8 Processing Text Homework

There are six exercises below. You are required to provide five solutions, with the same options for choosing languages as with the last exercise. You can provide solutions in two languages for one exercise only (for example, Ex. 1,2,3,5 in R and Ex. 1 in SAS is acceptable, Ex. 1,2,3 in SAS and Ex. 1,2 in R is not).

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)
```

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

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

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

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\
```

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

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.

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.

- Which wrestlers come from a `School` with a name starting with `St.`?
- Which wrestlers were intending to attend an Iowa `College` (look for `Iowa` in the `College` column)?
- Which wrestlers were intending to start `College` in 2016 or 2017 (`College` will end with 16 or 17)?
- 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.

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.

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**)

Part a.

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

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.

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.

Part d.

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

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.

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.