# Intro to Data Science HW 3

##### Copyright Jeffrey Stanton, Jeffrey Saltz, and Jasmina Tacheva

# Rutwik Ghag

### 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:

1. 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://intro-datascience.s3.us-east-2.amazonaws.com/companies1.csv>” **Hint:** use read\_csv( ), not read.csv( ). This is from the **tidyverse package**. Check the help to compare them.

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✓ ggplot2 3.3.5 ✓ purrr 0.3.4  
## ✓ tibble 3.1.4 ✓ dplyr 1.0.7  
## ✓ tidyr 1.1.3 ✓ stringr 1.4.0  
## ✓ readr 2.0.1 ✓ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

file <- "https://intro-datascience.s3.us-east-2.amazonaws.com/companies1.csv" # As taught in class  
help("read\_csv")  
dfComps <- read\_csv(file)

## Rows: 47758 Columns: 18

## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (16): permalink, name, homepage\_url, category\_list, market, funding\_tota...  
## dbl (2): funding\_rounds, founded\_year

##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

dfComps

## # A tibble: 47,758 × 18  
## permalink name homepage\_url category\_list market funding\_total\_u… status  
## <chr> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 /organiza… #wayw… http://www.w… |Entertainmen… News 1 750 000 acqui…  
## 2 /organiza… &TV C… http://enjoy… |Games| Games 4 000 000 opera…  
## 3 /organiza… 'Rock… http://www.r… |Publishing|E… Publi… 40 000 opera…  
## 4 /organiza… (In)T… http://www.I… |Electronics|… Elect… 1 500 000 opera…  
## 5 /organiza… #NAME? http://plusn… |Software| Softw… 1 200 000 opera…  
## 6 /organiza… -R- R… <NA> |Entertainmen… Games 10 000 opera…  
## 7 /organiza… .Club… http://nic.c… |Software| Softw… 7 000 000 <NA>   
## 8 /organiza… .Fox … http://www.d… |Advertising| Adver… 4 912 393 closed  
## 9 /organiza… 0-6.c… http://www.0… |Curated Web| Curat… 2 000 000 opera…  
## 10 /organiza… 004 T… http://004gm… |Software| Softw… - opera…  
## # … with 47,748 more rows, and 11 more variables: country\_code <chr>,  
## # state\_code <chr>, region <chr>, city <chr>, funding\_rounds <dbl>,  
## # founded\_at <chr>, founded\_month <chr>, founded\_quarter <chr>,  
## # founded\_year <dbl>, first\_funding\_at <chr>, last\_funding\_at <chr>

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

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

view(dfComps)  
head(dfComps)

## # A tibble: 6 × 18  
## permalink name homepage\_url category\_list market funding\_total\_u… status  
## <chr> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 /organiza… #wayw… http://www.w… |Entertainment… News 1 750 000 acqui…  
## 2 /organiza… &TV C… http://enjoy… |Games| Games 4 000 000 opera…  
## 3 /organiza… 'Rock… http://www.r… |Publishing|Ed… Publi… 40 000 opera…  
## 4 /organiza… (In)T… http://www.I… |Electronics|G… Elect… 1 500 000 opera…  
## 5 /organiza… #NAME? http://plusn… |Software| Softw… 1 200 000 opera…  
## 6 /organiza… -R- R… <NA> |Entertainment… Games 10 000 opera…  
## # … with 11 more variables: country\_code <chr>, state\_code <chr>, region <chr>,  
## # city <chr>, funding\_rounds <dbl>, founded\_at <chr>, founded\_month <chr>,  
## # founded\_quarter <chr>, founded\_year <dbl>, first\_funding\_at <chr>,  
## # last\_funding\_at <chr>

tail(dfComps)

## # A tibble: 6 × 18  
## permalink name homepage\_url category\_list market funding\_total\_u… status  
## <chr> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 /organizat… Zytop… http://www.z… |Biotechnolog… Biote… 2 686 600 opera…  
## 2 /organizat… Zzish http://www.z… |Analytics|Ga… Educa… 320 000 opera…  
## 3 /organizat… ZZNod… http://www.z… |Enterprise S… Enter… 1 587 301 opera…  
## 4 /organizat… Zzzza… http://www.z… |Web Developm… Web D… 97 398 opera…  
## 5 /organizat… [a]li… http://www.a… |Games| Games 9 300 000 opera…  
## 6 /organizat… [x+1] http://www.x… |Enterprise S… Enter… 45 000 000 opera…  
## # … with 11 more variables: country\_code <chr>, state\_code <chr>, region <chr>,  
## # city <chr>, funding\_rounds <dbl>, founded\_at <chr>, founded\_month <chr>,  
## # founded\_quarter <chr>, founded\_year <dbl>, first\_funding\_at <chr>,  
## # last\_funding\_at <chr>

