Intro to Data Science - HW 3


```
# Enter your name here: Benjamin Tisinger
```

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 (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 1.3.2 ──
## ✓ ggplot2 3.3.6      ✓ purrr   0.3.5
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.4.1
## ✓ readr   2.1.3      ✓ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

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

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 <- subset(dfComps, homepage_url != 'NA')
```

D. How many companies are missing a homepage URL?

```
count(urlComps)
```

```
##       n
## 1 47758
```

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)**.

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

```
str(urlComps)
```

```
## 'data.frame':    47758 obs. of  18 variables:
##  $ permalink        : chr  "/organization/waywire" "/organization/tv-communications" "/organi
zation/rock-your-paper" "/organization/in-touch-network" ...
##  $ name             : chr  "#waywire" "&TV Communications" "'Rock' Your Paper" "(In)Touch Net
work" ...
##  $ homepage_url     : chr  "http://www.waywire.com" "http://enjoyandtv.com" "http://www.rocky
ourpaper.org" "http://www.InTouchNetwork.com" ...
##  $ category_list    : chr  "|Entertainment|Politics|Social Media|News|" "|Games|" "|Publishin
g|Education|" "|Electronics|Guides|Coffee|Restaurants|Music|iPhone|Apps|Mobile|iOS|E-Commerce|"
...
##  $ market           : chr  " News " "Games" "Publishing" "Electronics" ...
##  $ funding_total_usd: chr  " 1 750 000 " " 4 000 000 " " 40 000 " " 1 500 000 " ...
##  $ status           : chr  "acquired" "operating" "operating" "operating" ...
##  $ country_code     : chr  "USA" "USA" "EST" "GBR" ...
##  $ state_code       : chr  "NY" "CA" "" "" ...
##  $ region           : chr  "New York City" "Los Angeles" "Tallinn" "London" ...
##  $ city             : chr  "New York" "Los Angeles" "Tallinn" "London" ...
##  $ funding_rounds   : int  1 2 1 1 2 1 1 1 1 1 ...
##  $ founded_at       : chr  "1/6/12" "" "26/10/2012" "1/4/11" ...
##  $ founded_month    : chr  "2012-06" "" "2012-10" "2011-04" ...
##  $ founded_quarter  : chr  "2012-Q2" "" "2012-Q4" "2011-Q2" ...
##  $ founded_year     : int  2012 NA 2012 2011 2012 2014 2011 NA 2007 2010 ...
##  $ first_funding_at : chr  "30/06/2012" "4/6/10" "9/8/12" "1/4/11" ...
##  $ last_funding_at  : chr  "30/06/2012" "23/09/2010" "9/8/12" "1/4/11" ...
```

```
# HAS 2 NUMERIC  = FUNDING_ROUNDS & FOUNDED_YEAR
```

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

```
## [1] 1.688576
```

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)

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

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

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:

```
urlComps$company <- str_replace(urlComps$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?

```
#Should be Numeric = Fund_total_usd

dfComps$NewFunds <- as.numeric(dfComps$funding_total_usd)
```

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

```
#Because of the Conversion, A lot of the Data has been transformed into NA Entries
```

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

```
Error in stri_replace_all_charclass(urlComps$funding_total_usd, "\\p{WHITE_SPACE}", : object 'ur
lComps' not found
Traceback:


1. 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().

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

```
urlComps$NewFunds<- as.numeric(urlComps$funding_total_usd)
```

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

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

```
## [1] 361.8108
```

Sample three unique observations from urlComps$funding_rounds, store the results in the vector ‘observations’

```
observations <- sample(urlComps$funding_rounds, size=3)
show(observations)
```

```
## [1] 5 1 1
```

Take the mean of those observations

```
mean(observations)
```

```
## [1] 2.333333
```

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

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

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

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

```
#Because it is sampling, which means pulling numbers at random. There is a chance you could get
 Duplicate numbers or a chance that you get a completely different set. Numbers can also change
 if companies get added/removed from the set.
```

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,sample(urlComps$funding_rounds, size=3))
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    5    1    7    1    1
## [2,]    1    1    1    1    1
## [3,]    1    1    1    2    2
```

