8 Processing Text Homework

Gavin Gunawardena

7/23/2021

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 final project.

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

Reuse

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

```
msb.solve <- function(x,sample_size) {
  #x can be either standard deviation or mean
  x_weighted_avg <- weighted.mean(x, rep(1,7))
  #print(x_weighted_avg)
  iterations <- rep(sample_size,length(x))
    pt_1 <- sum(iterations%*%(x-x_weighted_avg)^2)
  pt_2 <- length(x) - 1
  return(pt_1/pt_2)
}
msw.solve <- function(x, sample_size) {
  #x must be standard deviation
  iterations <- rep(sample_size,length(x))
  pt_1 <- sum((iterations-1)*(x^2))
  pt_2 <- sum(iterations)-length(x)
  return(pt_1/pt_2)
}
```

Exercise 1.

Write a loop or a function to convert a matrix to a `CSV` compatible string. For example, given a matrix of the form

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

produce a string of the form

`1,2,3\n4,5,6`

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

```
# Create Function
convert.to.string <- function(x) {
  matrix.Shape <- dim(x)
  Final.String <- ""
  # Go through each row of the matrix
  for (i in 1:matrix.Shape[2]){
    Final.String <- paste(Final.String,paste(paste(x[,i],
collapse=","),sep=""),sep="")
    # Concatenate \n if it's not the final row
    if(i != matrix.Shape[2])
    {
      Final.String <- paste(Final.String,"\n",sep="")
    }
  }
  return(Final.String)
}
# Test Data
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),ncol=2)

# Run Function
convert.to.string(Wansink)

## [1]
"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
,168.3"
```

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 3) where

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

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

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"
```

Split this string into 3 substrings based on the newline character (`'\n'`). Next, tokenize the strings based on the `','` character 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.

```
# Split and Tokenize the String
Wansink.Split <- strsplit(WansinkString, "\n")

# Wansink.Split[1]


# Tokenize the String
Wansink.Tokenized <- as.numeric(unlist(strsplit(unlist(Wansink.Split), ",")))
#
# Wansink.Tokenized




# Convert to Matrix
Wansink.Matrix <- matrix(c(Wansink.Tokenized), ncol=7, byrow = TRUE, dimnames
= list(c("CaloriesPerServingMean","CaloriesPerServingSD","n"),
c("1936","1946","1951","1963","1975","1997","2006")))
# print(Wansink.Matrix)
# class(Wansink.Matrix)

# Convert to Dataframe
Wansink.dat <- as.data.frame(Wansink.Matrix)


#Solve via functions from Week 4 assignment
MSB <- msb.solve(Wansink.Matrix[1,],Wansink.Matrix[3,3])
MSW <- msw.solve(Wansink.Matrix[2,],Wansink.Matrix[3,3])
df1 <- length(Wansink.dat[1,])-1
df2 <- sum(rep(Wansink.Matrix[3,3],length(Wansink.dat[1,])))-
length(Wansink.dat[1,])
F_Stat <- MSB / MSW
P_val <- pf(F_Stat,df1,df2,lower.tail=FALSE)

cat("MSB:", MSB, "\n")

## MSB: 28815.96

cat("MSW:", MSW, "\n")
```

```
## MSW: 16508.6

cat("F:",F_Stat, "\n")

## F: 1.745512

cat("P:", P_val, "\n")

## 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.

Exercise 3.

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`. Merge the data on `First` and `Last` names. 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`.

For example, the 22^{nd} wrestler in `openmat2015.csv`, `Danny  Vega` will be split into `Danny` and `Vega`. Danny did not wrestle in the 2019 NCAA tournament, but he did wrestle the 2021 tournament for SDSU. The 43^th wrestler in `openmat2015.csv` will be be split into `Yianni`, `Diakomihalis`. Yianni was the national champion in 2019 at 141 pounds. Thus, the merged data set will contain `Yianni`, `Diakomihalis`, but not `Danny`, `Vega`. **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.

```
# import data
OpenMat = "openmat2015.csv"
OpenMat.dat <- read.csv(OpenMat,header=TRUE)



NCAA = "ncaa2019.csv"
NCAA.dat <- read.csv(NCAA,header=TRUE)



# format into usable data frames
```

```r
OpenMat.dat$First <- sapply(strsplit(OpenMat.dat$Name, ' '), function(x)
x[1])
OpenMat.dat$Last <- sapply(strsplit(OpenMat.dat$Name, ' '), function(x) x[2])



# merge the data
Merged.OpenMat.NCAA <- merge(OpenMat.dat, NCAA.dat, by=c("First","Last"),
all=FALSE)



colnames(Merged.OpenMat.NCAA) <- c("First_Name", "Last_Name", "Weight_2015",
"Rank", "Full_Name", "Year", "School", "State", "College", "Previous_Rank",
"Weight_2019", "Finish")

# Merged.OpenMat.NCAA

# Create a contingency table

Contingency_Table.OpenMMat.NCAA <- table(Merged.OpenMat.NCAA$Weight_2015,
Merged.OpenMat.NCAA$Weight_2019)

Contingency_Table.OpenMMat.NCAA

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

Going by these contingency table results, most of these wrestlers gained weight between high school and college. Very few stayed at the same weight or lowered their weight.

Exercise 4

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 or numeric values 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**)

