Scott Kobos

1/20/19

HW1

Stat318

1) **Data Entry**

a) The head(air.data1,2) command does not display the whole data set, it only displays the first two rows after the header. From tail(air.data1,3) I would expect to see the last three rows of the data.

```
> air.data1 = read.delim ("clipboard",header= T)
> head(air.data1,2)
  City Fare Distance
1    1  360     1463
2    2  360     1448
> tail(air.data1,3)
   City Fare Distance
15   15  231      818
16   16   54       90
17   17  429     1813
```

b) -The dim(air.data2) command outputs the dimensions of the data set.

```
> city=seq(1,17,1)
> fare=c(360,360,207,111,93,141,291,183,309,300,90,162,477,84,231,54,429)
> distance=c(1463,1448,681,270,190,393,1102,578,1204,1138,184,502,1828,179,818,90,1813)
> air.data2=data.frame(city,fare,distance)
> air.data2
   city fare distance
1     1  360     1463
2     2  360     1448
3     3  207      681
4     4  111      270
5     5   93      190
6     6  141      393
7     7  291     1102
8     8  183      578
9     9  309     1204
10   10  300     1138
11   11   90      184
12   12  162      502
13   13  477     1828
14   14   84      179
15   15  231      818
16   16   54       90
17   17  429     1813
> dim(air.data2)
[1] 17  3
```

b) Creating a vector counting from 0 to 20 by 2's

```
> data3=seq(0,20,2)
> data3
 [1]  0  2  4  6  8 10 12 14 16 18 20
```

c) My preferred method is the first method, the data entry by copy and paste, because it is the quickest way to get the data into the software. The second method of manually defining the data set not only takes more time, but greatly increases the risk of making a typing error. The third method of entering data from a file path is an extremely close second to the copy and paste method as it is just as fast and has a lower risk of making typing errors.

```
> air.data3= read.delim("https://raw.githubusercontent.com/moh3nnn/stat870datasets/master/airfares.txt", hea
der=T)
> air.data3
   City Fare Distance
1     1  360     1463
2     2  360     1448
3     3  207      681
4     4  111      270
5     5   93      190
6     6  141      393
7     7  291     1102
8     8  183      578
9     9  309     1204
10   10  300     1138
11   11   90      184
12   12  162      502
13   13  477     1828
14   14   84      179
15   15  231      818
16   16   54       90
17   17  429     1813
```

d) The resulting error is that "Fare" is not found.

The calling of Fare and fare are identical.

The reason we would not have to use attach() for the data from air.data2 is because that data set was manually defined.

```
> Fare
Error: object 'Fare' not found
> air.data1$Fare
 [1] 360 360 207 111  93 141 291 183 309 300  90 162 477  84 231  54 429
> attach(air.data1)
> Fare
 [1] 360 360 207 111  93 141 291 183 309 300  90 162 477  84 231  54 429
> fare
 [1] 360 360 207 111  93 141 291 183 309 300  90 162 477  84 231  54 429
```

