HW 4

March 4, 2021

1 IST 387 HW 4

```
[3]: # Enter your name here: Connor Hanan
```

1.0.1 Attribution statement: (choose only one and delete the rest)

```
[4]: # 1. I did this homework by myself, with help from the book and the professor.
```

(Chapters 8, 9, and 10 of Introduction to Data Science)

Reminders of things to practice from previous weeks: Descriptive statistics: mean( ) max( ) min( ) Sequence operator: : (For example, 1:4 is shorthand for 1, 2, 3, 4) Create a function: myFunc <- function(myArg) { } ?command: Ask R for help with a command

This module: Sampling is a process of **drawing elements from a larger set**. In data science, when analysts work with data, they often work with a sample of the data, rather than all of the data (which we call the **population**), because of the expense of obtaining all of the data.

One must be careful, however, because **statistics from a sample rarely match the characteristics of the population**. The **goal of this homework** is to **sample from a data set several times and explore the meaning of the results**. Before you get started make sure to read Chapters 8-10 of An Introduction to Data Science. Don't forget your comments!

1.1 Part 1: Write a function to compute statistics for a vector of numeric values

A. Create a new function which takes a numeric vector as its input argument and returns a list of statistics about that vector as the output. As a start, it should return the min, mean, and max of the vector. The function should be called **vectorStats**:

```
[18]: vectorStats <- function(n_vector) {
          a <- min(n_vector)
          b <- mean(n_vector)
          c <- max(n_vector)
          return (c(a,b,c))
      }
```

B. Test your function by calling it with the numbers **one through ten**:

[19]:

```
vectorStats(1:10)
```

1. 1 2. 5.5 3. 10

C. Enhance the vectorStats() function to add the **median** and **standard deviation** to the output.

[20]:

```
vectorStats <- function(n_vector) {
    a <- min(n_vector)
    b <- mean(n_vector)
    c <- max(n_vector)
    d <- median(n_vector)
    e <- sd(n_vector)
    return (c(a,b,c,d,e))
}
```

D. Retest your enhanced function by calling it with the numbers **one through ten**:

[21]:

```
vectorStats(1:10)
```

1. 1 2. 5.5 3. 10 4. 5.5 5. 3.02765035409749

1.2 Part 2: Sample repeatedly from the mtcars built-in dataframe

E. Copy the mtcars dataframe:

[22]:

```
myCars <- mtcars
```

Use **head(myCars)** and **tail(myCars)** to show the data. Add a comment that describes what each variable in the data set contains. **Hint:** Use the ? or help( ) command with mtcars to get help on this dataset.

[26]:

```
head(myCars)
tail(myCars)

#mpg is Miles/(US) gallon
#cyl is Number of cylinders
#disp is Displacement (cu.in.)
#hp is Gross horsepower
#drat is Rear axle ratio
#wt is Weight (1000 lbs)
#qsec is 1/4 mile time
#vs is Engine (0 = V-shaped, 1 = straight)
#am is Transmission (0 = automatic, 1 = manual)
#gear is Number of forward gears
#carb is Number of carburetors
```

A data.frame: 6 × 11

	mpg <dbl>	cyl <dbl>	disp <dbl>	hp <dbl>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <db
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0
Valiant	18.1	6	225	105	2.76	3.460	20.22	1

A data.frame: 6 × 11

	mpg <dbl>	cyl <dbl>	disp <dbl>	hp <dbl>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <dbl>
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.7	0
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.9	1
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.5	0
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.5	0
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.6	0
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.6	1

F. Sample three observations from **myCars$mpg**.

[35]:
```r
sample(myCars$mpg, size=3, replace = T)
```

G. Call your vectorStats( ) function with a sample of three observations from **myCars$mpg**.

[37]:
```r
vectorStats(sample(myCars$mpg, size=3, replace = T))
```

1. 21 2. 24.2666666666667 3. 30.4 4. 21.4 5. 5.31538647074071

H. Use the replicate( ) function to repeat your sampling ten 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.

[73]:
```r
replicate(10,vectorStats(sample(myCars$mpg, size=3, replace = T)))
```

A matrix: 5 × 10 of type dbl

10.40000	16.400000	10.400000	15.0000000	15.20000	10.400000	15.800000
16.03333	21.733333	18.800000	15.7333333	21.43333	15.500000	21.800000
27.30000	24.400000	27.300000	16.4000000	33.90000	22.800000	30.400000
10.40000	24.400000	18.700000	15.8000000	15.20000	13.300000	19.200000
9.75722	4.618802	8.450444	0.7023769	10.79645	6.486139	7.639372

I. Write a comment describing why every replication produces a different result.

[42]:
```r
#every time it goes to replicate, it runs the sampling again, so it will be␣
↪gathering a different set of 3 numbers from
#myCars$mpg
```

