y){
    a <- unlist(strsplit(WansinkString[1],split = '\n'))
    z <- as.numeric(unlist(strsplit(a[y],split = ',')))
    return(z)
}

CaloriesPerServingMean <- Ex_2a.func(WansinkString,1)
CaloriesPerServingSD <- Ex_2a.func(WansinkString,2)
n <- Ex_2a.func(WansinkString,3)
```

Split this string into 3 substrings based on the newline character (`'\n'`), then tokenize the strings and convert the tokens to a create vectors of numeric values (i.e. `CaloriesPerServingMean`, `CaloriesPerServingSD`, `n`). Note this is roughly the reverse process from Exercise 1.

Use these vectors to compute and print MSW, MSB, F and p, where

```
MSW.func <- function(s,n){ # begin function
  k = length(n)
  N = sum(n)
  SSW <- 0
  for(i in 1:k){ # begin for loop
    SSW <- SSW + (n[i] - 1)*(s[i]^2)
  } # end for loop
  return(SSW/(N-k))
} # end function

MSW <- MSW.func(CaloriesPerServingSD,n)

MSB.func <- function(x,n){ # begin function
  k = length(n)
  SSA <- 0
  for(i in 1:k){ # begin for loop
    SSA <- SSA + (n[i]*((x[i]-mean(x))^2))
  } # end for loop
  return(SSA/(k-1))
} # end function

MSB <- MSB.func(CaloriesPerServingMean,n)

N <- sum(n)
k <- length(n)

F.ratio <- MSB/MSW
p <- 1-pf(F.ratio,k-1,N-k)

cat(' MSW =',MSW,'\n','MSB =',MSB,'\n','F ratio =',F.ratio,'\n','p =',p)
```

```
##  MSW = 16508.6
##  MSB = 28815.96
##  F ratio = 1.745512
##  p = 0.1163133
```

If you use SAS, I've provided macro variables that can be tokenized in either macro language or using SAS functions. You can mix and match macro, DATA, IML or SQL processing as you wish, but you must write code to convert the text into numeric tokens before processing.

Compare your results from previous homework, or to the resource given in previous homework, to confirm that the text was correctly converted to numeric values.

Comment - I compared my values to the values from Homework 4 Exercise 5 and they match.

Exercise 3.

Repeat the regression analysis from Homework 4, Exercise 4, but start with the text

```
CaloriesPerServingMean <- "268.1 | 271.1 | 280.9 | 294.7 | 285.6 | 288.6 |
384.4"
Year <- "1936 | 1946 | 1951 | 1963 | 1975 | 1997 | 2006"
```

Note that by default, `strsplit` in R will read `split` as a regular expression, and | is a special character in regular expressions. You will need to change one of the default parameters for this exercise.

Tokenize these strings and convert to numeric vectors, then use these vectors to define

$$y = \begin{pmatrix} 268.1 \\ 271.1 \\ \vdots \\ 384.4 \end{pmatrix} = \begin{pmatrix} 1 & 1936 \\ 1 & 1946 \\ \vdots & \vdots \\ 1 & 2006 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}^t = \mathbf{X\beta}$$

Solve for and print $\hat{\beta}$.

If you use SAS, I've provided macro variables that can be tokenized in either macro language or using SAS functions. You can mix and match macro, DATA, IML or SQL processing as you wish, but you must write code to convert the text into numeric tokens before processing.

Compare your results from previous homework, or to the resource given in previous homework, to confirm that the text was correctly converted to numeric values.

Exercise 4

Load the file `openmat2015.csv` from D2L into a data table or data frame. These data are from https://news.theopenmat.com/high-school-wrestling/high-school-wrestling-rankings/final-2015-clinch-gear-national-high-school-wrestling-individual-rankings/57136. This is a list of top-ranked high school wrestlers in 2015, their high `School`, `Weight` class and in some cases the `College` where they expected to enroll and compete after high school.

Use partial text matching to answer these questions. To show your results, print only the rows from the table that match the described text patterns, but to save space, print only `Name`, `School` and `College`. Each of these can be answered in a single step.

```
path = 'openmat2015.csv'
openmat.dat <- read.csv(path,header = TRUE)
```

- Which wrestlers come from a `School` with a name starting with `St.`?

