Grade: 16/15

Problem Set 1

Political Data Science - Spring 2020

Gencer, Alper Sukru

January 23, 2020

Instructions

1. The following questions should each be answered within an R script. Be sure to provide many comments in the script to facilitate grading. Undocumented code will not be graded. Once your script is finished, please email Dominique at dlockett@wustl.edu.
2. You may work in teams, but each student should develop their own R script. To be clear, there should be no copy and paste. Each keystroke in the assignment should be your own.
3. If you have any questions regarding the Problem Set, contact the TA or use her office hours.
4. For students new to programming, this may take a while. Get started.

Working with data in R

For this assignment, I have subsetted the expenditures data for all campaigns and PACs available from Open Secrets. The reduced dataset is available at:

https://www.dropbox.com/s/z6gw9lvve6jogi5/Expends2002.txt

Before you begin, you should get familiar with the variables. The codebook for this dataset is available at:

http://www.opensecrets.org/resources/datadictionary/Data%20Dictionary%20Expenditures.htm

Below is a detailed listing of the data management tasks that you will have to complete for this assignment. You should provide the R script needed to execute each task with clear documentation.

1. Open the dataset as a dataframe. This dataframe should have the following properties:

1. The column names should match the column names in the original dataset.
2. The row names should correspond to the variable ID in the original dataset.

```
rm(list = ls())
# install.packages("data.table")
library("data.table")
library(readr)
my.data <- read_csv("G:/My Drive/_WashU_courses/2020_Spring/Coding/Assignments/Week_1/Expends2002.txt")
View(my.data)
# head(my.data)
# tail(my.data)
# install.packages("readr")
```

2. Change the variable name **TransID to **Useless**.**

```
library("tidyverse")
my.data <- my.data %>%
  rename(
    Useless = TransID
  )
```

3. Remove the variables **Useless, and **Source** from the dataframe.**

```
my.data = select(my.data, -3, -21)
ls(my.data)
```

```
##  [1] "Amount"       "Candid"       "City"         "CmtelD_EF"    "CRPFilerid"
##  [6] "CRPRecipname" "Cycle"        "Date"         "Descrip"      "ElecOther"
## [11] "EntType"      "Expcode"      "ID"           "Pacshort"     "PG"
## [16] "Recipcode"    "State"        "Type"         "Zip"
```

4. Change the variable **EntType** to a factor. How many levels does this variable have?

```
# Note that right now, the variable EntType is a character (string) variable
my.data <- my.data %>%
  mutate(EntType = as.factor(EntType))
class(my.data$EntType)
```

```
## [1] "factor"
```

```
levels(my.data$EntType)
```

```
## [1] "CAN" "CCM" "COM" "IND" "ORG" "PAC" "PTY"
```

As it can be seen above, there are seven levels of the factor variable **EntType**, excluding the missing values.

5. The variable **State** contains several obvious errors, as it includes non-existent state codes.

- Identify observations that have non-existent state codes.
- Write a script to recode these observations. Use the additional information in the dataset (candidate name, city, zip code) to correctly identify each state.

```
# Note that right now, the variable State is a character (string) variable.
#Let's change it to factor variable.
my.data <- my.data %>%
  mutate(State = as.factor(State))
levels(my.data$State)
```

```
##  [1] "AK" "AL" "AR" "AS" "AZ" "CA" "CO" "CT" "DC" "DE" "FL" "GA" "GU" "HI" "IA"
## [16] "ID" "IL" "IN" "KS" "KY" "LA" "LL" "MA" "MD" "ME" "MI" "MN" "MO" "MS" "MT"
## [31] "NC" "ND" "NE" "NH" "NJ" "NM" "NV" "NY" "OH" "OK" "OR" "PA" "RI" "SC" "SD"
## [46] "St" "TN" "TX" "UT" "VA" "VI" "VT" "WA" "WI" "WV" "WY" "ZZ"
```

