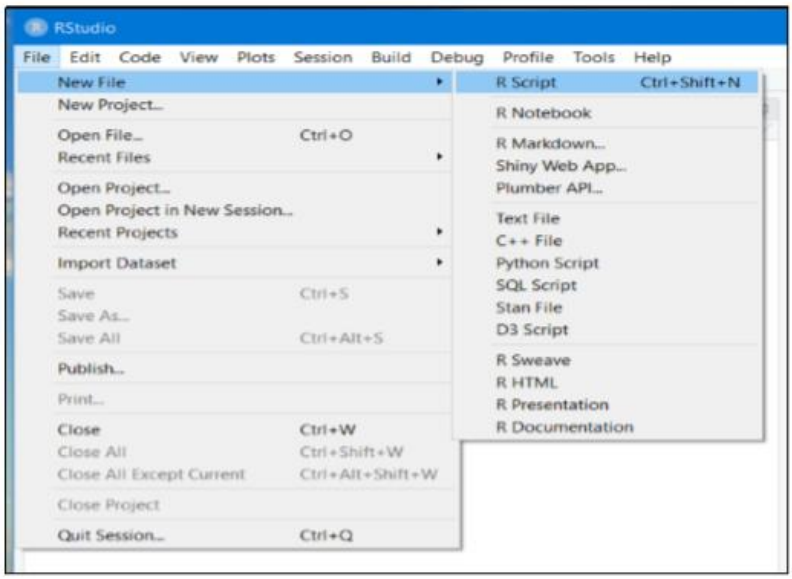
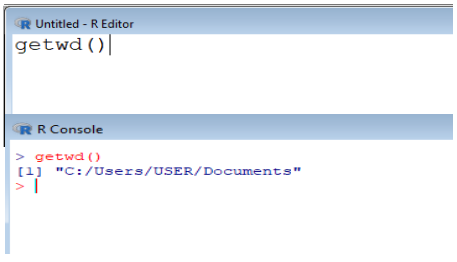
## Answers – Lab Sheet 05

- Create a folder in your desktop.
- Save the downloaded **Lab 05** folder in there.
- Unzip the folder.
- Open R Studio with new R Script (file→new file→R Script).

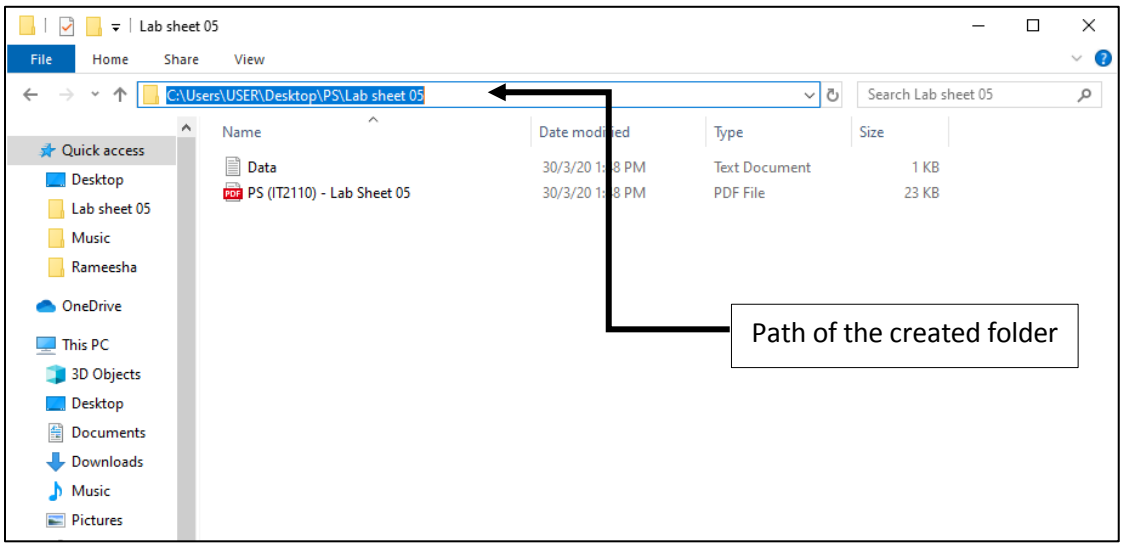



- Then check the current working directory by using **"getwd()"** command.




- To continue the lab sheet we have to upload the data set into R, we have to set the path to the particular folder where the data set is available. For that purpose we can use **"setwd()"** command as the previous lab sheet.