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

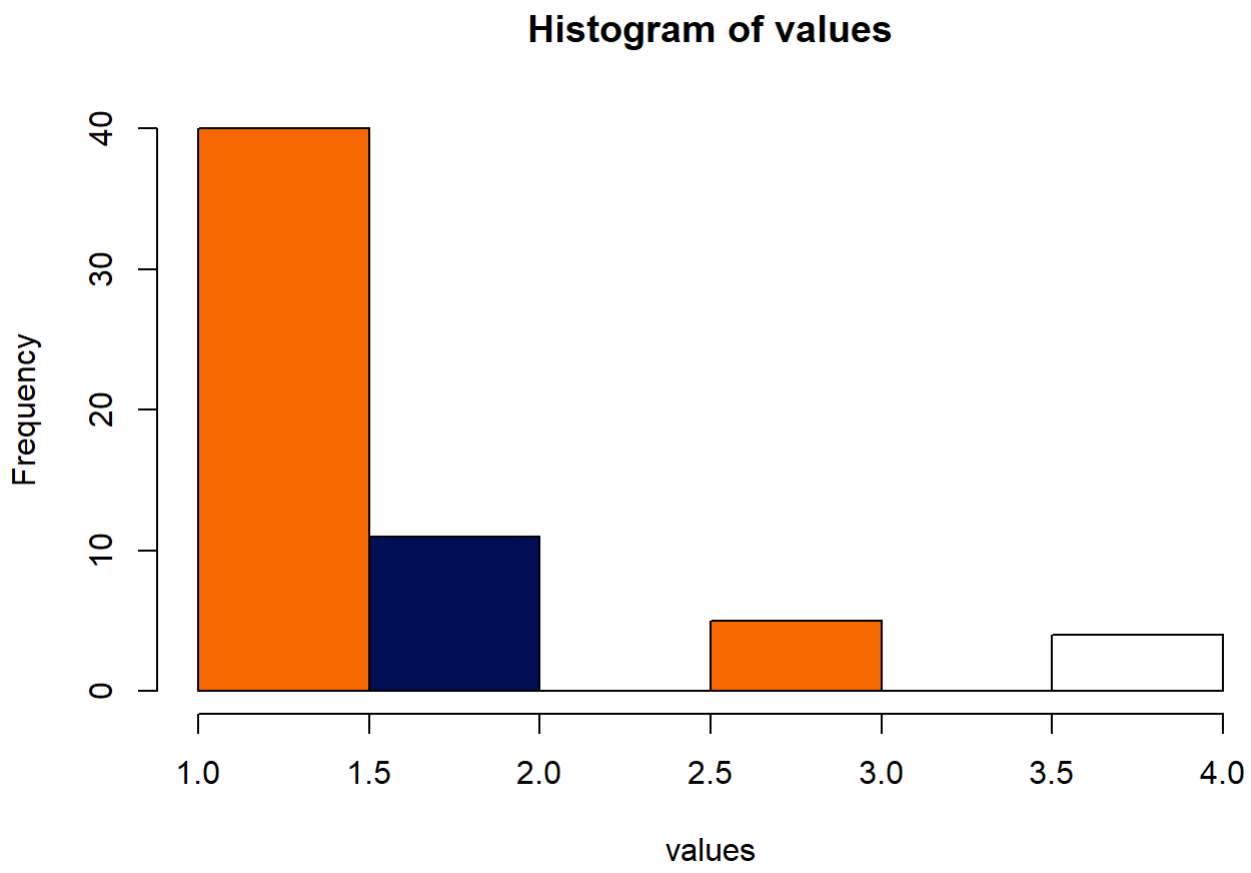
```
values <- replicate(20,sample(urlComps$funding_rounds, size=3))

show(values)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]    1    1    1    2    2    1    1    1    1     1     3     1     1     1
## [2,]    2    4    1    4    1    1    1    1    1     2     1     1     1     1
## [3,]    1    1    2    2    1    1    3    2    1     3     1     4     1     1
##      [,15] [,16] [,17] [,18] [,19] [,20]
## [1,]     2     1     2     2     4     1
## [2,]     1     2     1     1     1     1
## [3,]     1     1     1     3     1     3
```