```
Vehicles = "vehicles.csv"
Vehicles.dat <- read.csv(Vehicles,header=TRUE)
```

Part a.

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

```
Vehicles.4a.dat = Vehicles.dat[grepl('cars', Vehicles.dat$VClass, ignore.case
= TRUE), ]
# Vehicles.4a.dat
length(Vehicles.4a.dat[,1])

## [1] 17969
```

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.

```
include.words <- c("premium","regular")
Vehicles.4b.dat <- Vehicles.4a.dat[grepl(paste(include.words, collapse =
"|"), Vehicles.4a.dat$fuelType1, ignore.case = TRUE), ]
# Vehicles.4b.dat
length(Vehicles.4b.dat[,1])

## [1] 17451
```

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.

```
include.dat <- Vehicles.4b.dat[grepl("-spd", Vehicles.4b.dat$trany,
ignore.case = TRUE),]
```

```
# Vehicles.4b.dat[include, ]
exclude.dat <-include.dat[!grepl("doubled", include.dat$trany, ignore.case =
TRUE),]
Vehicles.4c.dat <- exclude.dat
# Vehicles.4c.dat
length(Vehicles.4c.dat[,1])

## [1] 13106
```

Part d.

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

```
Vehicles.4d.dat <-
Vehicles.4c.dat[grepl("[4,6,8]",Vehicles.4c.dat$cylinders),]
# Vehicles.4d.dat
length(Vehicles.4d.dat[,1])

## [1] 12514
```

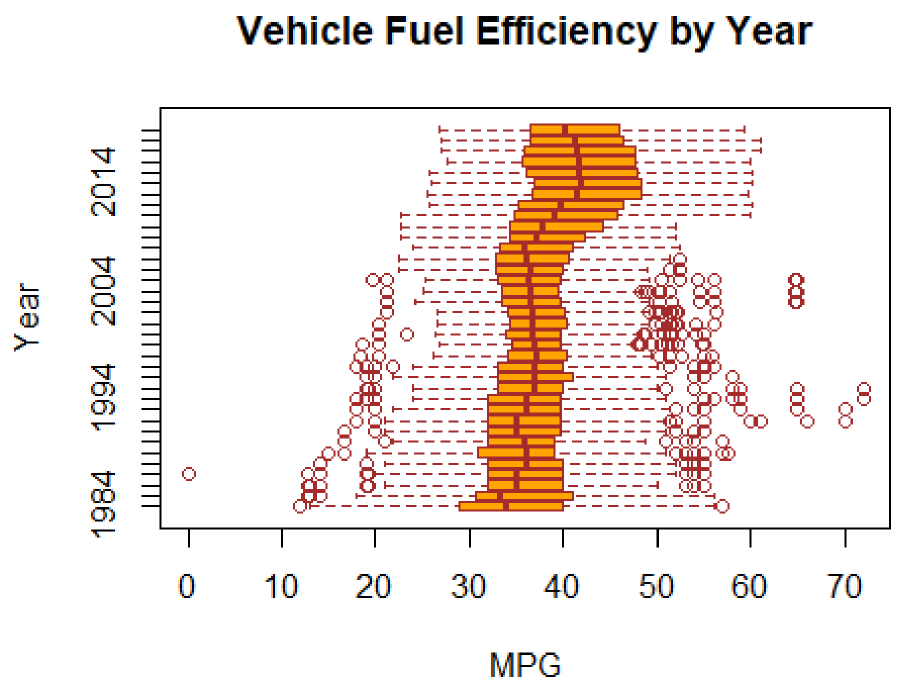
Part e.

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.

```
# Querying
Vehicles.4e.dat <- Vehicles.4d.dat[Vehicles.4d.dat$year<2020,]
# Vehicles.4e.dat