2) **Subsetting**

a) When typing Fare==300 into R we get 16 FALSE readings and 1 TRUE reading in the output, meaning only one of the air fares in our data set is $300.

```
> Fare==300
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

b) The output is a list of the distances of only the flights that have fares over $300.

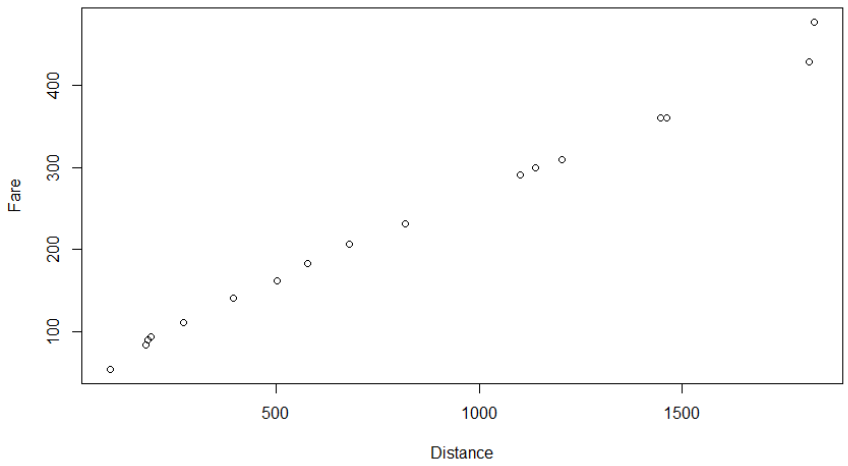
```
> Distance[Fare>300]
[1] 1463 1448 1204 1828 1813
```

c) The first entry in the bracket indicates the desired row, the second entry in the bracket indicated the desired column to choose from. When the first entry is blank, it takes the data from the entire column specified in the second entry in the bracket. When the second entry is blank, it takes the data from the entire row specified by the first entry in the bracket.

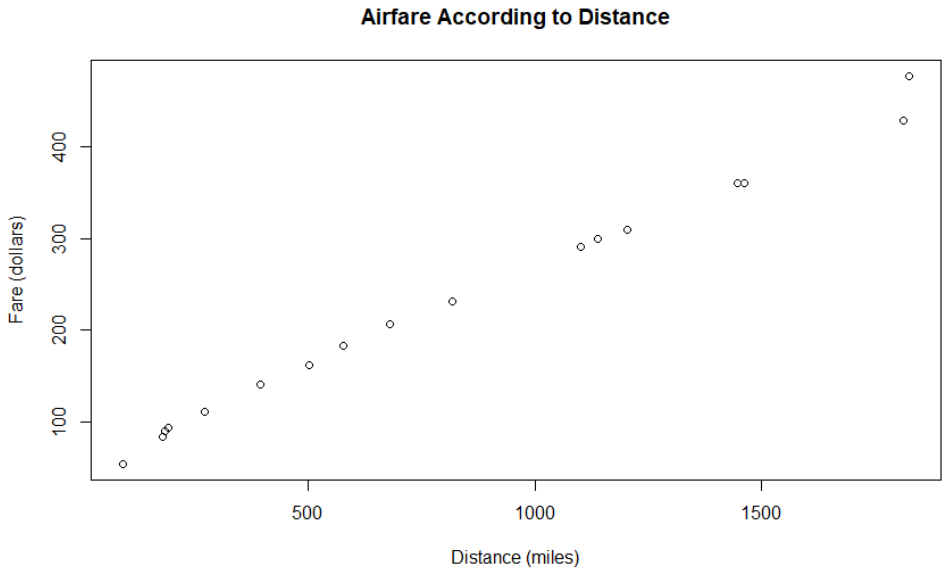
```
> air.data1[1,2]
[1] 360
> air.data1[1,3]
[1] 1463
> air.data1[2,2]
[1] 360
> air.data1[,2]
 [1] 360 360 207 111  93 141 291 183 309 300  90 162 477  84 231  54 429
> air.data1[1,]
  City Fare Distance
1    1  360     1463
> air.data1[Fare>300,]
   City Fare Distance
1     1  360     1463
2     2  360     1448
9     9  309     1204
13   13  477     1828
17   17  429     1813
```

3) **Graphing**

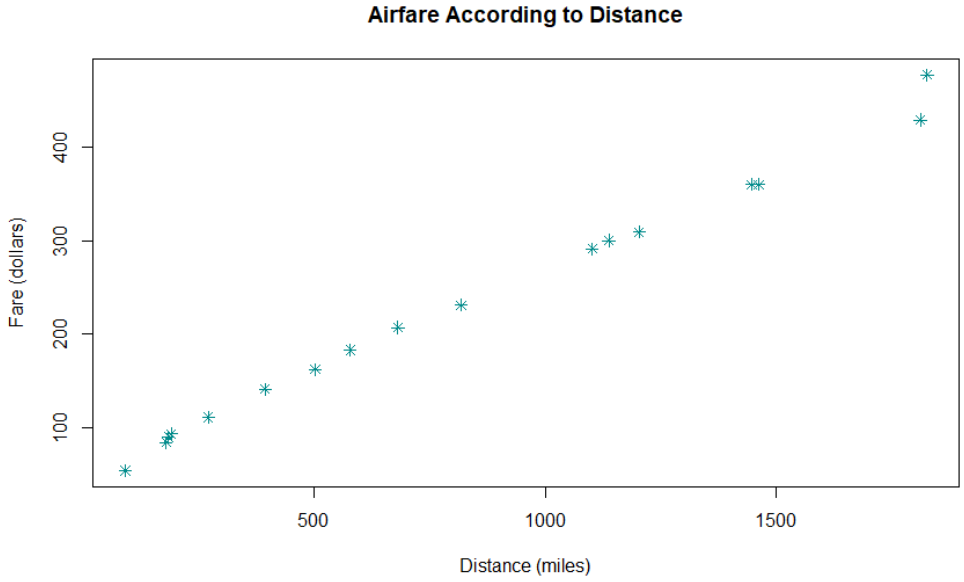
a) `> plot(Fare~Distance)`



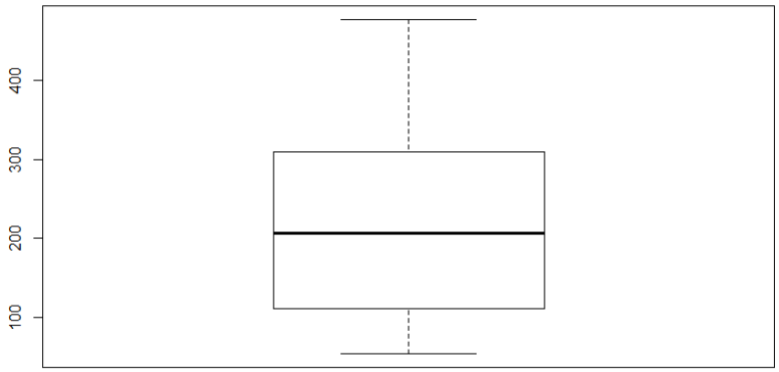
```
> plot(Fare~Distance, xlab= "Distance (miles)", ylab= "Fare (dollars)", main= "Airfare According to Distance")
```