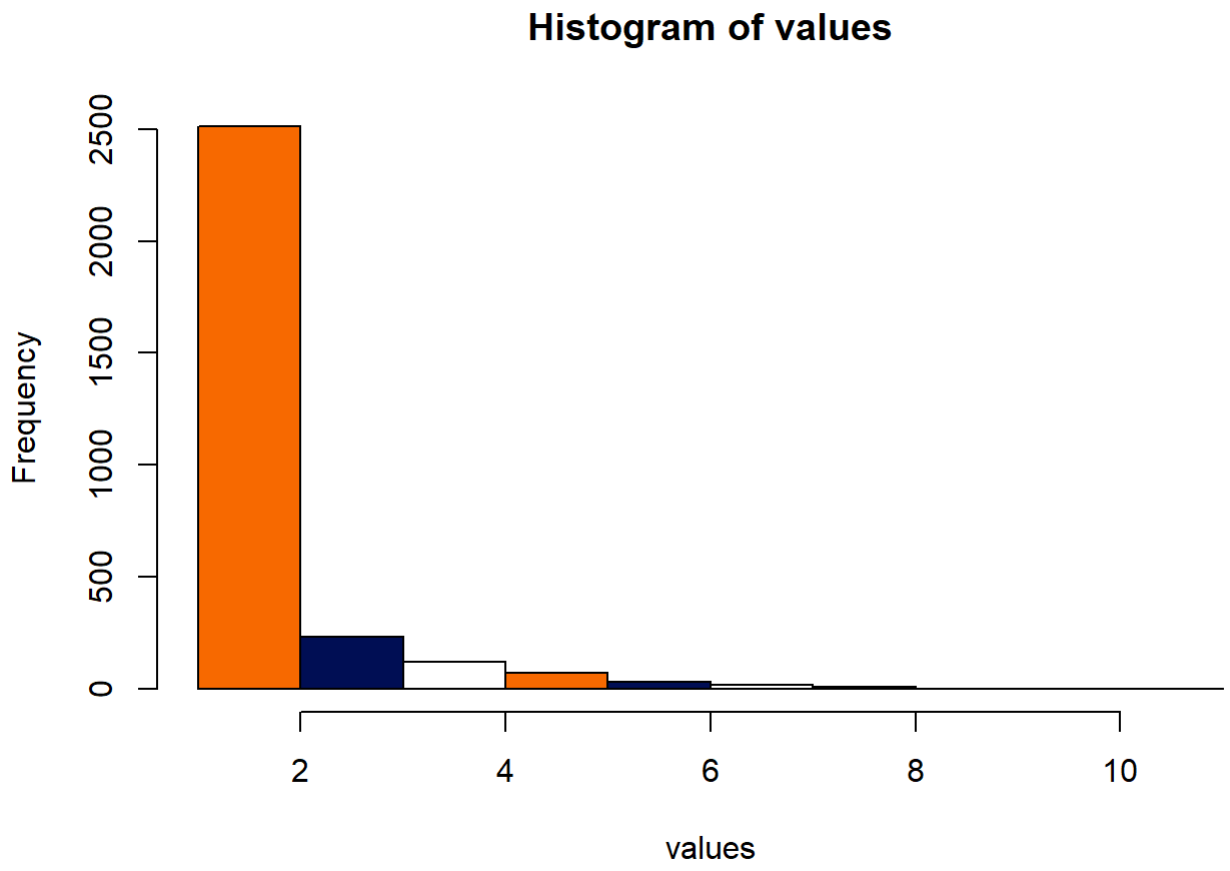
Generate a **histogram** of the means stored in **values**.

```
hist(values, col = c("#F76900","#000E54","#FFFFFF"))
```



