Intro to Data Science - HW 3

```
# Enter your name here: Nora Lin
```


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.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

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

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.

```
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.
```

```
#dfComps
```

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

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

```
urlComps <- drop_na(dfComps)

#checking to see if it dropped:
#urlComps
```

D. How many companies are missing a homepage URL?

```
nrow(dfComps)-nrow(urlComps)
```

```
## [1] 27289
```

```
#There were 27,289 companies that were missing a homepage URL.
```

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)
```

```
## tibble [20,469 x 18] (S3: tbl_df/tbl/data.frame)
##  $ permalink        : chr [1:20469] "/organization/waywire" "/organization/n-plusn" "/organization/0
##  $ name             : chr [1:20469] "#waywire" "#NAME?" "004 Technologies" "0xdata" ...
```

```
##  $ homepage_url     : chr [1:20469] "http://www.waywire.com" "http://plusn.com" "http://004gmbh.de/en
##  $ category_list    : chr [1:20469] "|Entertainment|Politics|Social Media|News|" "|Software|" "|Softw
##  $ market           : chr [1:20469] "News" "Software" "Software" "Analytics" ...
##  $ funding_total_usd: chr [1:20469] "1 750 000" "1 200 000" "-" "10 600 000" ...
##  $ status           : chr [1:20469] "acquired" "operating" "operating" "operating" ...
##  $ country_code     : chr [1:20469] "USA" "USA" "USA" "USA" ...
##  $ state_code       : chr [1:20469] "NY" "NY" "IL" "CA" ...
##  $ region           : chr [1:20469] "New York City" "New York City" "Springfield, Illinois" "SF Bay A
##  $ city             : chr [1:20469] "New York" "New York" "Champaign" "Mountain View" ...
##  $ funding_rounds   : num [1:20469] 1 2 1 2 1 1 4 1 2 5 ...
##  $ founded_at       : chr [1:20469] "1/6/12" "1/1/12" "1/1/10" "1/1/11" ...
##  $ founded_month    : chr [1:20469] "2012-06" "2012-01" "2010-01" "2011-01" ...
##  $ founded_quarter  : chr [1:20469] "2012-Q2" "2012-Q1" "2010-Q1" "2011-Q1" ...
##  $ founded_year     : num [1:20469] 2012 2012 2010 2011 1986 ...
##  $ first_funding_at : chr [1:20469] "30/06/2012" "29/08/2012" "24/07/2014" "3/1/13" ...
##  $ last_funding_at  : chr [1:20469] "30/06/2012" "4/9/14" "24/07/2014" "19/07/2014" ...
```

```
#The data set has 2 numeric variables, "Funding_rounds" and "Founded_year".
```

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

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

```
## [1] 2.033856
```

```
#The average number of funding rounds for the companies in urlComps data set was 2.033856.
```

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. You can ignore NAs with basic math calculations. For example, instead of running mean(urlComps$founded_year), something like this will work for determining the average (note that this question needs to use a different function than 'mean'.

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

#your code goes here
min(urlComps$founded_year, na.rm=TRUE)
```

```
## [1] 1902
```

```
#The oldest company in the dataframe was founded in 1902.
```

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 str_replace() in tidyverse or gsub() to clean the values of this variable:

```
#looking at  permalink variable:
#head(urlComps$permalink0
#we want to get rid of /organization/
urlComps$permalink <- str_replace(urlComps$permalink, "/organization/","")
#head(urlComps$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?

```
#funding_total_usd should be numeric but it is coded as character.
```

```
#Note: in character form, NA vlaues are noted as "-"
head(as.numeric(urlComps$funding_total_usd))
```

```
## Warning in head(as.numeric(urlComps$funding_total_usd)): NAs introduced by
## coercion
```

```
## [1] NA NA NA NA NA NA
```

```
#They are all NA.
```

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, and explain what it is doing:

```
library(stringi)
urlComps$funding_new <- stri_replace_all_charclass(urlComps$funding_total_usd,"\\p{WHITE_SPACE}", "")
```

```
#It got rid of all the white space that separated the digits.
```

```
#Checking to see if it worked
#urlComps$funding_new
```

N. You are now ready to convert urlComps$funding_new to numeric using as.numeric().

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

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