See that "LL", "St", "ZZ" are invalid state codes. Let's see what are these:

```
print(my.data$Zip[(my.data$State == "St" | my.data$State == "LL" | my.data$State == "ZZ")
                  & is.na(my.data$State) == FALSE])
```

```
## [1] "NW"    "00000" "XXXXX"
```

```
print(my.data$City[(my.data$State == "St" | my.data$State == "LL" | my.data$State == "ZZ")
                   & is.na(my.data$State) == FALSE])
```

```
## [1] "1300 L"        "KINGSHILL VI"  "DCI/New Media"
```

```
print(my.data$CmtelD_EF[(my.data$State == "St" | my.data$State == "LL" | my.data$State == "ZZ")
                        & is.na(my.data$State) == FALSE])
```

```
## [1] NA          NA          "C00014498"
```

```
print(my.data$Pacshort[(my.data$State == "St" | my.data$State == "LL" | my.data$State == "ZZ")
                       & is.na(my.data$State) == FALSE])
```

```
## [1] "Thurman for Congress"     "Republican National Cmte"
## [3] "Republican Party of Iowa"
```

```
print(my.data$Recipcode[(my.data$State == "St" | my.data$State == "LL" | my.data$State == "ZZ")
                        & is.na(my.data$State) == FALSE])
```

```
## [1] "DL" "RP" "RP"
```

```
print(my.data$Cycle[(my.data$State == "St" | my.data$State == "LL" | my.data$State == "ZZ")
                    & is.na(my.data$State) == FALSE])
```

```
## [1] 2002 2002 2002
```

We see that there are three problematic cases. When we see other information, we see that:

1. First candidate (THURMAN, Karen L.) is from Florida (note that she ran in 2002, she has a campaign called "Thurman for Congress", and she is Democrat) and funded by an organized located in Washington DC.
2. Second candidate is from Virgin Islands (note that her address is KINGSHILL VI).

3. Third candidate is from Iowa (note that she is a candidate from the Republican Party of Iowa with ID = C00014498).

Let's recode accordingly:

```
# Note that right now, the variable State is a character (string) variable.
#Let's change it to factor variable.
# help(sub)
# Note that right now, the variable State is a character (string) variable.
#Let's change it to factor variable.
my.data <- my.data %>%
  mutate(State = sub(pattern = "St", replacement = "DC", State)) %>%
  mutate(State = sub(pattern = "LL", replacement = "VI", State)) %>%
  mutate(State = sub(pattern = "ZZ", replacement = "IA", State))
my.data <- my.data %>%
  mutate(State = as.factor(State))
levels(my.data$State)
```

```
##  [1] "AK" "AL" "AR" "AS" "AZ" "CA" "CO" "CT" "DC" "DE" "FL" "GA" "GU" "HI" "IA"
## [16] "ID" "IL" "IN" "KS" "KY" "LA" "MA" "MD" "ME" "MI" "MN" "MO" "MS" "MT" "NC"
## [31] "ND" "NE" "NH" "NJ" "NM" "NV" "NY" "OH" "OK" "OR" "PA" "RI" "SC" "SD" "TN"
## [46] "TX" "UT" "VA" "VI" "VT" "WA" "WI" "WV" "WY"
```

6. Remove all observations from the dataset where the variable **State** is missing. Report the number of observations after removing missing values.

```
# In the data, the missing in State is represented "  ".
levels(my.data$State)
```

```
##  [1] "AK" "AL" "AR" "AS" "AZ" "CA" "CO" "CT" "DC" "DE" "FL" "GA" "GU" "HI" "IA"
## [16] "ID" "IL" "IN" "KS" "KY" "LA" "MA" "MD" "ME" "MI" "MN" "MO" "MS" "MT" "NC"
## [31] "ND" "NE" "NH" "NJ" "NM" "NV" "NY" "OH" "OK" "OR" "PA" "RI" "SC" "SD" "TN"
## [46] "TX" "UT" "VA" "VI" "VT" "WA" "WI" "WV" "WY"
```

```
# Let's see how many missing observations there are.
sum(is.na(my.data$State))
```

```
## [1] 88
```

```
# So there are 88 missing values. Let's remove these.
my.data <- my.data %>%
  drop_na(State)

# Let's see how many observations there remained:
sum(!is.na(my.data$State))
```

```
## [1] 19912
```

As it can be seen above, 19912 observations are left.

