Applied Statistical Programming - Spring 2022

Alma Velazquez

Problem Set 1

Due Wednesday, February 2, 10:00 AM (Before Class)

Instructions

1. The following questions should each be answered in this Rmarkdown document with R code to accompany the R output. Be sure to provide many comments in the script to facilitate grading. Undocumented code will not be graded. Once your work is finished, submit the Rmd file as well as the knitted PDF to the appropriate problem set module on Canvas.
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 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: a) The column names should match the column names in the original dataset. b) The row names should correspond to the variable `ID` in the original dataset.

```r
# Read the dropbox file as a URL; set row names equal to the 'ID' column
expenditures <- read.csv("https://www.dropbox.com/s/z6gw9lvve6jogi5/Expends2002.txt?raw=1",
    row.names = "ID")

# Compare column names to the names in the codebook
names(expenditures)
```

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

```r
# Recreate the 'ID' column to ensure names match
expenditures$ID <- rownames(expenditures)
```

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

```r
# Using the which() function, assign the name of the index where the df is
# named 'TransID' to now be 'Useless'
names(expenditures)[which(names(expenditures) == "TransID")] <- "Useless"
```

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

```r
# Assign each undesired column to NULL to remove
expenditures$Useless <- NULL
expenditures$Source <- NULL
```

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

```r
# Assign the column to itself, wrapped in as.factor() to perform the conversion
expenditures$EntType <- as.factor(expenditures$EntType)

# Check it worked, see how many levels there are
str(expenditures$EntType)
```

```
##  Factor w/ 8 levels "","CAN","CCM",..: 6 6 5 5 5 1 1 8 6 4 ...
```

```r
# View the levels themselves
levels(expenditures$EntType)
```

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

There are 8 levels.

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

a. Identify observations that have non-existent state codes.

```r
# View unique state codes to find non-existent ones
unique(expenditures$State)
```

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

```r
# Create a vector of row indices for observations that take on these values
# Left blanks alone for now since they are covered in a later question
non_ext <- which(expenditures$State %in% c("ZZ", "LL", "St"))

# View this subset of observations (only returning selected columns to save
# space)
expenditures[non_ext, c(1:3, which(colnames(expenditures) == "State"))]
```

```
##        Cycle CRPFilerid Recipcode State
## 368552  2002  N00002973        DL    St
## 634397  2002  C00003418        RP    ZZ
## 162479  2002  C00014498        RP    LL
```

b. Write a script to recode these observations. Use the additional information in the dataset (candidate name, city, zip code) to correctly identify each state.

```
# See how many values are in this vector of indices; next correct each
length(non_ext)
```

```
## [1] 3
```

```
# Based on information in the other columns, assign each nonexistent state code
# to its correct value
expenditures[non_ext[1], ]$State <- "FL"
expenditures[non_ext[2], ]$State <- "VI"
expenditures[non_ext[3], ]$State <- "IA"

# View subset again
expenditures[non_ext, c(1:3, which(colnames(expenditures) == "State"))]
```

```
##        Cycle CRPFilerid Recipcode State
## 368552  2002  N00002973        DL    FL
## 634397  2002  C00003418        RP    VI
## 162479  2002  C00014498        RP    IA
```

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

```
# Now address the blanks. First recode them as NA
expenditures[which(expenditures$State == "  "), ]$State <- NA

# Use assignment and the is.na() function to eliminate these rows
expenditures <- expenditures[!is.na(expenditures$State), ]

# View new number of observations
nrow(expenditures)
```

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

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?

```
# First spot check unique values to avoid inducing NAs by coercion with
# as.numeric() unique(expenditures$Zip)

# Correct potential problems: 1) Recode blanks as NA for easier manipulation
expenditures[which(expenditures$Zip == "         "), ]$Zip <- NA

# 2) Remove hyphens and any other non-numeric characters using a regular
# expression and gsub()
expenditures$Zip <- gsub("[^0-9]", "", expenditures$Zip)

# Finally make the conversion - note there are no warnings, so we are sure
# nothing was incorrectly converted
expenditures$Zip <- as.numeric(expenditures$Zip)

# Recall that we still have NAs
length(expenditures$Zip[which(is.na(expenditures$Zip))])
```

```
## [1] 65
```

```
# Calculate the mean of the column, excluding NA values
mean(expenditures$Zip, na.rm = TRUE)
```

```
## [1] 48224223
```

