Intro to Data Science - HW 3

```
# Enter your name here:  Ryan Tervo
# Course Number:         IST 687
# Assignement Name:      Homework #3
# Due Date:              31 Oct 2022
# Submitted Date:        31 Oct 2022
```

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.

```
#   CODE TO PREVENT WARNINGS/MESSAGES:
#```{r setup, include=FALSE}
#knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
#```

#   IMPORT LIBRARIES:
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
##     ggplot2 3.4.0        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()
```

```r
library(dplyr)
library(stringr)

#   DEFINE THE VARIABLES:
fileName <- "https://intro-datascience.s3.us-east-2.amazonaws.com/companies1.csv"

#   READ EXCEL FILE USING WEBSITE FILE:
dfComps  <- data.frame(read_csv(fileName, show_col_types = F))
dfComps0 <- 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**).

```r
#   Use Subsetting to create a new dataframe that contains hompage URLs.
    ##  Each homepage_url will check for either http or www.  If those exist then it subsetts
it into the urlComps
urlComps  <- dfComps[!is.na(dfComps$homepage_url), ]

#   Other alternative means to find website entries.
    #urlComps  <- dfComps[grepl("http", dfComps$homepage_url) | grepl("www", dfComps$homepage_
url), ]
    #urlComps  <- dfComps[grepl("", dfComps$homepage_url), ]
```

D. How many companies are missing a homepage URL?

```r
#   Perform Calculation:
numMissingURL <- nrow(dfComps) - nrow(urlComps)

#   Display Results:
printString <- paste('There are ', numMissingURL, ' companies missing there homepage URL.', se
p = "")
print(printString, quote = F)
```

```
## [1] There are 3323 companies missing there 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)**.

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

```
#   PART G:
#   Perform Calculation:
   #str(urlComps)    #   Initially ran, output examined, and then commented out for readabilit
y.

#   Display Results:
printString <- paste('Inspecting the structure of urlComps there are 2 numeric variables.', se
p = "")
print(printString, quote = F )
```

```
## [1] Inspecting the structure of urlComps there are 2 numeric variables.
```

```
#   PART H:
#   Perform Calculation:
aveNumFundingRounds <- mean(urlComps$funding_rounds)

#   Display Results:
print("", quote = F)
```

```
## [1]
```

```
printString <- paste('The average funding rounds in urlComps is ', round(aveNumFundingRounds,
2), ".", sep = "")
print(printString, quote = F)
```

```
## [1] The average funding rounds in urlComps is 1.73.
```

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$founded_year, na.rm=TRUE)
#   Example code:  mean(urlComps$founded_year, na.rm=TRUE)

#   Perform Calculation:
oldestCompYear <- min(urlComps$founded_year, na.rm = TRUE)

#   Display Results:
printString = paste('The oldest company was founded in the year: ', oldestCompYear, sep = "")
print(printString, quote = F)
```

```
## [1] The oldest company was founded in the year: 1900
```

```
##########################################################
# ALTERNATIVE CALCULATION:
##########################################################
#   Perform Calculation:
dfComp1  <- dfComps[ , c('name', 'founded_year' )] %>% drop_na()
dfComp1  <- dfComp1[dfComp1$founded_year == min(dfComp1$founded_year) , ]

#   Display Results:
printString = paste('The oldest company was "', dfComp1[1, 1], '" and was founded in the year:
 ', dfComp1[1, 2], '.', sep = "")
print(printString, quote = F)
```

```
## [1] The oldest company was "The University of Nottingham" and was founded in the year: 1900
.
```

```
#your code goes here
```

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:

```
#   Perform Calculation:
urlComps$permalink <- str_replace(urlComps$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?

```
#   Perform Calculation:
urlComps$funding_total_usd_num <- as.numeric(urlComps$funding_total_usd)
```