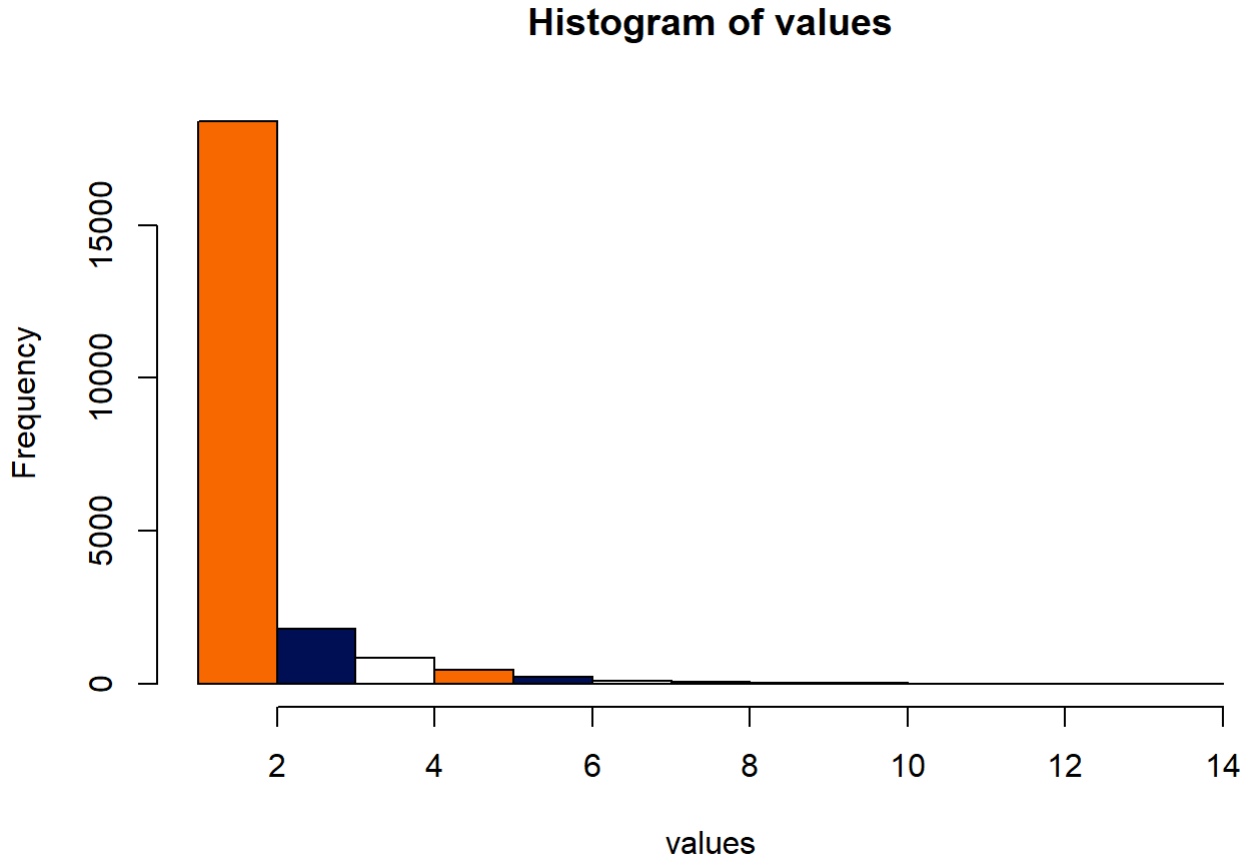
Rerun your replication, this time doing 1000 replications and storing the output of replicate() in a variable called **values**, and then generate a histogram of **values**.

```
values <- replicate(1000,sample(urlComps$funding_rounds, size=3))
hist(values, col = c("#F76900","#000E54","#FFFFFF"))
```



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.

```
values <- replicate(1000,sample(urlComps$funding_rounds, size=22))
hist(values, col = c("#F76900","#000E54","#FFFFFF"))
```




```
# The first 3 Hist grams are skewed due to the sample size. As we increase the data it should st
abilize and show a better normalization chart.
```

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

```
#The more we increase the sample size and the quantity sampled we see different results. The hig
her the numbers are the better the chart should look.
```