#we see a data frame of organizations in the US along with their details like URLs, last funding, etc

1. Create a variable (called **noURL**) that has a value of **TRUE** if a company is missing a homepage URL. This variable should be a part of **dfComps**, not just a standalone vector.

dfComps$noURL <- ifelse(is.na(dfComps$homepage\_url), TRUE, FALSE)   
view(dfComps)

1. 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)

##   
## FALSE TRUE   
## 44435 3323

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

urlComps <- dfComps[dfComps$noURL == FALSE,]

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

dim(urlComps)

## [1] 44435 19

## Part 3: Analyze the numeric variables in the dataframe.

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

str(urlComps)

## tibble [44,435 × 19] (S3: tbl\_df/tbl/data.frame)  
## $ permalink : chr [1:44435] "/organization/waywire" "/organization/tv-communications" "/organization/rock-your-paper" "/organization/in-touch-network" ...  
## $ name : chr [1:44435] "#waywire" "&TV Communications" "'Rock' Your Paper" "(In)Touch Network" ...  
## $ homepage\_url : chr [1:44435] "http://www.waywire.com" "http://enjoyandtv.com" "http://www.rockyourpaper.org" "http://www.InTouchNetwork.com" ...  
## $ category\_list : chr [1:44435] "|Entertainment|Politics|Social Media|News|" "|Games|" "|Publishing|Education|" "|Electronics|Guides|Coffee|Restaurants|Music|iPhone|Apps|Mobile|iOS|E-Commerce|" ...  
## $ market : chr [1:44435] "News" "Games" "Publishing" "Electronics" ...  
## $ funding\_total\_usd: chr [1:44435] "1 750 000" "4 000 000" "40 000" "1 500 000" ...  
## $ status : chr [1:44435] "acquired" "operating" "operating" "operating" ...  
## $ country\_code : chr [1:44435] "USA" "USA" "EST" "GBR" ...  
## $ state\_code : chr [1:44435] "NY" "CA" NA NA ...  
## $ region : chr [1:44435] "New York City" "Los Angeles" "Tallinn" "London" ...  
## $ city : chr [1:44435] "New York" "Los Angeles" "Tallinn" "London" ...  
## $ funding\_rounds : num [1:44435] 1 2 1 1 2 1 1 1 1 1 ...  
## $ founded\_at : chr [1:44435] "1/6/12" NA "26/10/2012" "1/4/11" ...  
## $ founded\_month : chr [1:44435] "2012-06" NA "2012-10" "2011-04" ...  
## $ founded\_quarter : chr [1:44435] "2012-Q2" NA "2012-Q4" "2011-Q2" ...  
## $ founded\_year : num [1:44435] 2012 NA 2012 2011 2012 ...  
## $ first\_funding\_at : chr [1:44435] "30/06/2012" "4/6/10" "9/8/12" "1/4/11" ...  
## $ last\_funding\_at : chr [1:44435] "30/06/2012" "23/09/2010" "9/8/12" "1/4/11" ...  
## $ noURL : logi [1:44435] FALSE FALSE FALSE FALSE FALSE FALSE ...

#There are 2 numeric variables

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

mean(urlComps$funding\_rounds)

## [1] 1.725194

1. 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)

## [1] 2007.289

Error in mean(urlComps$founded\_year, na.rm = TRUE): object 'urlComps' not found  
Traceback:  
  
  
1. mean(urlComps$founded\_year, na.rm = TRUE)

Now write the code to get the oldest company

min(urlComps$founded\_year, na.rm=TRUE)

## [1] 1900

1. 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.

newdfComps <- dfComps[dfComps$noURL == TRUE,]  
mean(newdfComps$funding\_rounds)

## [1] 1.198917

## Part 4: Use string operations to clean the data.

1. 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() or str\_replace (from tidyverse) to clean the values of this variable:

urlComps$company <- gsub("/organization/", "", urlComps$permalink)  
  
library(tidyverse)  
#write the code to do the same cleanup, but with str\_replace from tidyverse  
  
urlComps$company <- str\_replace\_all(urlComps$permalink, "/organization/", "")  
head(urlComps)