Go to the folder which you created inside desktop, copy the path to the folder.



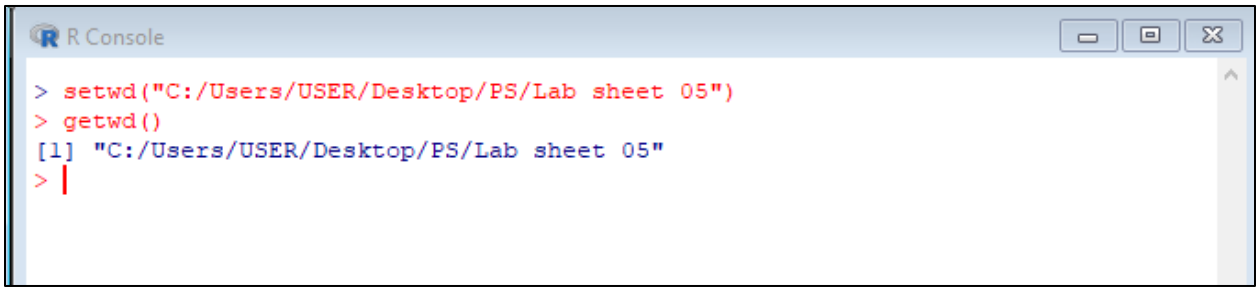

To set the path – **setwd("<*Add the path you copied*>")**

- In the path there is only one back slash. When you set the path in R studio you have to add another back slash or else instead of two back slashes you can add one forward slash.

**setwd("C:\\Users\\USER\\Desktop\\PS\\Lab sheet 05")**

**Or**

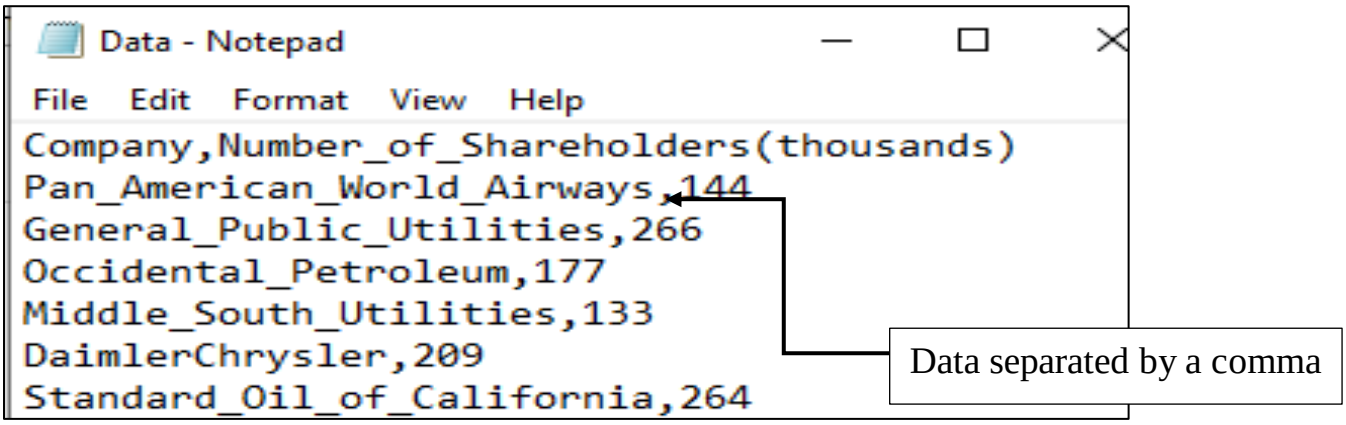
**setwd("C:/Users/USER/Desktop/PS/Lab sheet 05")**