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

```
#   Inspect urlComps
head(urlComps)
```

```
##           permalink               name                  homepage_url
## 1           waywire           #waywire        http://www.waywire.com
## 2 tv-communications &TV Communications         http://enjoyandtv.com
## 3   rock-your-paper  'Rock' Your Paper  http://www.rockyourpaper.org
## 4  in-touch-network  (In)Touch Network http://www.InTouchNetwork.com
## 5           n-plusn             #NAME?              http://plusn.com
## 7      club-domains      .Club Domains              http://nic.club/
##                                                                     category_list
## 1                                       |Entertainment|Politics|Social Media|News|
## 2                                                                         |Games|
## 3                                                          |Publishing|Education|
## 4 |Electronics|Guides|Coffee|Restaurants|Music|iPhone|Apps|Mobile|iOS|E-Commerce|
## 5                                                                      |Software|
## 7                                                                      |Software|
##        market funding_total_usd    status country_code state_code
## 1        News         1 750 000  acquired          USA         NY
## 2       Games         4 000 000 operating          USA         CA
## 3  Publishing            40 000 operating          EST       <NA>
## 4 Electronics         1 500 000 operating          GBR       <NA>
## 5    Software         1 200 000 operating          USA         NY
## 7    Software         7 000 000      <NA>          USA         FL
##           region         city funding_rounds founded_at founded_month
## 1  New York City     New York              1     1/6/12       2012-06
## 2    Los Angeles  Los Angeles              2       <NA>          <NA>
## 3        Tallinn      Tallinn              1 26/10/2012       2012-10
## 4         London       London              1     1/4/11       2011-04
## 5  New York City     New York              2     1/1/12       2012-01
## 7 Ft. Lauderdale Oakland Park              1   10/10/11       2011-10
##   founded_quarter founded_year first_funding_at last_funding_at
## 1         2012-Q2         2012       30/06/2012      30/06/2012
## 2            <NA>           NA           4/6/10      23/09/2010
## 3         2012-Q4         2012           9/8/12          9/8/12
## 4         2011-Q2         2011           1/4/11          1/4/11
## 5         2012-Q1         2012       29/08/2012          4/9/14
## 7         2011-Q4         2011       31/05/2013      31/05/2013
##   funding_total_usd_num
## 1                    NA
## 2                    NA
## 3                    NA
## 4                    NA
## 5                    NA
## 7                    NA
```

```
#urlComps[ , c(funding_total_usd)]#, funding_total_usd_num)])


#urlComps$funding_total_usd_num <- as.numeric(str_replace_all(urlComps$funding_total_usd, "[[:
space:]]", ""))

#   Inspection:
printString1 <- "It appears that all of the new FUNDING_TOTAL_USD values are NA."
printString2 <- "Upon further inspection it looks like there are spaces in the FUNDING_TOTAL_U
SD values which are preventing the numeric conversion."

#   Display Output:
print(printString1, quote = F)
```

```
## [1] It appears that all of the new FUNDING_TOTAL_USD values are NA.
```

```
print(printString2, quote = F)
```

```
## [1] Upon further inspection it looks like there are spaces in the FUNDING_TOTAL_USD values
which are preventing the numeric conversion.
```

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
}", "")


#   Perform Calculation:
    # In order to convert the dollar values to numbers the spaces need to be removed.
#   METHOD #1
library(stringi)
urlComps$funding_new <- stri_replace_all_charclass(urlComps$funding_total_usd,"\\p{WHITE_SPACE
}", "")

#   METHOD #2:
dfTemp <- str_replace_all(urlComps$funding_total_usd, "[[:space:]]", "")

#   DISPLAY RESULTS:
head(dfTemp)
```

```
## [1] "1750000" "4000000" "40000"   "1500000" "1200000" "7000000"
```

```
head(urlComps$funding_new)
```

```
## [1] "1750000" "4000000" "40000"   "1500000" "1200000" "7000000"
```

```
printString <- 'The "stri_replace_all_charclass" removes all types of the white space in the s
tring regardless of position.'
print(printString, quote = F)
```

```
## [1] The "stri_replace_all_charclass" removes all types of the white space in the string reg
ardless of position.
```

```
Error in stri_replace_all_charclass(urlComps$funding_total_usd, "\\p{WHITE_SPACE}", : object '
urlComps' 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.

```
#   Perform the Calculation:
urlComps$funding_new <- as.numeric(urlComps$funding_new)
```

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

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

printMeanFundingAmount <- format(round(meanFunding, 2), nsmall = 2, big.mark =",")
    #    Take the mean funding value and format for better display.

#   Display Output:
printString <- paste('The mean funding amount was $', printMeanFundingAmount, ".", sep = "")
print(printString, quote = F)
```

```
## [1] The mean funding amount was $18,321,551.47.
```

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

```
#   Perform Calculation:
numObservations = 3
observations <- sample(urlComps$funding_rounds, numObservations, replace = FALSE)

#   Display Output:
print(observations, quote = FALSE)
```

```
## [1] 1 3 1
```

Take the mean of those observations

```
#   Perform Calculation:
meanObservations <- round(mean(observations), 2)

#   Display Output:
printString <- paste('The mean observations is: ', meanObservations, sep = '')
print(printString, quote = FALSE)
```

```
## [1] The mean observations is: 1.67
```

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

```
#   Perform Calculation:
meanObservations = round(mean(sample(urlComps$funding_rounds, numObservations, replace = FALSE
)), 2)

#   Display Output:
printString <- paste('The mean observations is: ', meanObservations, sep = '')
print(printString, quote = FALSE)
```

```
## [1] The mean observations is: 2.67
```

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

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.