7. Change the variable **Zip into a numeric. Be sure to document what you do with missing cases. What is the mean of this variable?**

```
# In the data, the missing in State is represented "         ".
sum(is.na(my.data$Zip))
```

```
## [1] 62
```

```
my.data <- my.data %>%
  mutate(Zip = as.numeric(Zip))

my.data$Zip[my.data$ID == 915857]
```

```
## [1] NA
```

```
mean(my.data$Zip, na.rm = TRUE)
```

```
## [1] 48214902
```

As it can be seen, the average Zip number is 48214902, which is 8 digit number. The reason is that some Zip codes are entered as 5 digit while other registered as 5 + 4 digits.

8. Create new variables that contain the following information (you will be making several variables), and answer the questions:

1. The number of words in the **Descrip** variable. What is the median value of this new variable?
2. A variable containing the numeric portion of **CRPFilerid**. This variable should be of length 8 for all observations. What is the number of unique values of this variable?
3. A vector containing the first four digits of **Zip**. What is the most frequent value of this vector?
4. A boolean indicating whether the **Descrip** variable contains the word "Communications" REGARDLESS OF CAPITALIZATION. Report the number of **TRUE** values in this boolean.
5. A variable indicating that either **CRPFilerid** is "N" or that BOTH Amount is greater than 500 and **Descrip** is non-missing. Report the number of TRUE values.
6. EXTRA CREDIT: A variable that provides the most common letter in the Descrip variable.

```
# The number of words in the **Descrip** variable. What is the median value of this new variable?
word.counter <- function(x){
  length(unlist(strsplit(as.character(x), "\\W+")))
}

my.data <- my.data %>%
  mutate(Descrip.length = sapply(Descrip, word.counter))

median(my.data$Descrip.length, na.rm = TRUE)
```

```
## [1] 2
```

```
sum(my.data$Descrip.length, na.rm = TRUE)
```

```
## [1] 61164
```

The median value of the length of Describ is 20.

```
# A variable containing the numeric portion of **CRPFilerid**.
# This variable should be of length 8 for all observations.
# What is the number of unique values of this variable?

my.data <- my.data %>%
  mutate(CRPFilerid.numeric = substr(CRPFilerid, 2, 9))

length(unique(my.data$CRPFilerid.numeric))
```

```
## [1] 2243
```

As it can be seen above, there are 2243 unique values.

```
# A vector containing the first four digits of **Zip**.
# What is the most frequent value of this vector?
my.data <- my.data %>%
  mutate(Zip.4digit = substr(Zip, 1, 4))

fun_mode <- function(x) {
  ux <- unique(x)
```

```
  ux[which.max(tabulate(match(x, ux)))]
}

fun_mode(my.data$Zip.4digit)
```

```
## [1] "2000"
```

```
length(my.data$Zip.4digit[my.data$Zip.4digit == 2000])
```

```
## [1] 1631
```

See that the most recurring ZIP values are "2000" and there are 1631 times of this value.

```
#  A boolean indicating whether the **Descrip** variable contains the word "Communications"
# REGARDLESS OF CAPITALIZATION. Report the number of **TRUE** values in this boolean.
my.data <- my.data %>%
  mutate(Descrip.commu.boolean = grepl("Communications", Descrip, ignore.case = TRUE))

sum(my.data$Descrip.commu.boolean)
```

```
## [1] 9
```