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

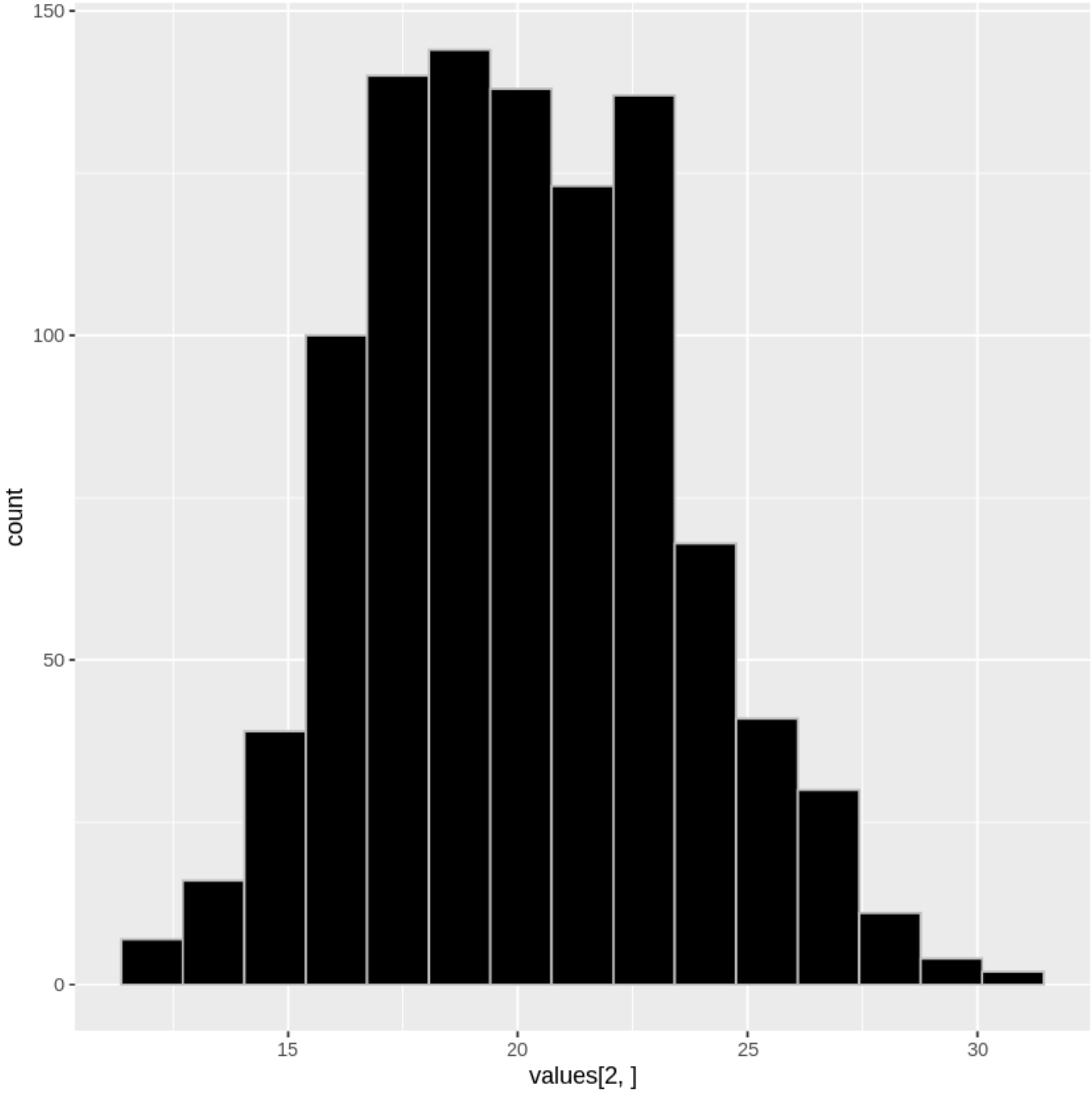
[75]:
```r
values <- (replicate(1000,vectorStats(sample(myCars$mpg, size=3, replace = T))))
values
```

```
                                  10.400000  14.30000  17.300000  21.400000  18.100000  10.400000  10.40000
                                  16.900000  18.50000  21.433333  24.400000  24.333333  16.366667  15.96666
A matrix: 5 × 1000 of type dbl    26.000000  26.00000  26.000000  30.400000  33.900000  24.400000  22.80000
                                  14.300000  15.20000  21.000000  21.400000  21.000000  14.300000  14.70000
                                  8.118497   6.51076   4.366158   5.196152   8.410906   7.225187   6.296295
```

