project01

Danny Ceron

September 11, 2018

Exercise 1: An engineer is studying the formulation of a Portland cement mortar. He has added a polymer latex emulsion during mixing to determine if this impacts tension bond strength of the mortar. The experimenter prepared 10 samples of the original formulation and 10 samples of the modified formulation. The tension bond strength data from this experiment are shown in the following table.

Modified Mortar 16.85 16.4 17.21 16.35 16.52 17.04 16.96 17.15 16.59 16.57 Unmodified Mortar
16.62
16.75
17.37
17.12
16.98
16.87
17.34
17.02
17.08
17.27

(a) Use c(16.62 ,16.75,17.37 ,17.12 ,16.98 ,16.87 ,17.34 ,17.02 ,17.08,17.27) to generate two datasets.

```
ModMortar <- c(16.85,   16.4,   17.21,  16.35,  16.52,  17.04,  16.96,  17.15,  16.59,  16.57)
UnModMortar <- c(16.62  ,16.75,17.37    ,17.12  ,16.98  ,16.87  ,17.34  ,17.02  ,17.08,17.27)
```

(b) Use mean() and median() to calculate the mean and median of each dataset.

```
"Mean modified Mortar:"
```

```
## [1] "Mean modified Mortar:"
```

```
mean(ModMortar)
```

```
## [1] 16.764
```

```
"Median modified Mortar:"
```

```
## [1] "Median modified Mortar:"
```

```
median(ModMortar)
```

```
## [1] 16.72
```

```
"Mean un modified Mortar:"
```

```
## [1] "Mean un modified Mortar:"
```

```
mean(UnModMortar)
```

```
## [1] 17.042
```

```
"Median Un modified Mortar:"
```

```
## [1] "Median Un modified Mortar:"
```

```
median(UnModMortar)
```

```
## [1] 17.05
```

(c) Use sd(), var() and IQR() to calculate the sample standard deviation, sample variance and IQR of each dataset.

```
"modified"
```

```
## [1] "modified"
```

```
"Standard Diviation:"
```

```
## [1] "Standard Diviation:"
```

```
sd(ModMortar)
```

```
## [1] 0.3164455
```

```
"Varience"
```

```
## [1] "Varience"
```

```
var(ModMortar)
```

```
## [1] 0.1001378
```

```
"IQR"
```

```
## [1] "IQR"
```

```
IQR(ModMortar)
```

```
## [1] 0.4875
```

```
"Unmodified"
```

```
## [1] "Unmodified"
```

```
"Standard Diviation:"
```

```
## [1] "Standard Diviation:"
```

```
sd(UnModMortar)
```

```
## [1] 0.2479158
```

```
"Varience"
```

```
## [1] "Varience"
```

```
var(UnModMortar)
```

```
## [1] 0.06146222
```

```
"IQR"
```

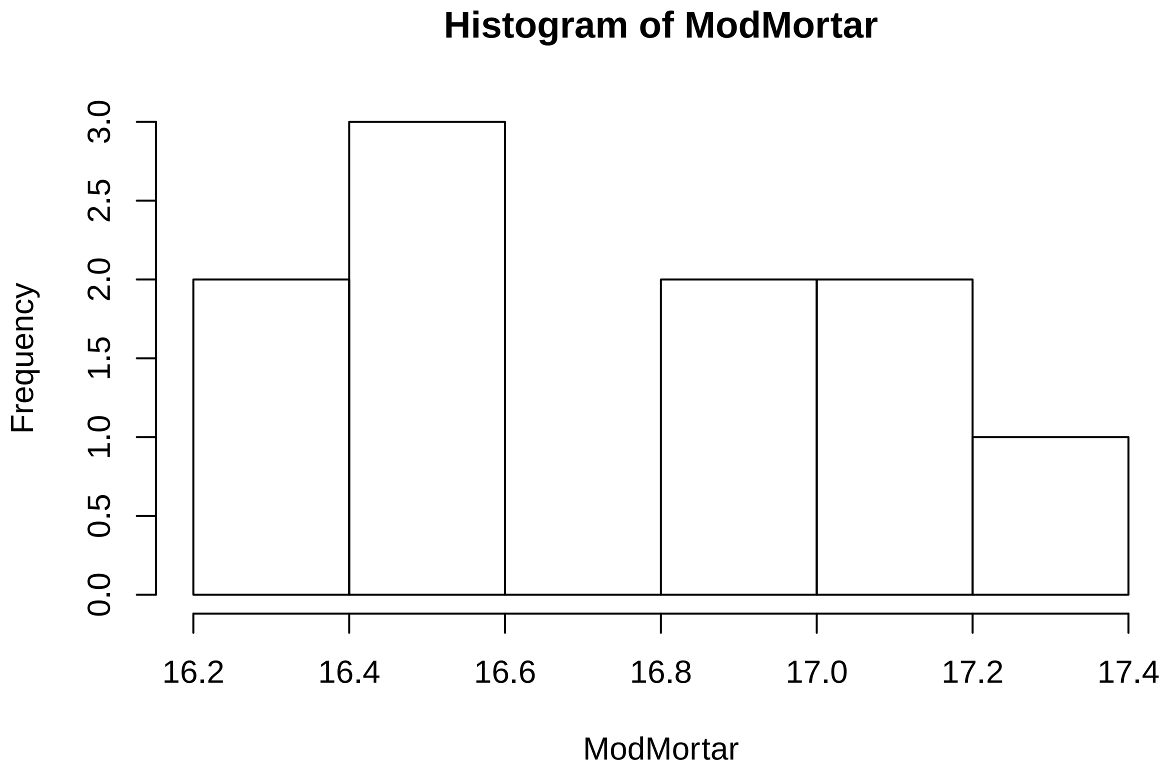
```
## [1] "IQR"
```

```
IQR(UnModMortar)
```

```
## [1] 0.335
```