```
openmat.dat[grepl("^St\\.",openmat.dat$School),c(3,5,7)]
```

```
##                  Name                   School       College
## 1         Cade Olivas           St. John Bosco
## 17        John Tropea      St. Joseph Montvale
## 30        Mitch Moore         St. Paris Graham
## 37         Joey Prata        St. Christopher's
## 50       Eli Stickley         St. Paris Graham     Wisconsin
## 64     Mitchell McKee St. Michael-Albertville Minnesota '16
## 67        Eli Seipel         St. Paris Graham    Pittsburgh
## 76       Ben Lamantia             St. Anthonys      Michigan
## 94    Austin O'Connor                 St. Rita
## 99    Hunter Ladnier                St. Edward
## 128       Brent Moore         St. Paris Graham
## 153       Kyle Lawson         St. Paris Graham
## 161    Alex Marinelli         St. Paris Graham      Iowa '16
## 182  Anthony Valencia           St. John Bosco Arizona State
## 183       Logan Massa                St. Johns      Michigan
## 201    Zahid Valencia           St. John Bosco Arizona State
## 217     Jordan Joseph St. Michael-Albertville
## 251 Christian Colucci         St. Peter's Prep        Lehigh
## 255   Ian Butterbrodt           St. Johns Prep         Brown
```

- Which wrestlers were intending to attend an Iowa College (look for Iowa in the College column)?

```
openmat.dat[grepl("Iowa",openmat.dat$College),c(3,5,7)]
```

```
##                Name                  School       College
## 21     Justin Mejia                  Clovis      Iowa '17
## 24   Jason Renteria  Oak Park-River Forest      Iowa '17
## 65   Markus Simmons            Broken Arrow    Iowa State
## 121 Michael Kemerer       Franklin Regional          Iowa
## 122     Max Thomsen         Union Community Northern Iowa
## 155     Kaleb Young             Punxsutawney      Iowa '16
## 161  Alex Marinelli        St. Paris Graham      Iowa '16
## 166   Bryce Steiert      Waverly-Shell Rock Northern Iowa
## 176     Paden Moore Jackson County Central Northern Iowa
## 194   Isaiah Patton        Dowling Catholic Northern Iowa
## 196 Jacob Holschlag                   Union Northern Iowa
## 197 Colston DiBlasi                Park Hill    Iowa State
## 204    Taylor Lujan               Carrollton Northern Iowa
## 233     Cash Wilcke                  OA-BCIG          Iowa
## 244    Ryan Parmely        Maquoketa Valley    Upper Iowa
```

- Which wrestlers were intending to start College in 2016 or 2017 (College will end with 16 or 17)?

```
openmat.dat[grepl("[6-7]$",openmat.dat$College),c(3,5,7)]
```

```
##                Name                   School           College
## 21    Justin Mejia                   Clovis          Iowa '17
## 24  Jason Renteria   Oak Park-River Forest          Iowa '17
## 45   Kyle Norstrem                 Brandon Virginia Tech '16
```

```
## 46    Jack Mueller        Wyoming Seminary      Virginia '16
## 51      Ty Agaisse               Delbarton     Princeton '16
## 64  Mitchell McKee St. Michael-Albertville     Minnesota '16
## 126  Hayden Hidlay          Mifflin County      NC State '16
## 145   Jake Wentzel              South Park          Pitt '16
## 155    Kaleb Young            Punxsutawney          Iowa '16
## 161 Alex Marinelli        St. Paris Graham          Iowa '16
## 186    Nick Reenan        Wyoming Seminary  Northwestern '16
```

- Which wrestlers are intending compete in a sport other than wrestling? (look for a sport in parenthesis in the `College` column. Note - `(` is a special character in regular expressions, so to match the exact character, it needs to be preceded by the escape character `\`. However, `\` in text strings is a special character, so itself must be preceded by the escape character.