# Box Plot
boxplot(UHighway~year,
data = Vehicles.4e.dat,
main = "Vehicle Fuel Efficiency by Year",
xlab = "MPG",
ylab = "Year",
col = "orange",
border = "brown",
horizontal = TRUE
)
```



Part f.

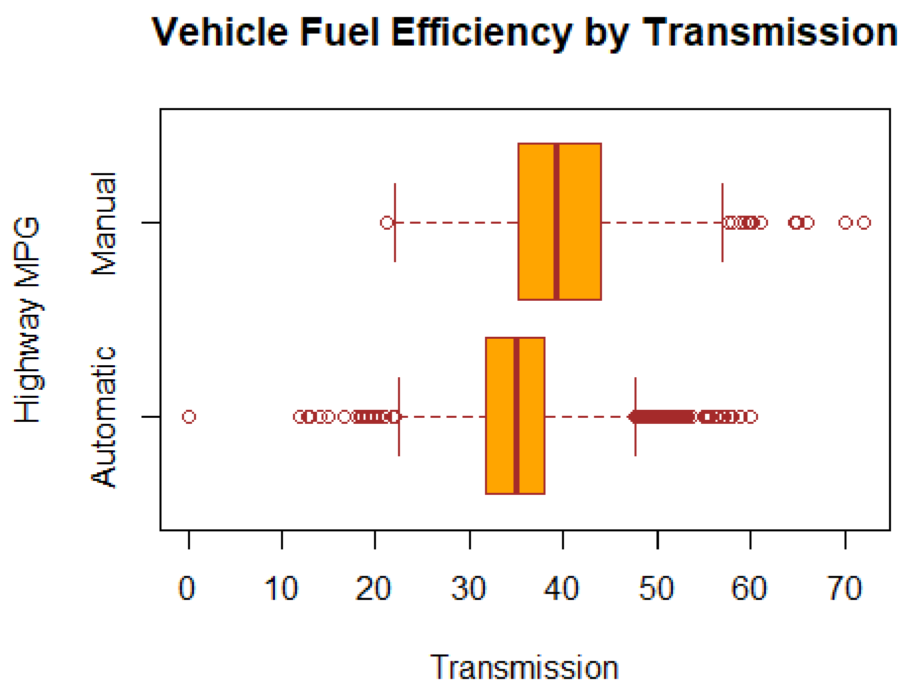
Tokenize the strings in the `trany` column into two substrings. The first will identify the type of transmission (`Manual` or `Automatic`) and the second will identify the number of gears (`3-spd`, `4-spd`), etc. Use first substring for each row to create new string data column `Transmission`, with values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

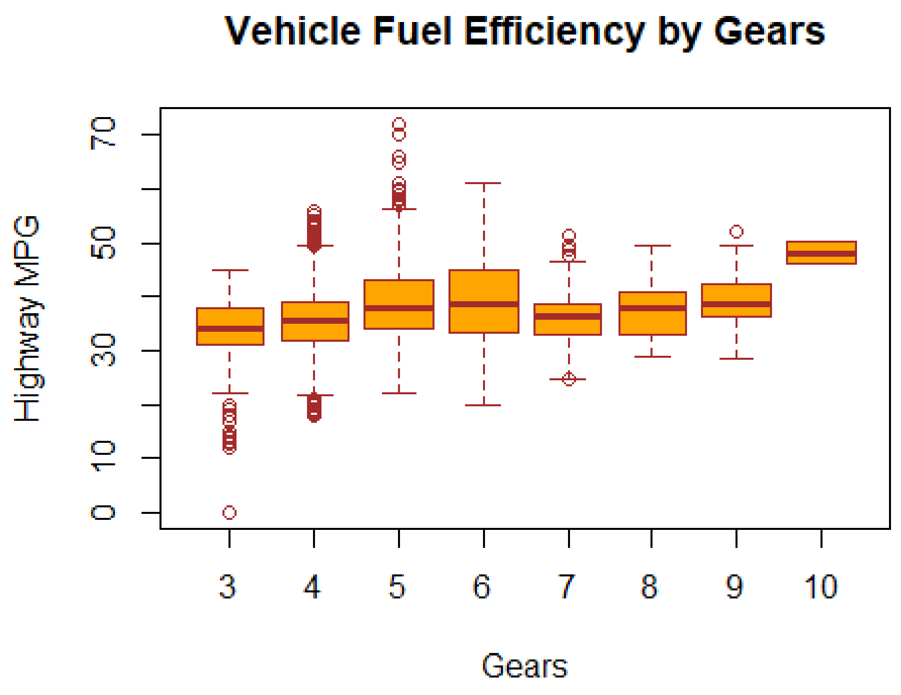
```
Vehicles.4f.dat <- Vehicles.4e.dat

# Tokenize and Create new Columns for Transmission and Gears
Vehicles.4f.dat$Transmission <- sapply(strsplit(Vehicles.4f.dat$trany, ' '),
function(x) x[1])
Vehicles.4f.dat$GearsSpd <- sapply(strsplit(Vehicles.4f.dat$trany, ' '),
function(x) x[2])
Vehicles.4f.dat$Gears <- sapply(strsplit(Vehicles.4f.dat$GearsSpd, '-'),
function(x) as.integer(x[1]))
# Vehicles.4f.dat
# str(Vehicles.4f.dat)

# Box Plot 1
boxplot(UHighway~Transmission,
data = Vehicles.4f.dat,
main = "Vehicle Fuel Efficiency by Transmission",
xlab = "Transmission",
ylab = "Highway MPG",
col = "orange",
border = "brown",
horizontal = TRUE
)
```



```
# Box Plot 2
boxplot(UHighway~Gears,
data = Vehicles.4f.dat,
main = "Vehicle Fuel Efficiency by Gears",
xlab = "Gears",
ylab = "Highway MPG",
col = "orange",
border = "brown",
horizontal = FALSE
)
```


Vehicle Fuel Efficiency by Gears
Highway MPG
0
10
30
50
70
3
4
5
6
7
8
9
10
Gears