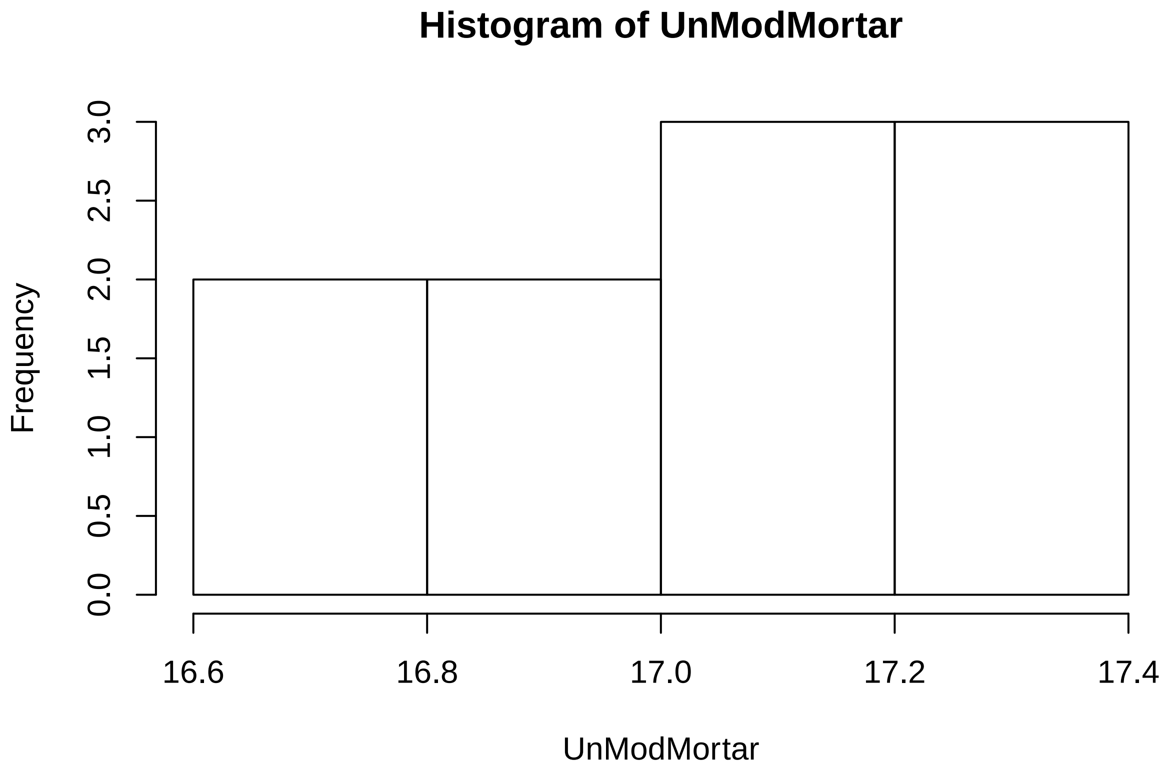
(d) Construct the histograms for the two datasets and make comments about the shapes.

```
hist(ModMortar)
```