```
openmat.dat[grepl("\\)$",openmat.dat$College),c(3,5,7)]
```

```
##                Name           School                        College
## 218    Chase Osborn             Penn         Minnesota (Baseball)
## 225   Tevis Barlett    Cheyenne East              Washington (FB)
## 230     Jan Johnson Governor Mifflin                    Akron(FB)
## 261 Michael Johnson Montini Catholic              Yale (Football)
## 264   Gage Cervenka          Emerald           Clemson (Football)
## 267     Jake Marnin   Southeast Polk Southern Illinois (Football)
## 277     Que Overton            Jenks          Oklahoma (Football)
## 279  Norman Oglesby   Benjamin Davis        Cincinnati (Football)
```

Exercise 5.

Load the file `openmat2015.csv` (for SAS use `openmat2015SAS.csv`) into a data table or data frame. We wish to know how many went on to compete in the national championship in 2019, so we will merge this table with the data from Homework 7, `ncaa2019.csv`. The `openmat2015.csv` data contains only a single column, `Name`. You will need to split the text in this column to create the columns `First` and `Last` required to merge with `ncaa2019.csv`. **Do not print these tables in the submitted work**

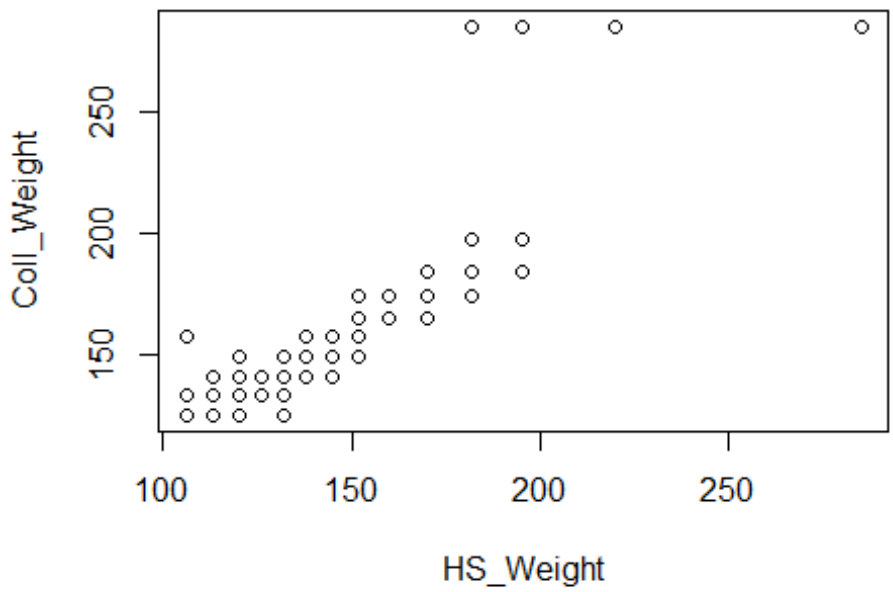
What is the relationship between high school (`openmat2015.csv`) and college weight classes (`ncaa2019.csv`)? print a contingency `table` comparing `Weight` from `openmat2015.csv` and `Weight` from `ncaa2019.csv`, or produce a scatter plot or box-whisker plot, using high school weight class as the independent variable.

```
path='ncaa2019.csv'
ncaa.dat <- read.csv(path,header=TRUE)

openmat.dat$First = sapply(strsplit(as.character(openmat.dat$Name),split = "
"), function(x) x[1])
openmat.dat$Last = sapply(strsplit(as.character(openmat.dat$Name),split = "
"), function(x) x[length(x)])
```

```r
Ex_5.dat <- merge(openmat.dat,ncaa.dat,by = c('First','Last'))
Ex_5.dat <- data.frame(First = Ex_5.dat$First, Last = Ex_5.dat$Last,
                       HS_Weight = Ex_5.dat$Weight.x,
                       Coll_Weight = Ex_5.dat$Weight.y)

Weight_Comp <- with(Ex_5.dat, table(Coll_Weight,HS_Weight))
print(Weight_Comp)
```

```
##             HS_Weight
## Coll_Weight 106 113 120 126 132 138 145 152 160 170 182 195 220 285
##         125   2   1   5   0   2   0   0   0   0   0   0   0   0   0
##         133   1   1   3   1   2   0   0   0   0   0   0   0   0   0
##         141   0   1   1   1   4   2   1   0   0   0   0   0   0   0
##         149   0   0   1   0   1   1   1   1   0   0   0   0   0   0
##         157   1   0   0   0   0   1   4   1   0   0   0   0   0   0
##         165   0   0   0   0   0   0   0   2   4   2   0   0   0   0
##         174   0   0   0   0   0   0   0   2   4   2   2   0   0   0
##         184   0   0   0   0   0   0   0   0   0   2   2   2   0   0
##         197   0   0   0   0   0   0   0   0   0   0   3   1   0   0
##         285   0   0   0   0   0   0   0   0   0   0   1   1   5   1
```