K. Generate a **histogram** of the means stored in values. You need to **index into values** for that.

[76]:
```
library(tidyverse)
```

[77]:
```
ggplot(data.frame(values[2,]),aes(x=values[2,])) +␣
↪geom_histogram(bins=15,fill='black',col='grey')
```

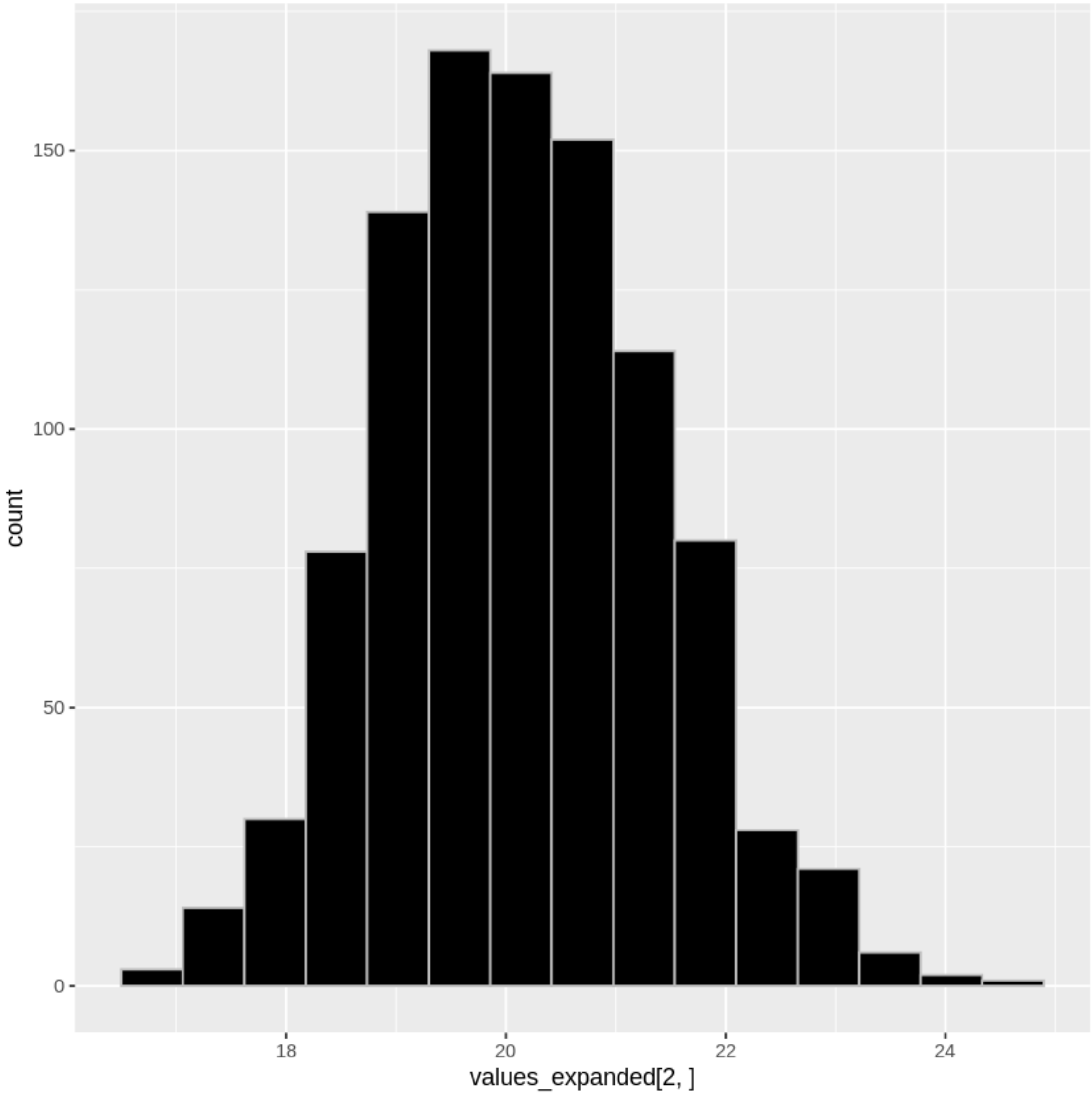


L. Repeat the replicated sampling, but this time, raise your sample size from **3 to 22**. How does that affect your histogram? Explain in a comment.

[78]:

```
values_expanded <- (replicate(1000,vectorStats(sample(myCars$mpg, size=22,␣
 ↪replace = T))))
ggplot(data.frame(values_expanded[2,]),aes(x=values_expanded[2,])) +␣
 ↪geom_histogram(bins=15,fill='black',col='grey')

#the histogram has moved towards a slightly more normal distribution
#increasing the sample size betters the accuracy of the mean, allowing fewer␣
 ↪extremes to occur
```



[ ]:

```
```