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

a. The number of words in the `Descrip` variable. What is the median value of this new variable?

```
# First get a sense of the Descrip character strings str(expenditures$Descrip)
# expenditures$Descrip[grepl('[^A-Za-z ]', expenditures$Descrip)]
# tail(expenditures$Descrip[grepl(' \\- |\\/|\\^|\\-', expenditures$Descrip)])

# Attempt to clean up some characters that could affect the word count, defined
# by strings separated by a single space

# 1) Some strings contained words separated by ' - ', '/', '-', ':', address
# these using a regular expression
expenditures$Descrip <- trimws(gsub(" \\- |\\/|\\^|\\-|\\:", " ", expenditures$Descrip))

# 2) Some strings contained words separated by more than 1 space; address this
# using a regular expression
expenditures$Descrip <- gsub("\\s+", " ", expenditures$Descrip)

# 3) Turn empty strings to NA
expenditures[which(expenditures$Descrip == ""), ]$Descrip <- NA

# Perform the string split by single space Since this returns a list, use
# lengths() (not length()) to count its contents
expenditures$N_Descrip <- lengths(strsplit(expenditures$Descrip, " "))

# Take median, taking care to remove NA values
median(expenditures$N_Descrip, na.rm = TRUE)
```

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

b. 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?

```
# Use regular expression and gsub() to eliminate non-numeric characters; assign
# to new column
expenditures$Num_CRPFilerid <- gsub("[^0-9]", "", expenditures$CRPFilerid)


# Check variable is length 8 for all observations
unique(nchar(expenditures$Num_CRPFilerid))
```

```
## [1] 8
```

```
# Use length(unique()) to count unique values
length(unique(expenditures$Num_CRPFilerid))
```

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

c. A vector containing the first four digits of `Zip`. What is the most frequent value of this vector?

```
# Turn the vector to character to use substr(); turn back to numeric
expenditures$Zip <- as.numeric(substr(as.character(expenditures$Zip), 1, 4))
```

```r
# The table function returns a count of each unique value; as.numeric()
# accesses its counts, and names() accesses the value being counted This line
# gives us the most frequently appearing value with its count
table(expenditures$Zip)[as.numeric(table(expenditures$Zip)) == max(as.numeric(table(expenditures$Zip)))]
```

```
## 2000
## 1561
```

```r
# This line simply returns that value
names(table(expenditures$Zip)[as.numeric(table(expenditures$Zip)) == max(as.numeric(table(expenditures$
```

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

This tells us the value 2000, occurring 1561 times, is the mode of this vector.

d. A boolean indicating whether the `Descrip` variable contains the word "Communications'' REGARDLESS OF CAPITALIZATION. Report the number of `TRUE` values in this boolean.

```r
# grepl() returns a logical vector; we first make everything lowercase to avoid
# cases affecting matches
expenditures$Comm_Descrip <- grepl("communications", tolower(expenditures$Descrip))

# sum() gives a count of TRUEs since TRUE = 1
sum(expenditures$Comm_Descrip)
```

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

e. 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.

```r
# Define the new variable using a compound logical test, with plenty of
# parentheses to specify what we want
expenditures$Logical_Test <- (startsWith(expenditures$CRPFilerid, "N")) | ((expenditures$Amount >
    500) & (!is.na(expenditures$Descrip)))

# Use sum() to count TRUEs again
sum(expenditures$Logical_Test)
```

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

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

```r
# Interpreting this as: for each OBSERVATION, what is the most common letter in
# the Descrip variable.  If we wanted the most common letter over the whole
# vector, it would look very much like 8c.

# First get rid of any character that isn't a letter; save the modified string
# a new variable
expenditures$Mode_Descrip <- strsplit(tolower(gsub("[^A-Za-z]", "", expenditures$Descrip)),
    "")

# The following works around the fact that strsplit() returns a list.

# First we define a function meant to operate on on a per-observation list
table_max <- function(x) {
    # We want to table all the letters in this observation
    tab <- unlist(table(x))
    # Recall that we had NAs; we want these to keep being NAs
    if (length(tab) == 0) {
        return(NA)
        # For those that aren't NA, return the values with the highest count in
        # this observantion's Descrip
    } else {
        tab <- names(tab)[as.numeric(tab) == max(as.numeric(tab))]
        # Paste with collapse was added to make transforming to a character
        # vector easier after This will return a string of 1 or more letters
        # that had the max count per observation
        return(paste0(tab, collapse = ""))
    }
}

# Apply to each observation using lapply() (recall the variable is currently a
# list); wrap in as.character() so we can export to csv later
expenditures$Mode_Descrip <- as.character(lapply(expenditures$Mode_Descrip, FUN = table_max))
```

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).

```
# Make State a factor to pass to the INDICES argument of by()
expenditures$State <- as.factor(expenditures$State)

# Write a function that creates a filename for each subset and writes to csv
write_by_chunk <- function(chunk) {
    st <- max(as.character(chunk$State))
    fn <- paste0("expenditures_", st, ".csv")
    write.csv(chunk, fn, row.names = FALSE)
    return(paste0(st, " has written to csv"))
}

# Execute the function by State
by(expenditures, expenditures$State, write_by_chunk)
```

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