IST 387 HW 3


```
# Enter your name here: Ezra Cohen
```

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this homework by myself, with help from the book and the professor.
```

Reminders of things to practice from last week:

Make a data frame data.frame( )
Row index of max/min which.max( ) which.min( )
Sort value or order rows sort( ) order( )
Descriptive statistics mean( ) sum( ) max( )
Conditional statement if (condition) "true stuff" else "false stuff"

This Week:

Often, when you get a dataset, it is not in the format you want. You can (and should) use code to refine the dataset to become more useful. As Chapter 6 of Introduction to Data Science mentions, this is called "data munging." In this homework, you will read in a dataset from the web and work on it (in a data frame) to improve its usefulness.

Part 1: Use read_csv( ) to read a CSV file from the web into a data frame:

A. Use R code to read directly from a URL on the web. Store the dataset into a new dataframe, called dfComps.
The URL is:
"https://ist387.s3.us-east-2.amazonaws.com/data/Companies.csv"
Hint: use read_csv( ), not read.csv( ). This is from the **tidyverse package**. Check the help to compare them.

```
read_csv(url("https://ist387.s3.us-east-2.amazonaws.com/data/Companies.csv"))
```

Part 2: Create a new data frame that only contains companies with a homepage URL:

B. Use View( ), head( ), and tail( ) to examine the **dfComps** dataframe.
Add a block comment that briefly describes what you see.

```
dfComps<-data.frame(read_csv(url("https://ist387.s3.us-east-2.amazonaws.com/data/Companies.csv")))
View(dfComps)
head(dfComps)
tail(dfComps)
#dfComps Has data on a bunch of different companies, including what area of the market they deal with, links to t
heir websites, where they are located, funds and more
```

C. Create a variable (called **noURL**) that has a value of **TRUE** if a company is missing a homepage URL.

```
dfComps$noURL <- is.na(dfComps$homepage_url)
```

D. Use the table( ) command to summarize the contents of **noURL**.
Write a comment interpreting what you see – how many companies are missing a homepage URL?

```
table(dfComps$noURL)
#there are 3323 companies without a homepage and 44435 with one
```

E. Use **subsetting** to create a new dataframe that contains only the companies with homepage URLs (store that dataframe in **urlComps**).

```
urlComps <- data.frame(dfComps[-c(which(dfComps$noURL==TRUE)),])
View(urlComps)
```

F. Use the dim() command on **urlComps** to confirm that the data frame contains **44,435** observations and **19** columns/variables.

```
dim(urlComps)
```

Part 3: Analyze the numeric variables in the dataframe.

G. How many **numeric variables** does the dataframe have? You can figure that out by looking at the output of **str(urlComps)**.

```
str(urlComps)
# funding_rounds, and founded_year are numeric
totalnumeric <-length(urlComps$funding_rounds) + length(urlComps$funding_year)
totalnumeric
```

H. What is the average number of funding rounds for the companies in **urlComps**?

```
mean(urlComps$funding_rounds)
```

I. What year was the oldest company in the dataframe founded?
Hint: If you get a value of "NA," most likely there are missing values in this variable which preclude R from properly calculating the min & max values. Instead of running, for example, mean(urlComps$founded_year), something like this will work for determining the average:

```
#mean(urlComps$founded_year, na.rm=TRUE)
#Modify the code above to find the oldest company in the df.
min(urlComps$founded_year, na.rm=TRUE)
```

```
Error in mean(urlComps$founded_year, na.rm = TRUE): object 'urlComps' not found
Traceback:


1. mean(urlComps$founded_year, na.rm = TRUE)
```

J. Create another dataframe containing the companies that do not have homepage URLs. Find out the mean number of funding rounds for those companies. Compare that to the answer you recorded for problem H.

```
nourlComps <- data.frame(dfComps[-c(which(dfComps$noURL==FALSE)),])
View(nourlComps)
mean(urlComps$funding_rounds)
```

Part 4: Use gsub() to clean the data.

K. The **permalink variable** in **urlComps** contains the name of each company but the names are currently preceded by the prefix "/organization/". We can use gsub() to clean the values of this variable:

```
urlComps$company <- gsub("/organization/", "", urlComps$permalink)
#Write a comment explaining what this line of code does.
#gets rid of /organization/ by turning it into nothing in the company column
```

```
Error in gsub("/organization/", "", urlComps$permalink): object 'urlComps' not found
Traceback:


1. gsub("/organization/", "", urlComps$permalink)
```

L. Can you identify another variable which should be numeric but is currently coded as character? Use the as.numeric() function to add a new variable to **urlComps** which contains the values from the char variable as numbers. Do you notice anything about the number of NA values in this new column compared to the original "char" one?

```
str(urlComps)
#funding_total_usd should be numeric
match(TRUE,is.na(urlComps$funding_total_usd))
urlComps$funding_total_usd_numeric <- as.numeric(urlComps$funding_total_usd)
match(FALSE,is.na(urlComps$funding_total_usd_numeric))
#it increased by a lot
```

M. To ensure the char values are converted correctly, we first need to remove the spaces between the numbers in the variable. Use the gsub() command to do that. Check if this works:

```
urlComps$funding_new <- gsub("\\s","", urlComps$funding_total_usd)
#Does this variable look better than the one we created in L? Explain in a comment:
#yes, there are many more numbers rather than just NA's
```

```
Error in gsub("\\s", "", urlComps$funding_total_usd): object 'urlComps' not found
Traceback:


1. gsub("\\s", "", urlComps$funding_total_usd)
```

N. You are now ready to convert urlComps$funding_new to numeric using as.numeric() again. Calculate the average funding amount for **urlComps**. If you get "NA," try using the na.rm=TRUE argument from problem I.

```
mean(as.numeric(urlComps$funding_new), na.rm=TRUE)
```

Part 5: Create a function to automate the process from L-N:

O. The following function should work most of the time. Make sure to run this code before trying to test it. That is how you make the new function known to R. **Add comments to each line explaining what it does:**

```
convertCharToNum <- function(char_string) {
#names the function and names what the input should be
  step1 <- gsub("\\s","", char_string)
  #gets rid of all spaces
  step2 <- as.numeric(step1)
  #turns the new string made in step 1 from characters into numbers
  return(step2)
  #returns the result of step 2
}
```

P. Run your new function on the **funding_total_usd** variable in **urlComps**:

```
convertCharToNum(urlComps$funding_total_usd)
```

```
Error in gsub("\\s", "", char_string): object 'urlComps' not found
Traceback:


1. convertCharToNum(urlComps$funding_total_usd)

2. gsub("\\s", "", char_string)   # at line 2 of file <text>
```

Q. Assign the result of P to a variable in the dataframe:

```
urlComps$funding_total_num <-  convertCharToNum(urlComps$funding_total_usd)
```

```
Error in gsub("\\s", "", char_string): object 'urlComps' not found
Traceback:


1. convertCharToNum(urlComps$funding_total_usd)

2. gsub("\\s", "", char_string)   # at line 2 of file <text>
```

Calculate the average of this new variable (you may need to use the rm.na=TRUE argument again). Is it the same as the value you got in N? Explain.

```
mean(urlComps$funding_total_num, na.rm=TRUE)
```