```r
plot(Coll_Weight~HS_Weight, data = Ex_5.dat)
```



Exercise 6

Use the file `vehicles.csv` (or `vehiclesSAS.csv` for SAS). These data were downloaded and modified from https://www.fueleconomy.gov/feg/download.shtml.

Read the data into a data frame or data table. This file has ~35000 rows; we will reduce the size of this data by filtering for text in different columns. You should use pattern matching (i.e. regular expressions - `grep` - or wildcard operators in SQL) for the filters on string data columns. **Do not print these tables in the submitted work**

It may help debugging if you print the number of rows in the table after each step. You will be required to produce plots for parts **e** and **f**, but it may also help you to produce box-whisker plots at each step, using the selection column for each plot (i.e. `plot(UHighway ~ factor(VClass), data=vehicles.dat)` after part **a**)

```
path='vehicles.csv'
vehicles.dat <- read.csv(path,header = TRUE)
pre <- dim(vehicles.dat)[1]
```

Part a.

Select only rows with data for cars (not vans, etc.). Match `Cars` in the `VClass` column. This should remove ~17000 rows.

```
vehicles.dat <- vehicles.dat[grepl('Cars',vehicles.dat$VClass),]
post <- dim(vehicles.dat)[1]
cat(pre-post)

## 16791
```

Part b.

Select only rows with data for regular or premium gasoline. You can match `Gasoline` in the `fuelType1` column and exclude rows with `Midgrade` in that column.

```
vehicles.dat <- vehicles.dat[grepl('Gasoline',vehicles.dat$fuelType1),]
dim(vehicles.dat)[1]

## [1] 17488
```

Part c.

Select for certain classes of transmissions. Match for the pattern `*-spd` in the `trany` column and exclude rows with `Doubled` in that column. There should be ~13000 rows remaining at this step.

```
vehicles.dat <- vehicles.dat[grepl('*-spd',vehicles.dat$trany),]
vehicles.dat <- vehicles.dat[!grepl('Doubled',vehicles.dat$trany),]
dim(vehicles.dat)[1]

## [1] 13143
```

Part d.

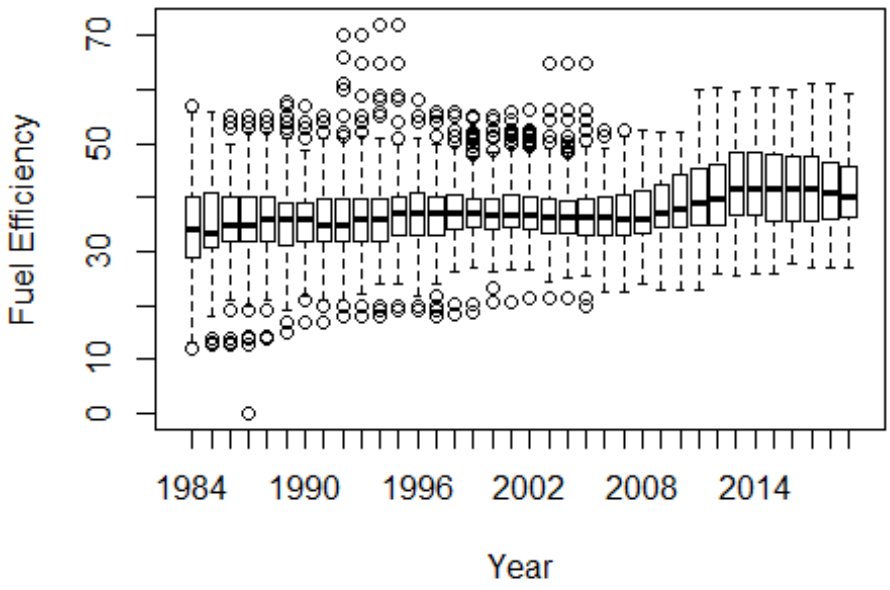
Select only rows with values of 4,6,8 in the `cylinders` column.

```
vehicles.dat <- vehicles.dat[grepl('4|6|8',vehicles.dat$cylinders),]
dim(vehicles.dat)[1]

## [1] 12551
```

Part e.

Select only rows with `year` before 2020. Produce a box-whisker plot of fuel efficiency (`UHighway`) with `year` as the independent variable. There should be <12500 rows remaining at this step.