```
#   Define Variables:
numSamples <- 3
numTrials  <- 5

#   Perform Calculations:
sampleMeans <- replicate(numTrials, mean(sample(urlComps$funding_rounds, numSamples, replace =
 FALSE)), simplify = TRUE)
sampleMeans <- round(sampleMeans, 2)

#   Display Results:
print(sampleMeans)
```

```
## [1] 1.67 1.00 1.33 1.00 2.67
```

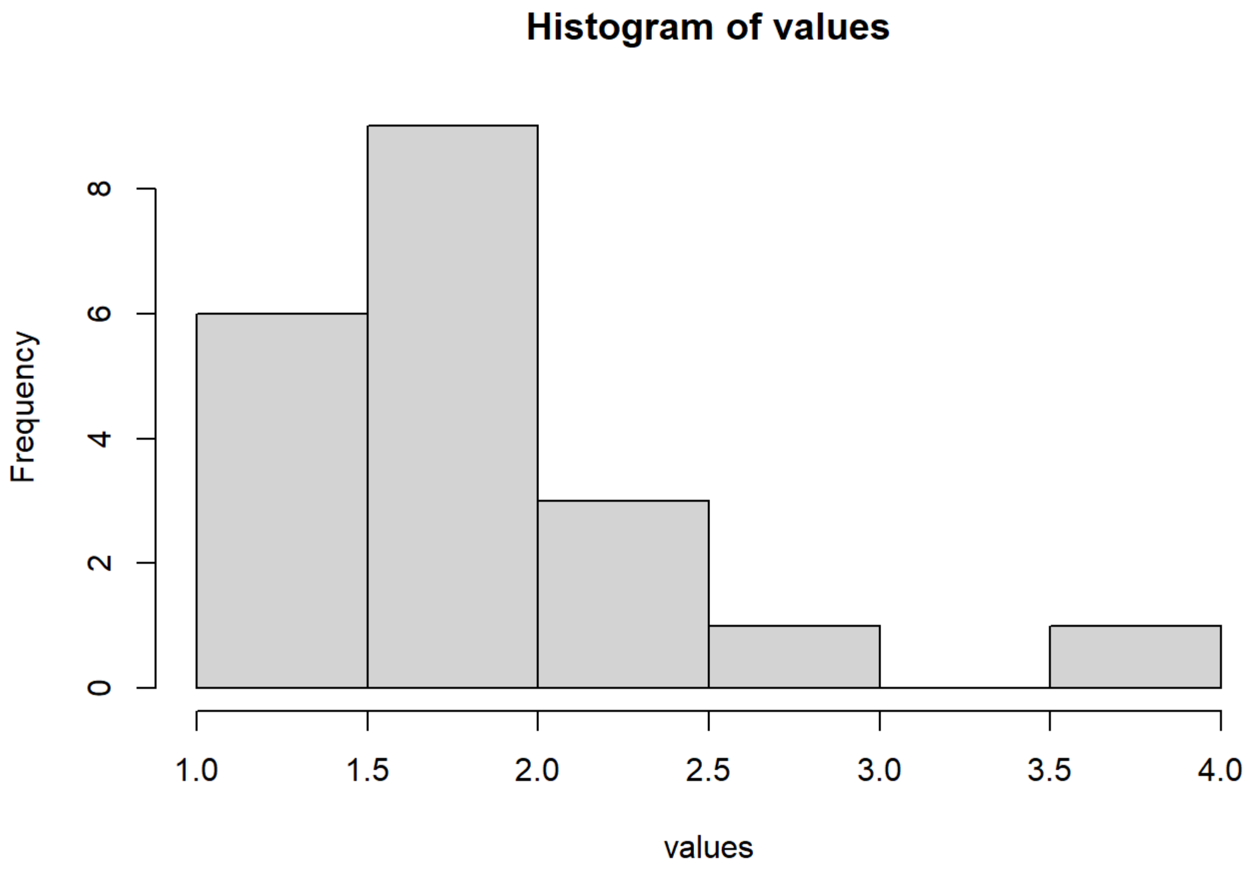
Rerun your replication, this time doing 20 replications and storing the output of replicate() in a variable called **values**.