```
> plot(Fare~Distance, xlab= "Distance (miles)", ylab= "Fare (dollars)", main= "Airfare According to Distance", col= "c
yan4", pch=8)
```

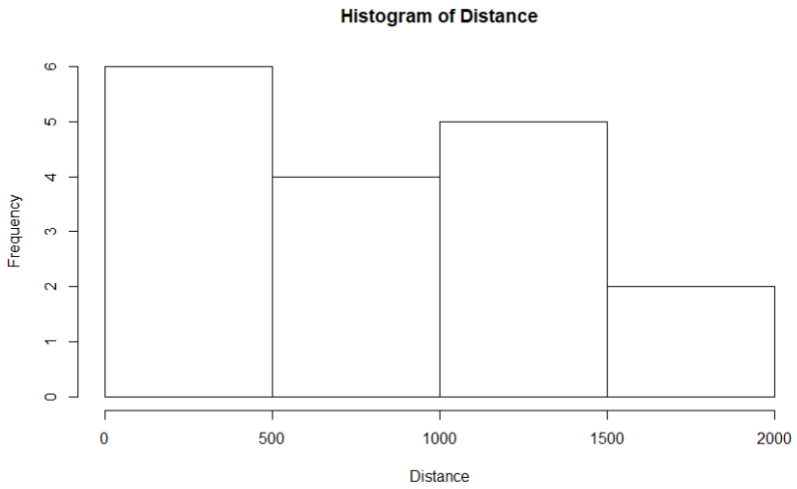


> boxplot(Fare)



The boxplot command creates a basic box plot of the desired data, in this case, Fare. It provides nothing on the x-axis, as it is not necessary and labeling the y-axis with relevant numbers for reference, but don't match up exactly (in this case) with the max, min, median, or quartiles.

> hist(Distance)



The hist command creates a basic histogram of the desired data, in this case, Distance. The x-axis is the distance and the y-axis, the frequency, and unlike the boxplot, comes with a title "Histogram of Distance." Like the boxplot, the numbers on the x-axis are not exact to the data, but general references.

4) **Summary Statistics**

a)

```
> mean(Distance)
[1] 816.5294
> sd(Distance)
[1] 588.7945
> summary(Distance)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   90.0   270.0   681.0   816.5  1204.0  1828.0
```

b)

```
> distance.data1= Distance[Fare>300]
> distance.data1
[1] 1463 1448 1204 1828 1813
> mean(distance.data1)
[1] 1551.2
> sd(distance.data1)
[1] 266.5215
> distance.data2=Distance[Fare<=300]
> distance.data2
 [1]  681  270  190  393 1102  578 1138  184  502  179  818   90
> mean(distance.data2)
[1] 510.4167
> sd(distance.data2)
[1] 361.7589
```

For the Distance when the fares are greater than $300, then mean is 1551.22 miles and the standard deviation is 266.5215 miles. For the Distance when the fares are less than or equal to $300, the mean is 510.4167 miles and the standard deviation is 361.7589 miles. The mean is greater when the fares are above $300, which isn't surprising; the farther you fly, the more expensive the ticket will be. The standard deviation of the distance when the fares are less than or equal to $300 is greater than that for the fares over $300 which could be due to the fact that there are 7 more data points for the less than or equal to category than the greater than $300 category.