```
vehicles.dat <- vehicles.dat[!grepl('2020|2021',vehicles.dat$year),]
dim(vehicles.dat)[1]

## [1] 12397

boxplot(UHighway~year,data = vehicles.dat,ylab='Fuel Efficiency',xlab='Year')
```



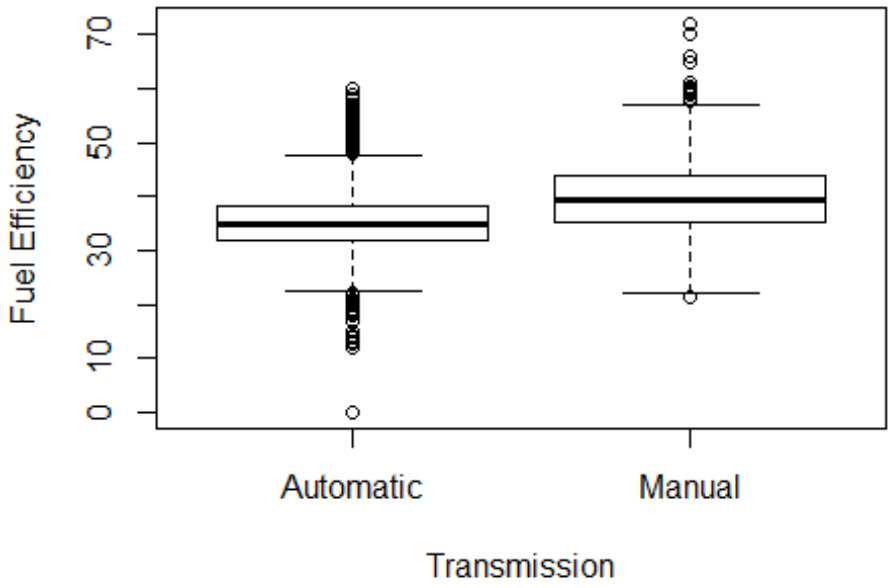
Part f.

Tokenize the strings in the `trany` column into two substrings. The first will identify the type of transmission (`Manual` or `Automatic`) and the second will identify the number of gears (`3-spd`, `4-spd`), etc. Use first substring for each row to create new string data column

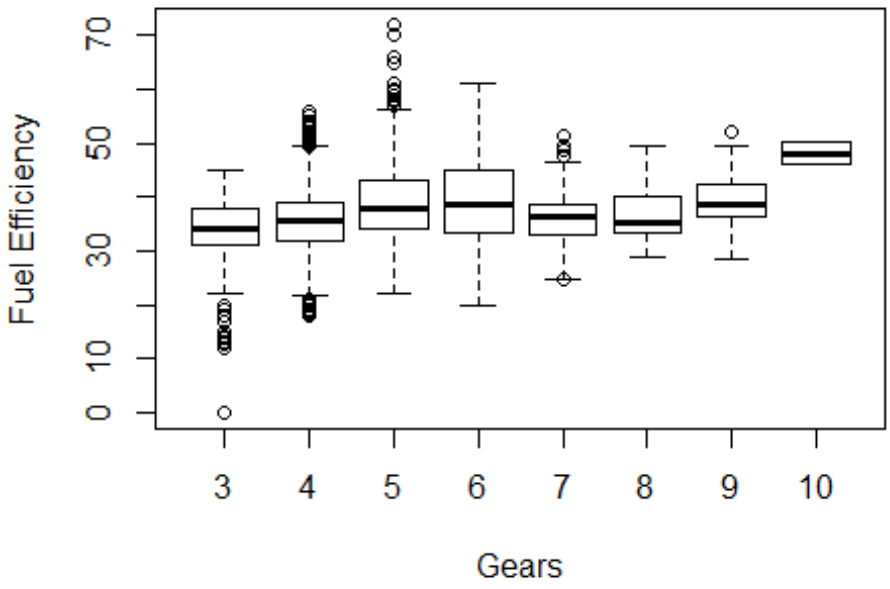
`Transmission`, with values `Manual` or `Automatic`. Tokenize the second substring and convert the integer characters to integer values; add this as a new numeric data column `Gears`.

Produce two box-whisker plot of fuel efficiency (`UHighway`) as the dependent variable, with `Transmission` and `Gears` as the independent variables.

```
vehicles.dat$Transmission <-
sapply(strsplit(as.character(vehicles.dat$trany),split = " "), function(x)
x[1])
vehicles.dat$Gears <- sapply(strsplit(as.character(vehicles.dat$trany),split
= " "), function(x) x[length(x)])
vehicles.dat$Gears <-
as.numeric(sapply(strsplit(as.character(vehicles.dat$Gears),split = "-"),
function(x) x[1]))

boxplot(UHighway~Transmission,data=vehicles.dat,ylab='Fuel Efficiency')
```




```
boxplot(UHighway~Gears,data=vehicles.dat,ylab='Fuel Efficiency')
```


Fuel Efficiency
0
10
30
50
70
3
4
5
6
7
8
9
10
Gears