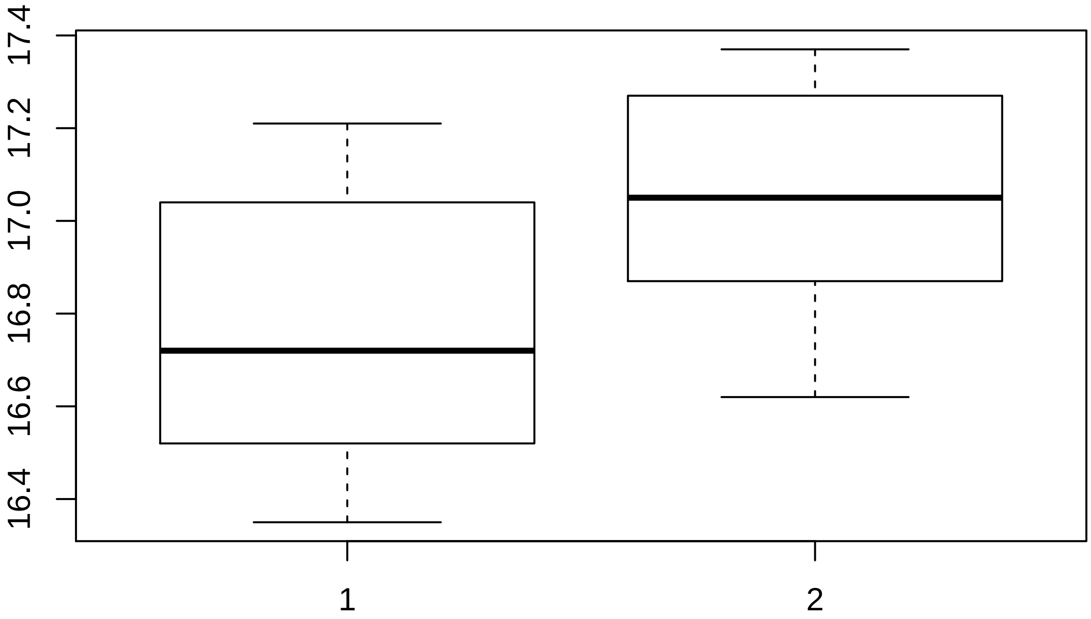
The barplot for modified Mortar no values between 16.6 and 16.8 weren't collected or never happen dont belong in the set. Really Unsecure of what happen when collecting the data. But if we were to have values. we could say that the data is right skewed, since most of the data collected is towards the left, it is easy to make the assumption that it is right skewed but we cant determine how many peacks it contains because we are bissing data in between.

```
hist(UnModMortar)
```



(e) Construct comparative box-plots for the two groups and make comments about the similarity and difference.

```
boxplot(ModMortar,UnModMortar)
```



The only similarities is that they dont have outliers, theu both have whiskers, some of their values are the same. Their max and min values appear to be different their sample median are also different. the whiskers for boxplot 1 look more symetric from both sides, while boxplot one of the whiskers seem to be farther to the lower bound of the data.

Exercise 2: A survey was conducted in a class where 30 students are enrolled. The survey question is: how many courses is the student currently taking this semester? The response data are shown in the data file “P1_Classes.csv”.

```
data1 = read.csv("P1_Classes.csv")
```

(a) Construct the frequency table using table().