```
#   Define Variables:
numSamples <- 3
numTrials  <- 20

#   Perform Calculations:
values <- replicate(numTrials, mean(sample(urlComps$funding_rounds, numSamples, replace = FALS
E)), simplify = TRUE)
values <- round(values, 2)

#   Display Results:
print(values)
```

```
##  [1] 2.00 1.00 2.00 1.67 1.00 1.00 1.67 2.00 1.67 2.33 1.67 2.33 3.67 1.33 2.33
## [16] 1.33 2.00 1.67 1.00 3.00
```

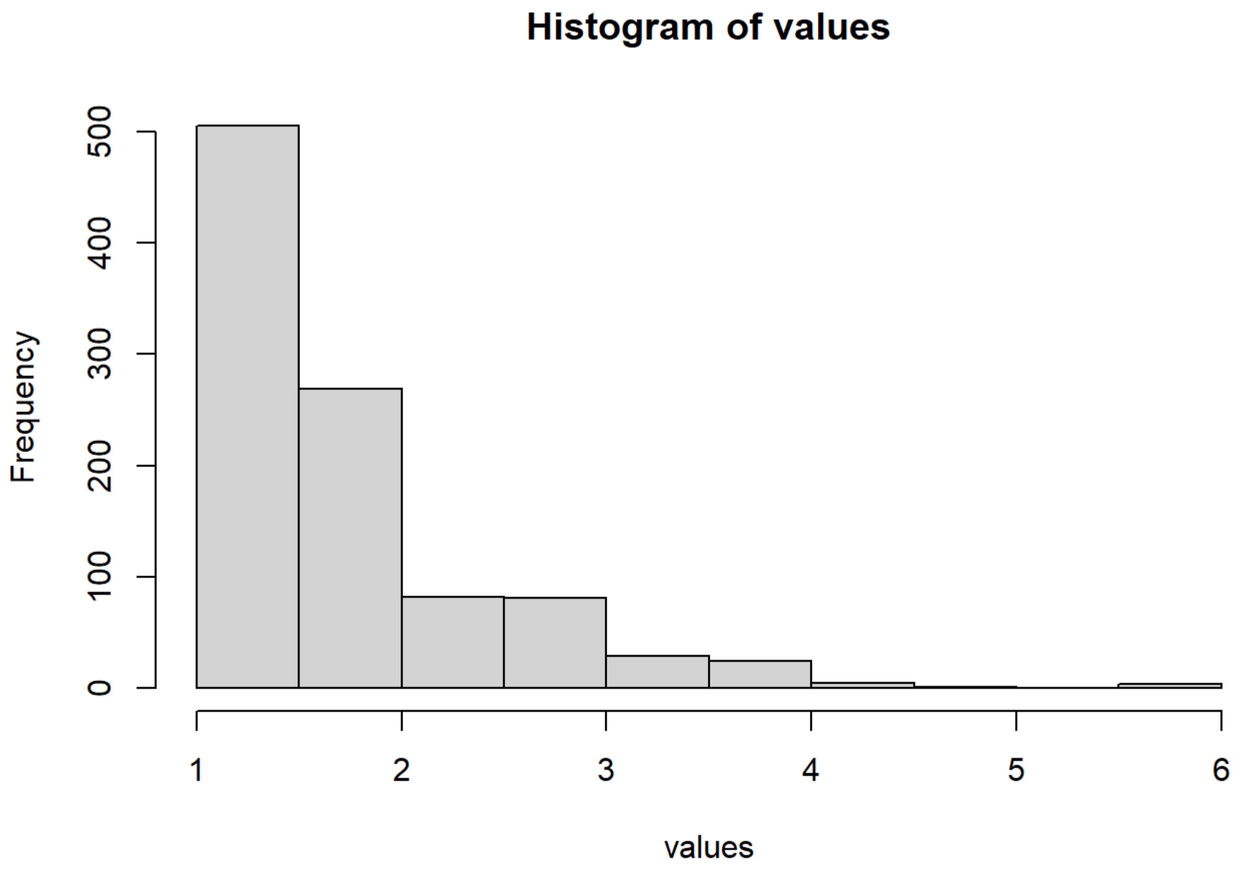
Generate a **histogram** of the means stored in **values**.