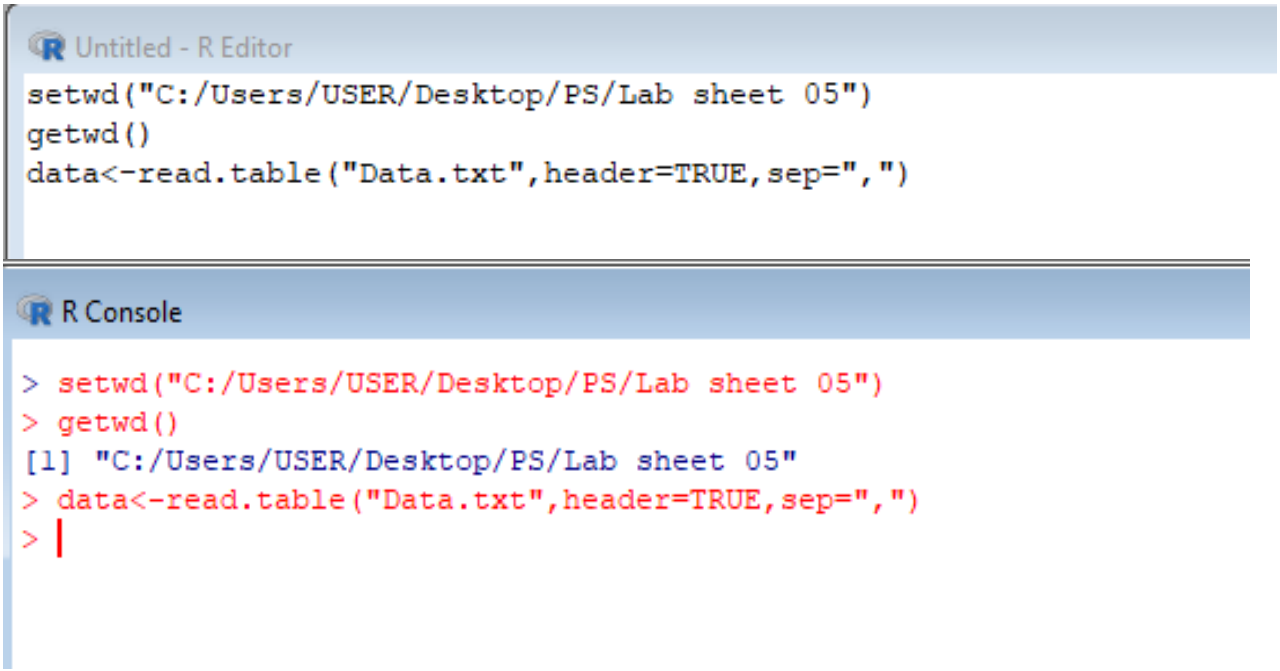


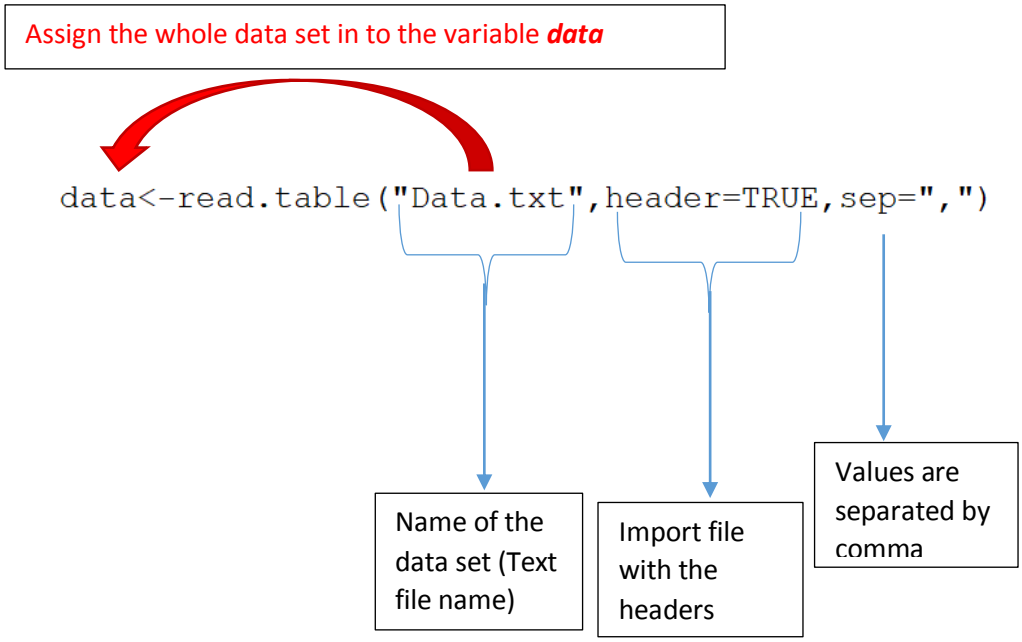
To check whether the path has set correctly to the correct folder,

**getwd()** # To get the current path

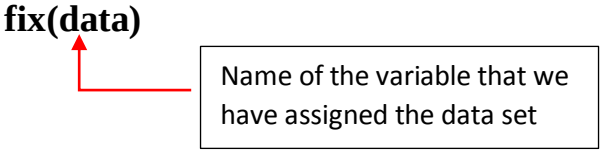
- As the data set is in a text file and all the values have been separated with a comma. We have to indicate that in the command also. So we have to use **sep= ","** parameter in the **"read.table ()"** command.