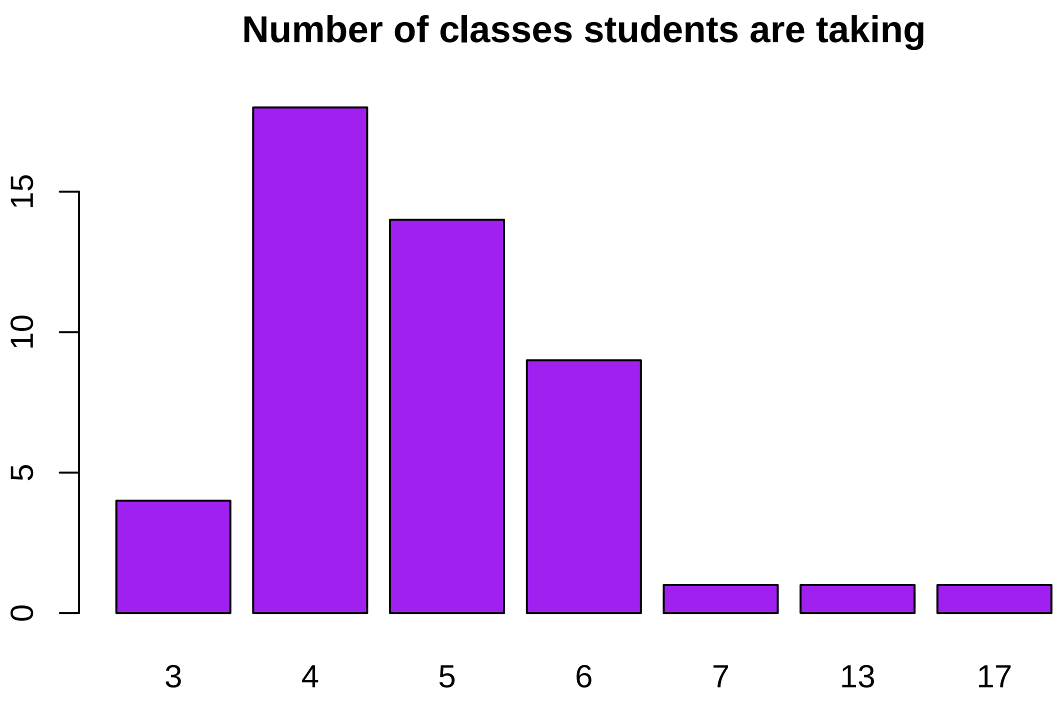
```
"freaqency table"
```

```
## [1] "freaqency table"
```

```
tabledata = table(data1$Classes)
```

(b) Construct bar-plot using barplot(). Add a title to the plot. You get a bonus point if you can add colors.

```
barplot(tabledata, main = "Number of classes students are taking",col = "purple")
```



(c) Using R functions to count, how many students are taking more than three courses?

```
count = which(data1[[1]]>=3)
"the amount of students taking 3 or more classes:"
```

```
## [1] "the amount of students taking 3 or more classes:"
```

```
length(count)
```

```
## [1] 48
```

(d) Are there any outliers in this data? If so, what numbers are they? Use IQR and quartile().

```
"IQR:"
```

```
## [1] "IQR:"
```

```
theIQR = IQR(data1[[1]])
theIQR
```

```
## [1] 1.25
```

```
#"Quartiles:"
q25 = quantile(data1[[1]], probs = .25)#Q1
q75 = quantile(data1[[1]], probs = .75)#Q3

#calculating lower and upper fence
lowerFence = q25-1.5*theIQR
upperFence = q75+1.5*theIQR
#Outlier values
```

```
lowerOutlier = which(data1[[1]]<=lowerFence)
upperOutlier = which(data1[[1]]>=upperFence)
"lower outlier"
```

```
## [1] "lower outlier"
```

```
data1[[1]][c(lowerOutlier)]
```

```
## integer(0)
```

```
"UpperoutLier:"
```

```
## [1] "UpperoutLier:"
```

```
data1[[1]][c(upperOutlier)]
```

```
## [1] 17 13
```

there are two outliers which are higer than most of the data. 17 and 13 are the only values that are outliers.

Exercise 3:

(a) Use seq() to generate a sequence 2, 4, . 24.

```
leseq = seq(2,24,by = 2)
leseq
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20 22 24
```

(b) Use log() to generate a new sequence where each element is log-transformed from the

```
log(leseq)
```

```
##  [1] 0.6931472 1.3862944 1.7917595 2.0794415 2.3025851 2.4849066 2.6390573
##  [8] 2.7725887 2.8903718 2.9957323 3.0910425 3.1780538
```

sequence in (a).

(c) Remove the second to fifth elements in the resulting sequence in (b).

```
x = leseq[-5]#Assigning the leftover elements into x
x
```

```
##  [1]  2  4  6  8 12 14 16 18 20 22 24
```

6 is the number that dissapears becaus 6 is located in the 5 location of the array.

(d) Use length() to obtain the length of the resulting sequence in (c).

```
length(x)#getting the length of the stored sequence
```

```
## [1] 11
```

(e) Sort the resulting sequence in (c) from high to low using sort().

```
sort(x,decreasing = TRUE)#this is the sequence from part c
```

```
##  [1] 24 22 20 18 16 14 12  8  6  4  2
```