See that there are 9 cases of "Communications" regardless of capitalization in the Descrip variable.

```
#  A variable indicating that either **CRPFilerid** is
# 1) "N" or
# 2) that BOTH Amount is greater than 500 and **Descrip** is non-missing.
# Report the number of TRUE values.

# First let's create a new variable shows the first letter of **CRPFilerid**:
my.data <- my.data %>%
  mutate(CRPFilerid.first = substr(CRPFilerid, 1, 1))

# Now let's see the numbers individually:
sum(!is.na(my.data$Descrip)) # There are 18213 not missing in Descrip.
```

```
## [1] 18213
```

```
sum((my.data$Amount > 500) & is.na(my.data$Amount) == FALSE) # 9019 observations with **Amount** greate
```

```
## [1] 9019
```

```
sum((my.data$CRPFilerid.first == "N") & is.na(my.data$Amount) == FALSE)# 7247 observations starting wit
```

```
## [1] 7247
```

```
# Now let's create the variable of interest.
my.data <- my.data %>%
  mutate(Indicator = ((CRPFilerid.first == "N") | ((Amount > 500) & (!is.na(Descrip)))))

sum(my.data$Indicator)
```

```
## [1] 12392
```

See that there are 12392 TRUE values in our logical variable.

```r
#  EXTRA CREDIT: A variable that provides the most common letter in the Descrip variable.

Descrip.letters <- unlist(str_split(my.data$Descrip, ""))
Descrip.letters <- tolower(Descrip.letters)
Descrip.letters <- gsub(" " , NA, Descrip.letters)
Descrip.letters <- Descrip.letters[!is.na(Descrip.letters)]

fun_mode(Descrip.letters)
```

```
## [1] "e"
```

Notice thatthe most common character is “e”.

9. Write a script that subsets the data by state, and writes out a unique CSV file for each subset, where each file has a unique (and meaningful) name (hint: look at by() function).

```
# Creating a list of dataframes by state:
my.data.list <- by(my.data, INDICES=my.data$State, FUN = function(x) {my.data})

# Exporting all these different files by state:
lapply(1:length(my.data.list), function(i) write.csv(my.data.list[[i]],
                                        file = paste0(names(my.data.list[i]), ".csv"),
                                        row.names = FALSE))
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
## NULL
##
## [[7]]
## NULL
##
## [[8]]
## NULL
##
## [[9]]
## NULL
##
## [[10]]
## NULL
##
## [[11]]
## NULL
##
## [[12]]
## NULL
##
## [[13]]
## NULL
##
## [[14]]
## NULL
```

```
##
## [[15]]
## NULL
##
## [[16]]
## NULL
##
## [[17]]
## NULL
##
## [[18]]
## NULL
##
## [[19]]
## NULL
##
## [[20]]
## NULL
##
## [[21]]
## NULL
##
## [[22]]
## NULL
##
## [[23]]
## NULL
##
## [[24]]
## NULL
##
## [[25]]
## NULL
##
## [[26]]
## NULL
##
## [[27]]
## NULL
##
## [[28]]
## NULL
##
## [[29]]
## NULL
##
## [[30]]
## NULL
##
## [[31]]
## NULL
##
## [[32]]
## NULL
```

```
##
## [[33]]
## NULL
##
## [[34]]
## NULL
##
## [[35]]
## NULL
##
## [[36]]
## NULL
##
## [[37]]
## NULL
##
## [[38]]
## NULL
##
## [[39]]
## NULL
##
## [[40]]
## NULL
##
## [[41]]
## NULL
##
## [[42]]
## NULL
##
## [[43]]
## NULL
##
## [[44]]
## NULL
##
## [[45]]
## NULL
##
## [[46]]
## NULL
##
## [[47]]
## NULL
##
## [[48]]
## NULL
##
## [[49]]
## NULL
##
## [[50]]
## NULL
```

```
##
## [[51]]
## NULL
##
## [[52]]
## NULL
##
## [[53]]
## NULL
##
## [[54]]
## NULL
```