Intro to Data Science HW 3

```
# Enter your name here: Chaithra Kopparam Cheluvaiah
```


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://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 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
dfComps <- read_csv("https://intro-datascience.s3.us-east-2.amazonaws.com/companies1.csv")
```

```
## 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.
```

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.**

```
View(dfComps)  # lists all the rows in data frame
head(dfComps)  # lists first few rows in data frame
```

```
## # A tibble: 6 x 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)  # lists last few rows in data frame
```

```
## # A tibble: 6 x 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>
```

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

```
##
## FALSE  TRUE
## 44435  3323
```

```
# 3323 companies are missing home url
```

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[ !dfComps$noURL, ])
View(urlComps)

library(tidyverse)
urlComps <- dfComps%>%filter(!noURL)
```

F. 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.

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

```
str(urlComps)  # There are 2 numeric variables - funding_rounds and founded_year
```

```
## spec_tbl_df[,19] [44,435 x 19] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ permalink        : chr [1:44435] "/organization/waywire" "/organization/tv-communications" "/orga
##  $ name             : chr [1:44435] "#waywire" "&TV Communications" "'Rock' Your Paper" "(In)Touch N
##  $ homepage_url     : chr [1:44435] "http://www.waywire.com" "http://enjoyandtv.com" "http://www.roc
##  $ category_list    : chr [1:44435] "|Entertainment|Politics|Social Media|News|" "|Games|" "|Publish
##  $ 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 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   permalink = col_character(),
##   ..   name = col_character(),
##   ..   homepage_url = col_character(),
##   ..   category_list = col_character(),
##   ..   market = col_character(),
##   ..   funding_total_usd = col_character(),
##   ..   status = col_character(),
##   ..   country_code = col_character(),
##   ..   state_code = col_character(),
##   ..   region = col_character(),
##   ..   city = col_character(),
##   ..   funding_rounds = col_double(),
##   ..   founded_at = col_character(),
##   ..   founded_month = col_character(),
##   ..   founded_quarter = col_character(),
##   ..   founded_year = col_double(),
##   ..   first_funding_at = col_character(),
##   ..   last_funding_at = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
library(dplyr)
length(select_if(urlComps,is.numeric))  # returns the number of numeric variables in a data frame
```

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

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

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

```
## [1] 1.725194
```

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)
```

```
## [1] 2007.289
```

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

```
## [1] 1900
```

```
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

```
minIndex <- which.min(urlComps$founded_year)
urlComps[minIndex ,]
```

```
## # A tibble: 1 x 19
##   permalink   name    homepage_url  category_list market funding_total_u~ status
##   <chr>       <chr>   <chr>         <chr>         <chr>  <chr>            <chr>
## 1 /organizat~ The Un~ http://www.n~ |Education|   Educa~ 23 650 306       opera~
## # ... with 12 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>
```

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.

```
withoutHomeURL <- data.frame(dfComps[ dfComps$noURL,])
mean(withoutHomeURL$funding_rounds, na.rm = TRUE)
```

```
## [1] 1.198917
```

Part 4: Use string operations 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() 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$company1 <- str_replace( urlComps$permalink, "/organization/", "")
```

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)
```

```
## spec_tbl_df[,21] [44,435 x 21] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ permalink        : chr [1:44435] "/organization/waywire" "/organization/tv-communications" "/orga
##  $ name             : chr [1:44435] "#waywire" "&TV Communications" "'Rock' Your Paper" "(In)Touch N
##  $ homepage_url     : chr [1:44435] "http://www.waywire.com" "http://enjoyandtv.com" "http://www.roc
##  $ category_list    : chr [1:44435] "|Entertainment|Politics|Social Media|News|" "|Games|" "|Publish
##  $ 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 ...
##  $ company          : chr [1:44435] "waywire" "tv-communications" "rock-your-paper" "in-touch-networ
##  $ company1         : chr [1:44435] "waywire" "tv-communications" "rock-your-paper" "in-touch-networ
##  - attr(*, "spec")=
##   .. cols(
##   ..   permalink = col_character(),
##   ..   name = col_character(),
##   ..   homepage_url = col_character(),
##   ..   category_list = col_character(),
##   ..   market = col_character(),
##   ..   funding_total_usd = col_character(),
##   ..   status = col_character(),
##   ..   country_code = col_character(),
##   ..   state_code = col_character(),
##   ..   region = col_character(),
##   ..   city = col_character(),
##   ..   funding_rounds = col_double(),
##   ..   founded_at = col_character(),
##   ..   founded_month = col_character(),
##   ..   founded_quarter = col_character(),
##   ..   founded_year = col_double(),
##   ..   first_funding_at = col_character(),
##   ..   last_funding_at = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# funding_total_usd is coded as character but it should be numeric
urlComps$num_funding_total_usd <- as.numeric(urlComps$funding_total_usd)
```

```
## Warning: NAs introduced by coercion
```

```
# all the values in the new column are NAs
```

M. 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}", "")
```

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.

```
urlComps$num_funding_total_usd <- as.numeric(urlComps$funding_new)
```

```
## Warning: NAs introduced by coercion
```

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

```
## [1] 18321551
```

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:**

```
library(stringi)  # importing the stringi library
convertCharToNum <- function(char_string) {
  step1 <- stri_replace_all_charclass(char_string,"\\p{WHITE_SPACE}", "")  # removing all white spaces
  step2 <- as.numeric(step1) # converting string to numeric type
  return(step2)
}
```

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

```
convertCharToNum(" 1234")
```

```
## [1] 1234
```

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

```
## Warning in convertCharToNum(urlComps$funding_total_usd): NAs introduced by
## coercion
```

```
##     [1]     1750000     4000000       40000     1500000     1200000     7000000
##     [7]     4912393     2000000          NA       41250    10600000       40000
##    [13]          NA     1750000     2050000       40000      500000          NA
##    [19]     2535000     4962651     4059079    10000000     3000000     3000000
```

```
## [44413]     3384225      800000       75000      666154    12039999          NA
## [44419]     2257464    38900000          NA          NA     3805520   866550786
## [44425]    25000000    14750000    34275015    15419877     1510500     2686600
## [44431]      320000     1587301       97398     9300000    45000000
```

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,"\\p{WHITE_SPACE}", "")
  step2 <- as.numeric(step1)
  return(step2)
  #make sure there is a return statement
}
```

```
#test the new function
convertCharToNumNew(" 1234")
```

```
## [1] 1234
```

```
urlComps$test <- convertCharToNumNew(urlComps$funding_total_usd)
```

```
## Warning in convertCharToNumNew(urlComps$funding_total_usd): NAs introduced by
## coercion
```

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

```
urlComps$funding_total_usd1 <- convertCharToNum(urlComps$funding_total_usd)
```

```
## Warning in convertCharToNum(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$funding_total_usd1, na.rm = TRUE)
```

```
## [1] 18321551
```