Problem Set 2

Due February 7, 2023

Instructions

- Read all of these instructions closely.
- This problem set is due Tuesday, February 7, 2023 at 4pm.
- Submit files via Github:
  1. the .Rmd (R Markdown) file
  2. the knitted .pdf file
  3. anything else the particular problem set might require
- Use a copy of this file, perhaps with your name or initials appended to the file name, to write your answers to the questions. You'll see there is a designated space where your answers should begin.
- Knitting the .Rmd file to a .pdf file *as you work* will ensure your code runs without errors and is working how you expect. Knit early and often. You've already read the instruction that a knitted .pdf is required when you submit.
- Per the syllabus, I will not accept any late work. Keep in mind the two lowest problem set scores are dropped. Turn in what you have.

Overview

This problem set uses a subset of expenditures data for all campaigns and PACs available from Open Secrets for 2002 cycle. The reduced dataset is available here. (While not the point of this question, I encourage you to visit the link to see how data shared on Dropbox can be imported directly into R via its url.)

Before you being this question, you should familiarize yourself with the variables. The codebook is available here.

```
expenditures_url <- "https://www.dropbox.com/s/z6gw9lvve6jogi5/Expends2002.txt?raw=1"
df <- read.csv(expenditures_url)
```

Question 1–Working with logicals

Use R code to answer the following questions.

1a

Are any `Amount` values missing?

```
#code here
```

Answer:

No, there are no `NA` values in the variable.

```
any(is.na(df$Amount))
```

```
## [1] FALSE
```

1b

How many observations are for refunds?

Hint: Read the codebook carefully for the `Amount` variable.

```
#code here
```

Answer:

276 observations are for refunds.

```
sum(df$Amount < 0)
```

```
## [1] 276
```

1c

What are the row indices for observations that indicate an amount spent of $1,000,000 or more?

```
#code here
```

Answer:

The following six rows spent 1million or more.

```
which(df$Amount >= 1000000)
```

```
## [1]  9169 14586 14868 14886 17290 17367
```

1d

Double check that all of the `Cycle` values equal 2002.

```
#code here
```

Answer:

Lots of ways to do this!

```
sum(df$Cycle != 2002)
```

```
## [1] 0
```

```
all(df$Cycle == 2002)
```

```
## [1] TRUE
```

```
unique(df$Cycle)
```

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

1e

How many observations are for "Club for Growth OR the"Madison Project" OR the "Republican National Cmte"?

```
#code here
```

Answer:

1337 observations are for expenditures for one of the three listed filing committees.

```
sum(df$Pacshort == "Club for Growth" |
      df$Pacshort == "Madison Project" |
      df$Pacshort == "Republican National Cmte")
```

```
## [1] 1337
```

```r
# more elegantly
sum(df$Pacshort %in% c("Club for Growth", "Madison Project", "Republican National Cmte"))
```

```
## [1] 1337
```

Question 2–Working with dataframes

2a

Using R functions, describe the following properties of the `df` object: class, dimensions, columnnames, rownnames, and anything else you think is pertinent.

```r
#code here
```

Answer:

There are multiple ways to answer this question. I used the `str()` function which tells me the object is a data.frame with 20,000 rows (observations) and 21 columns (variables). This function is handy because it also gives an overview of each variable's class. For example, I can see the `Cycle` variable is integer numeric, but the `Zip` is a character vector which I might have expected to also be numeric.

```r
str(df)
```

```
## 'data.frame':    20000 obs. of  21 variables:
##  $ Cycle      : int  2002 2002 2002 2002 2002 2002 2002 2002 2002 2002 ...
##  $ ID         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ TransID    : chr  "SB17.8045" "SB17.8024" "SB17.8004" "SB17.8027" ...
##  $ CRPFilerid : chr  "N00004887" "N00004887" "N00004887" "N00004887" ...
##  $ Recipcode  : chr  "DW" "DW" "DW" "DW" ...
##  $ Pacshort   : chr  "Citizens for Rush" "Citizens for Rush" "Citizens for Rush" "Citizens for Rush
##  $ CRPRecipname: chr  "ComEd" "David L.Andrukikis, Inc." "Hyde Leron" "Hyde Leron" ...
##  $ Expcode    : chr  "  " "  " "01" "01" ...
##  $ Amount     : int  323 306 6000 600 600 375 2545 500 541 510 ...
##  $ Date       : chr  "08/08/2001" "08/31/2001" "07/17/2001" "08/16/2001" ...
##  $ City       : chr  "Chicago" "Washington" "Chicago" "Chicago" ...
##  $ State      : chr  "IL" "DC" "IL" "IL" ...
##  $ Zip        : chr  "60600    " "20003    " "60617    " "60617    " ...
##  $ CmtelD_EF  : chr  "         " "         " "         " "         " ...
##  $ Candid     : chr  "" "" "" "" ...
##  $ Type       : chr  "" "" "" "" ...
##  $ Descrip    : chr  "Debt Reduction" "Printing - Invoice 11602" "Salary" "Salary" ...
##  $ PG         : chr  "P2000" "P2002" "P2002" "P2002" ...
##  $ ElecOther  : chr  "" "" "" "" ...
##  $ EntType    : chr  "ORG" "ORG" "IND" "IND" ...
##  $ Source     : chr  "" "" "@W02" "@W02" ...
```

2b

For the `TransID` variable, change its column name to `Useless_Var`.