## # A tibble: 6 × 20  
## permalink name homepage\_url category\_list market funding\_total\_u… status  
## <chr> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 /organiza… #wayw… http://www.w… |Entertainment… News 1 750 000 acqui…  
## 2 /organiza… &TV C… http://enjoy… |Games| Games 4 000 000 opera…  
## 3 /organiza… 'Rock… http://www.r… |Publishing|Ed… Publi… 40 000 opera…  
## 4 /organiza… (In)T… http://www.I… |Electronics|G… Elect… 1 500 000 opera…  
## 5 /organiza… #NAME? http://plusn… |Software| Softw… 1 200 000 opera…  
## 6 /organiza… .Club… http://nic.c… |Software| Softw… 7 000 000 <NA>   
## # … with 13 more variables: country\_code <chr>, state\_code <chr>, region <chr>,  
## # city <chr>, funding\_rounds <dbl>, founded\_at <chr>, founded\_month <chr>,  
## # founded\_quarter <chr>, founded\_year <dbl>, first\_funding\_at <chr>,  
## # last\_funding\_at <chr>, noURL <lgl>, company <chr>

1. 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?

as.numeric(urlComps$funding\_total\_usd)[1:50]

## Warning: NAs introduced by coercion

## [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA  
## [26] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA

# The output is NA as we have spaces there

1. To ensure the char values are converted correctly, we first need to remove the spaces between the digits in the variable. Check if this works:

library(stringi)  
urlComps$funding\_new <- stri\_replace\_all\_charclass(urlComps$funding\_total\_usd,"\\p{WHITE\_SPACE}", "")  
#urlComps

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)

## Warning in mean(as.numeric(urlComps$funding\_new), na.rm = TRUE): NAs introduced  
## by coercion

## [1] 18321551

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

1. 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:**

library(stringi)  
convertCharToNum <- function(char\_string) { # Initialize a function to convert numbers to numbers  
 step1 <- stri\_replace\_all\_charclass(char\_string,"\\p{WHITE\_SPACE}", "") # Remover whitespace  
 step2 <- as.numeric(step1) # Converts them to numeric and stores the new series into step2  
 return(step2)  
}

1. Run your new function on the **funding\_total\_usd** variable in **urlComps**:

convertCharToNum(urlComps$funding\_total\_usd)[1:50]

## Warning in convertCharToNum(urlComps$funding\_total\_usd): NAs introduced by  
## coercion

## [1] 1750000 4000000 40000 1500000 1200000 7000000 4912393 2000000  
## [9] NA 41250 10600000 40000 NA 1750000 2050000 40000  
## [17] 500000 NA 2535000 4962651 4059079 10000000 3000000 3000000  
## [25] 1250000 35000000 50000 1600000 100000 NA 3000000 77500  
## [33] NA 20000000 1800000 NA NA 270000 794000 650267  
## [41] NA NA 378812 130636 619494 NA 66450000 NA  
## [49] 700000 5500000

Q. Create a new function, that does the same functionality as ‘convertCharToNum’, but uses tidyverse stringr commands

library(tidyverse)  
convertCharToNumNew <- function(char\_string) {  
 step1 <- str\_replace\_all(char\_string, pattern = "\\p{WHITE\_SPACE}", repl="")  
 step2 <- as.numeric(step1)  
 return(step2)  
}

#test the new function  
convertCharToNumNew(urlComps$funding\_total\_usd)[1:50]

## Warning in convertCharToNumNew(urlComps$funding\_total\_usd): NAs introduced by  
## coercion

## [1] 1750000 4000000 40000 1500000 1200000 7000000 4912393 2000000  
## [9] NA 41250 10600000 40000 NA 1750000 2050000 40000  
## [17] 500000 NA 2535000 4962651 4059079 10000000 3000000 3000000  
## [25] 1250000 35000000 50000 1600000 100000 NA 3000000 77500  
## [33] NA 20000000 1800000 NA NA 270000 794000 650267  
## [41] NA NA 378812 130636 619494 NA 66450000 NA  
## [49] 700000 5500000

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

urlComps$result <- convertCharToNumNew(urlComps$funding\_total\_usd)

## Warning in convertCharToNumNew(urlComps$funding\_total\_usd): NAs introduced by  
## coercion

S. 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$result, na.rm = TRUE)

## [1] 18321551

#The values are the same