```
#check:
str(urlComps)
```

```
## tibble [20,469 x 19] (S3: tbl_df/tbl/data.frame)
##  $ permalink        : chr [1:20469] "waywire" "n-plusn" "004-technologies" "0xdata" ...
##  $ name             : chr [1:20469] "#waywire" "#NAME?" "004 Technologies" "0xdata" ...
##  $ homepage_url     : chr [1:20469] "http://www.waywire.com" "http://plusn.com" "http://004gmbh.de/en
##  $ category_list    : chr [1:20469] "|Entertainment|Politics|Social Media|News|" "|Software|" "|Softw
##  $ market           : chr [1:20469] "News" "Software" "Software" "Analytics" ...
##  $ funding_total_usd: chr [1:20469] "1 750 000" "1 200 000" "-" "10 600 000" ...
##  $ status           : chr [1:20469] "acquired" "operating" "operating" "operating" ...
##  $ country_code     : chr [1:20469] "USA" "USA" "USA" "USA" ...
##  $ state_code       : chr [1:20469] "NY" "NY" "IL" "CA" ...
##  $ region           : chr [1:20469] "New York City" "New York City" "Springfield, Illinois" "SF Bay A
##  $ city             : chr [1:20469] "New York" "New York" "Champaign" "Mountain View" ...
##  $ funding_rounds   : num [1:20469] 1 2 1 2 1 1 4 1 2 5 ...
##  $ founded_at       : chr [1:20469] "1/6/12" "1/1/12" "1/1/10" "1/1/11" ...
##  $ founded_month    : chr [1:20469] "2012-06" "2012-01" "2010-01" "2011-01" ...
##  $ founded_quarter  : chr [1:20469] "2012-Q2" "2012-Q1" "2010-Q1" "2011-Q1" ...
##  $ founded_year     : num [1:20469] 2012 2012 2010 2011 1986 ...
##  $ first_funding_at : chr [1:20469] "30/06/2012" "29/08/2012" "24/07/2014" "3/1/13" ...
##  $ last_funding_at  : chr [1:20469] "30/06/2012" "4/9/14" "24/07/2014" "19/07/2014" ...
##  $ funding_new      : num [1:20469] 1750000 1200000 NA 10600000 NA ...
```

O.Calculate the average funding amount for urlComps. If you get "NA," try using the na.rm=TRUE argument from problem I.

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

```
## [1] 20359138
```

```
#The mean funding amount was 20,359,138.
```

P.Sample three unique observations from urlComps$funding_rounds, store the results in the vector 'observations'

```
#Checking out urlComps$funding_roounds:
#all values are numeric, no NA values
observations <- sample(urlComps$funding_rounds,size=3,replace=TRUE)
observations
```

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

Q.Take the mean of those observations

```
mean(observations)
```

```
## [1] 1.333333
```

R.Do the two steps (sampling and taking the mean) in one line of code

```
mean(sample(urlComps$funding_rounds, size=3, replace=TRUE))
```

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

S.Explain why the two means are (or might be) different

```
#The two means are different because line 152 is generating a new sample. This may or may not be the sa
```

T. Use the replicate( ) function to repeat your sampling of three observations of urlComps$funding_rounds observations five times. The first argument to replicate( ) is the number of repeats you want. The second argument is the little chunk of code you want repeated.