Bonus: If you want to challenge yourself, try to write code that is flexible, meaning it will work correctly if `TransID` is the 3rd variable, 20th variable, or any position in the dataframe.

```r
#code here
```

Answer:

While I could "hard-code" in the third position index, a better way would be to keep the code flexible in case something happens and this isn't the third variable next time I use this dataset.

```
colnames(df)[colnames(df) == "TransID"] <- "Useless_Var"

# Also okay, but keep in mind we're trying to move beyond this:
#colnames(df)[3] <- "Useless_Var"
```

2c

Remove the variables `Useless_Var` and `Source` from the dataframe.

Bonus: Make this code flexible as well.

```
#code here
```

Answer:

```
# This solution stems from class materials:
rm_idx <- c(which(colnames(df) == "Useless_Var"),
            which(colnames(df) == "Source"))
df <- df[,-rm_idx]

# A more efficient line of code would use the %in%
# function, but we haven't covered that yet.
# rm_idx <- which(colnames(df) %in% c("Useless_Var", "Source"))
# df <- df[,-rm_idx]
```

2d

The variable `State` has many obvious errors. I've created the variable `StateWrong` with `NA` placeholders. Recode `StateWrong` to be `TRUE` if the `State` variable contains an error or a missing value, and `FALSE` otherwise.

Hint: We did a recoding exercise in the inclass activity.

Bonus: Try to use the `%in%` function. We haven't used it in class yet. It is similar to `==`. The syntax is `x %in% y`, which assesses each value of vector `x` and asks, is it equal to any of the values in vector `y`? I've included a simple example below.

```
df$StateWrong <- NA

# Example of %in%
# In words: For each letter in the alphabet,
# check if it is it equal (TRUE) or not (FALSE)
# to A, D, or F
x <- LETTERS
y <- c("A", "D", "F")
x %in% y
```

```
##  [1]  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE
```

Answer:

```
sort(table(df$State))
```

```
##
##   LL   St   VI   ZZ   AS   GU   VT   ID   AK   WY   ND   RI   MT   NE        HI
##     1     1     1     1     4     7    28    43    44    48    59    71    75    76    88    94
##    DE    KS    MS    ME    OR    SC    SD    WV    NH    NV    WI    CT    WA    UT    NM    OK
##   115   117   117   140   140   146   150   153   157   184   200   202   204   206   220   226
##    AR    AL    MA    KY    CO    IA    NC    MO    LA    AZ    IN    GA    TN    MN    OH    NJ
##   262   269   282   306   336   347   366   386   394   397   414   448   489   543   594   607
##    MI    MD    FL    IL    NY    PA    TX    VA    CA    DC
##   624   629   649   717   741   888  1170  1322  1414  2088
```

```
any(is.na(df$State))
```

```
## [1] FALSE
```

```
bad_state_vals <- c("  ", "St", "ZZ", "LL", "VI", "AS", "GU")
df$StateWrong[df$State %in% bad_state_vals] <- TRUE
df$StateWrong[!df$State %in% bad_state_vals] <- FALSE
# double check 50 states + DC
length(unique(df$State[!df$StateWrong]))
```

```
## [1] 51
```

2e

Using the `StateWrong` variable, report how many observations in the dataset have a wrong or missing value. Then remove these observations. Confirm that you've removed the correct number of rows by checking the dimensions of the data.

```
#code here
```

Answer:

91 observations are incorrect or missing

```
# Lots of ways to report number
table(df$StateWrong)
```

```
##
## FALSE  TRUE
## 19897   103
```

```
sum(df$StateWrong)
```

```
## [1] 103
```

```
# remove observations
df <- df[!df$StateWrong, ]

# new dimensions less 91 rows
dim(df)
```

```
## [1] 19897    20
```

2f

Create the variable in the dataframe called `Payroll`. It should be a logical indicating whether the `Descrip` variable contains the string "payroll" *regardless* of capitalization. Report the number of `TRUE` values in this variable.

Hint: Use the grepl function and read the helpfile closely.

```
#code here
```

Answer:

“payroll” appears in 466 observations.

```
df$Payroll <- grepl(pattern = "payroll", x = df$Descrip, ignore.case = T)
sum(df$Payroll)
```

```
## [1] 462
```

2g

Write a function named `sum_state_exp` that takes one character argument called `state_code`. The function should return the mean amount of expenditures in given state.

```
# Write function

# After writing the function, run it for IA, IL, and CA
# sum_state_exp(state_code = "IA")
# sum_state_exp(state_code = "IL")
# sum_state_exp(state_code = "CA")
```

Answer:

```
sum_state_exp <- function(state_code){
  sum(df$Amount[df$State == state_code])
}

sum_state_exp(state_code = "IA")
```

```
## [1] 478163
```

```
sum_state_exp(state_code = "IL")
```

```
## [1] 1165688
```

```
sum_state_exp(state_code = "CA")
```

```
## [1] 1994622
```