```
hist(values)
```



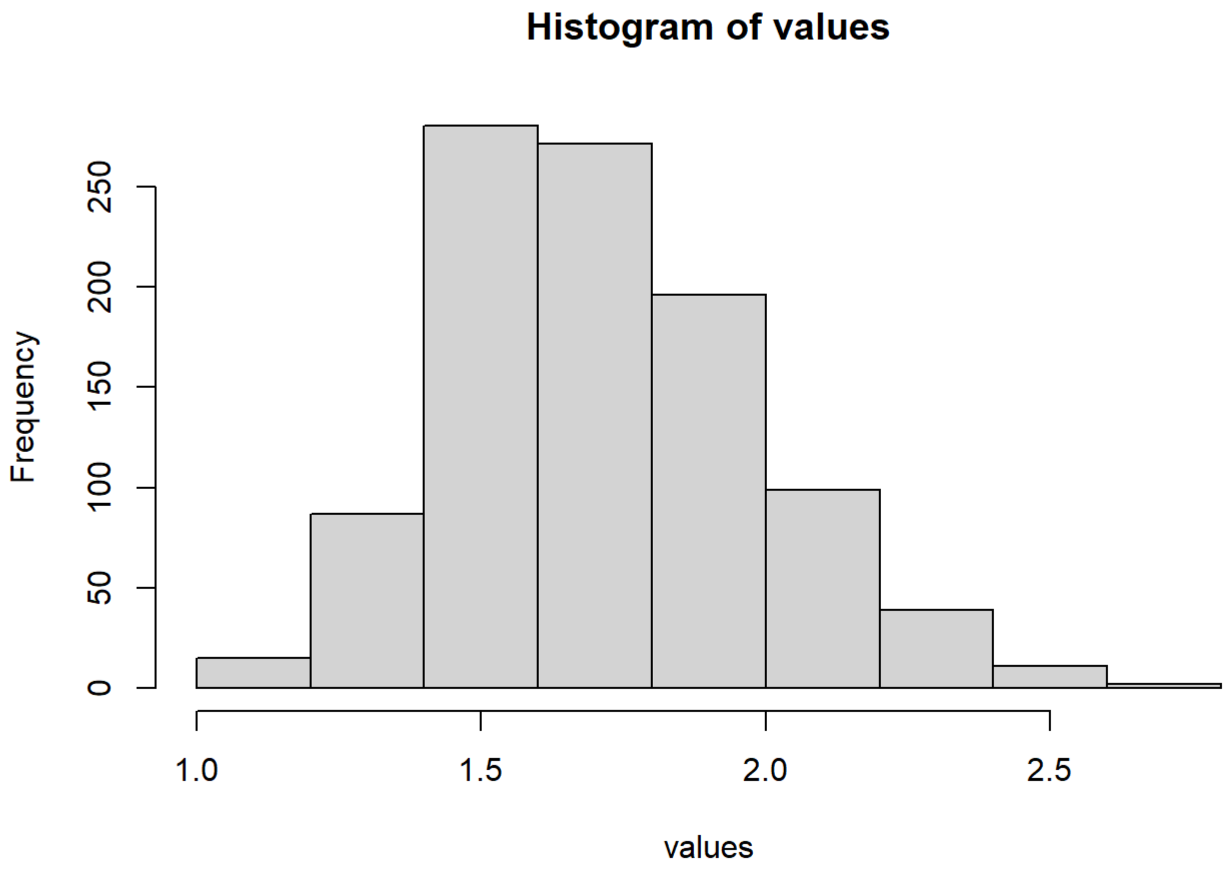
Rerun your replication, this time doing 1000 replications and storing the output of replicate() in a variable called **values**, and then generate a histogram of **values**.

```
#   Define Variables:
numSamples <- 3
numTrials  <- 1000

#   Perform Calculations:
values <- replicate(numTrials, mean(sample(urlComps$funding_rounds, numSamples, replace = FALSE)), simplify = TRUE)

#   Display Results:
hist(values)
```



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.

```
#   Define Variables:
numSamples <- 22
numTrials  <- 1000

#   Perform Calculations:
values <- replicate(numTrials, mean(sample(urlComps$funding_rounds, numSamples, replace = FALS
E)), simplify = TRUE)
#values <- round(values, 2)

#   Display Results:
#print(values)
hist(values)
```



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

```
printString1 = 'As the number of samples and trials increase then the sample mean distribution
 approaches a normal distribution.'
printString2 = 'Initially there were too few samples and trials conducted so the histogram did
 not show a normal distribution.'

print(printString1, quote = FALSE)
```

```
## [1] As the number of samples and trials increase then the sample mean distribution approach
es a normal distribution.
```

```
print(printString2, quote = FALSE)
```

```
## [1] Initially there were too few samples and trials conducted so the histogram did not show
 a normal distribution.
```