```
replicate(5,mean(sample(urlComps$funding_rounds,size=3,replace=TRUE)))
```

```
## [1] 1.666667 1.333333 1.333333 2.666667 1.333333
```

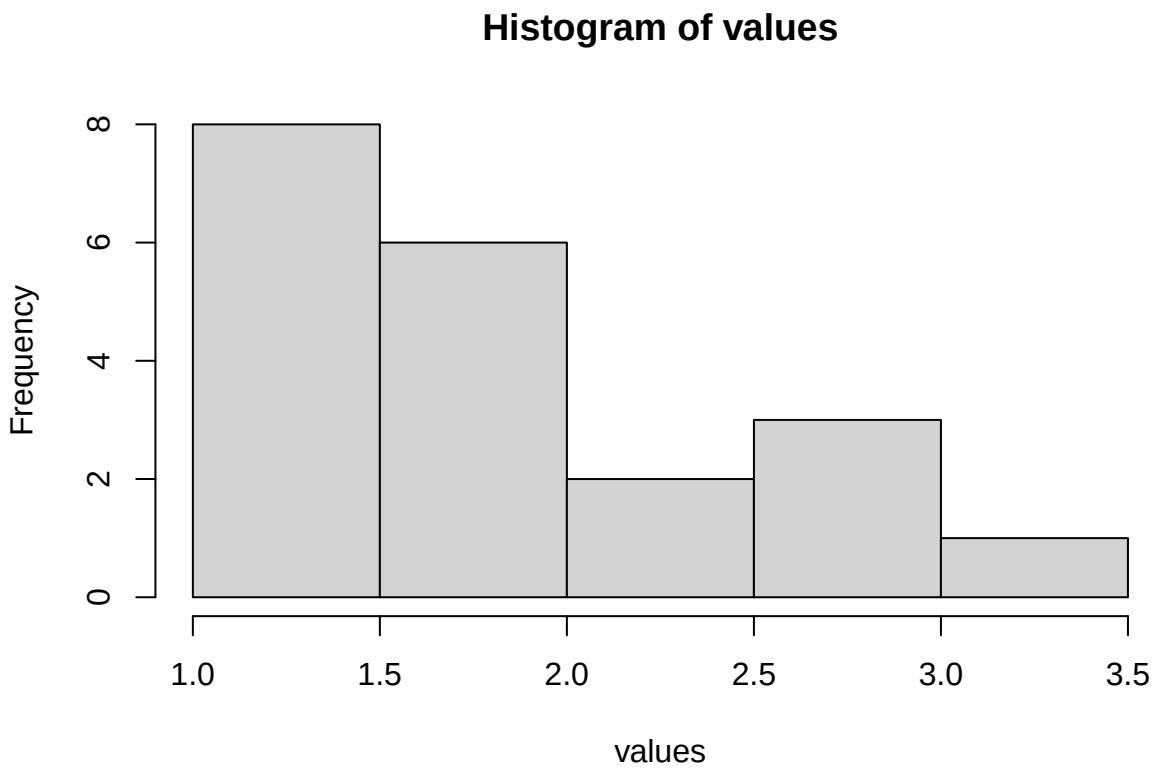
U. Rerun your replication, this time doing 20 replications and storing the output of replicate() in a variable called values.

```
values <- replicate(20,mean(sample(urlComps$funding_rounds,size=3,replace=TRUE,)))
values
```

```
##  [1] 2.000000 2.666667 1.666667 2.333333 1.000000 1.333333 1.000000 1.333333
##  [9] 2.666667 2.333333 2.000000 1.000000 1.000000 1.666667 3.333333 1.333333
## [17] 1.000000 2.000000 1.666667 3.000000
```

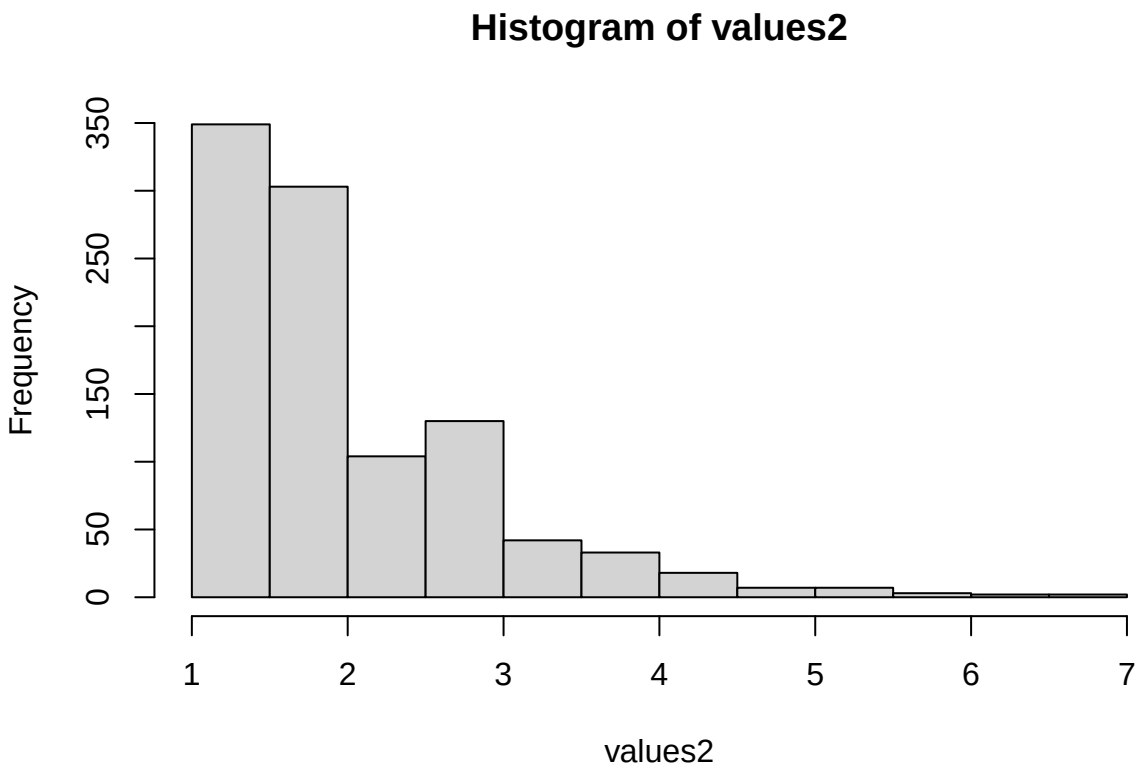
V. Generate a histogram of the means stored in values.

```
hist(values)
```



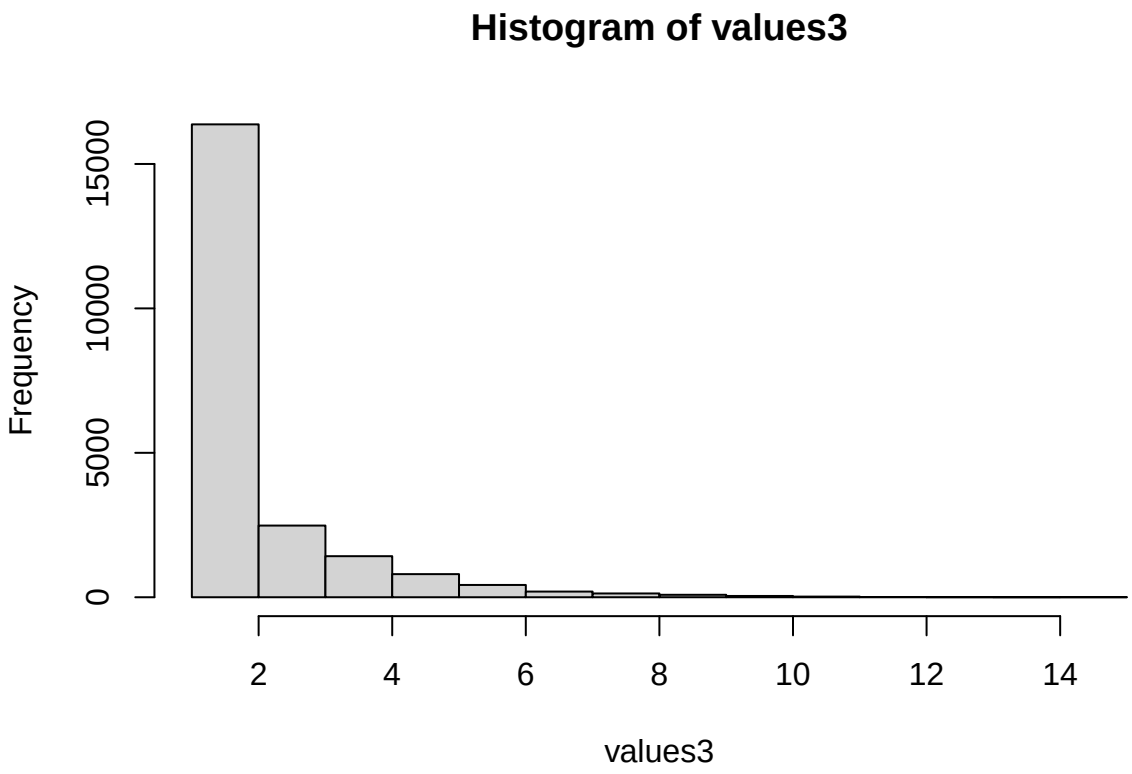
W. Rerun your replication, this time doing 1000 replications and storing the output of replicate() in a variable called values, and then generate a histogram of values.

```
values2 <- replicate(1000,mean(sample(urlComps$funding_rounds,size=3,replace=TRUE)))
#values2
hist(values2)
```



X.Repeat the replicated sampling, but this time, raise your sample size from 3 to 22. How does that affect your histogram? Explain in a comment.

```
values3 <- replicate(1000,sample(urlComps$funding_rounds,size=22,replace=TRUE))
hist(values3)
```




```
#Increasing the sample size, increases the mean for each sample. Therefore, the histogram above is dras
```

Y. Explain in a comment below, the last three histograms, why do they look different?

```
#The last three histograms all look different, however they follow a similar shape. All three are skewe
```