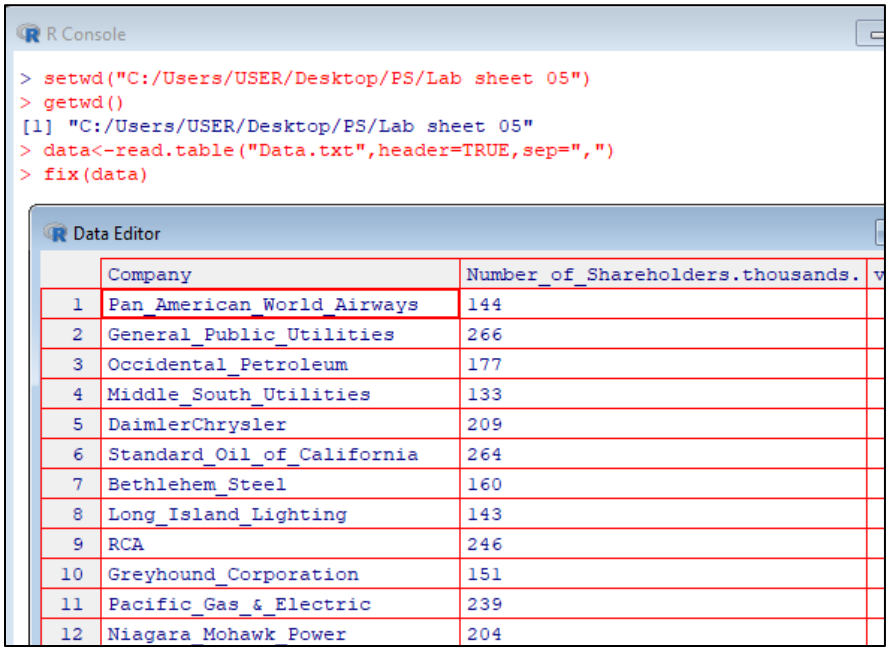





- With the **"fix()"** command, you can open the data set by using ***data editor***.



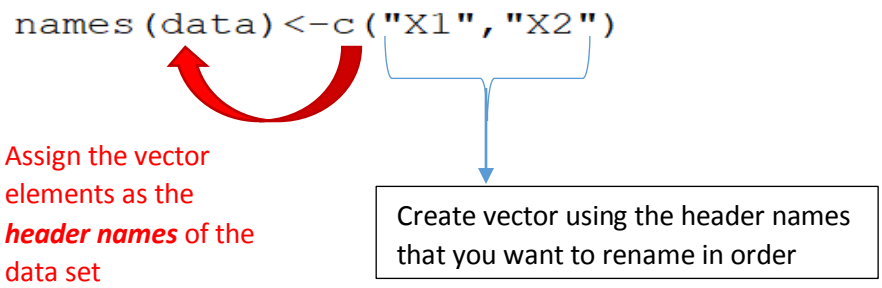




| | Company | Number_of_Shareholders.thousands. |
|---|---|---|
| 1 | Pan_American_World_Airways | 144 |
| 2 | General_Public_Utilities | 266 |
| 3 | Occidental_Petroleum | 177 |
| 4 | Middle_South_Utilities | 133 |
| 5 | DaimlerChrysler | 209 |
| 6 | Standard_Oil_of_California | 264 |
| 7 | Bethlehem_Steel | 160 |
| 8 | Long_Island_Lighting | 143 |
| 9 | RCA | 246 |
| 10 | Greyhound_Corporation | 151 |
| 11 | Pacific_Gas_&_Electric | 239 |
| 12 | Niagara_Mohawk_Power | 204 |

**NOTE:** Once you open the ***Data Editor*** make sure to ***close*** it before do the cording.

- Now you have to rename the ***Company*** as ***X1*** and ***Number of Shareholders (thousands)*** as ***X2.***

  For that you can use the following **"names()"** command,




- After that we need to attached the data, for that we need to execute the **attach("<*variable name of the data set*>")** two times.

```
> names(data)<-c("X1","X2")
> attach(data)
>
> attach(data)
The following objects are masked from data (pos = 3):

    X1, X2

>
```

If you execute "***attach()"*** once the data will be partially attached. So you need to execute it twice.

1. **Draw a histogram for the above data.**

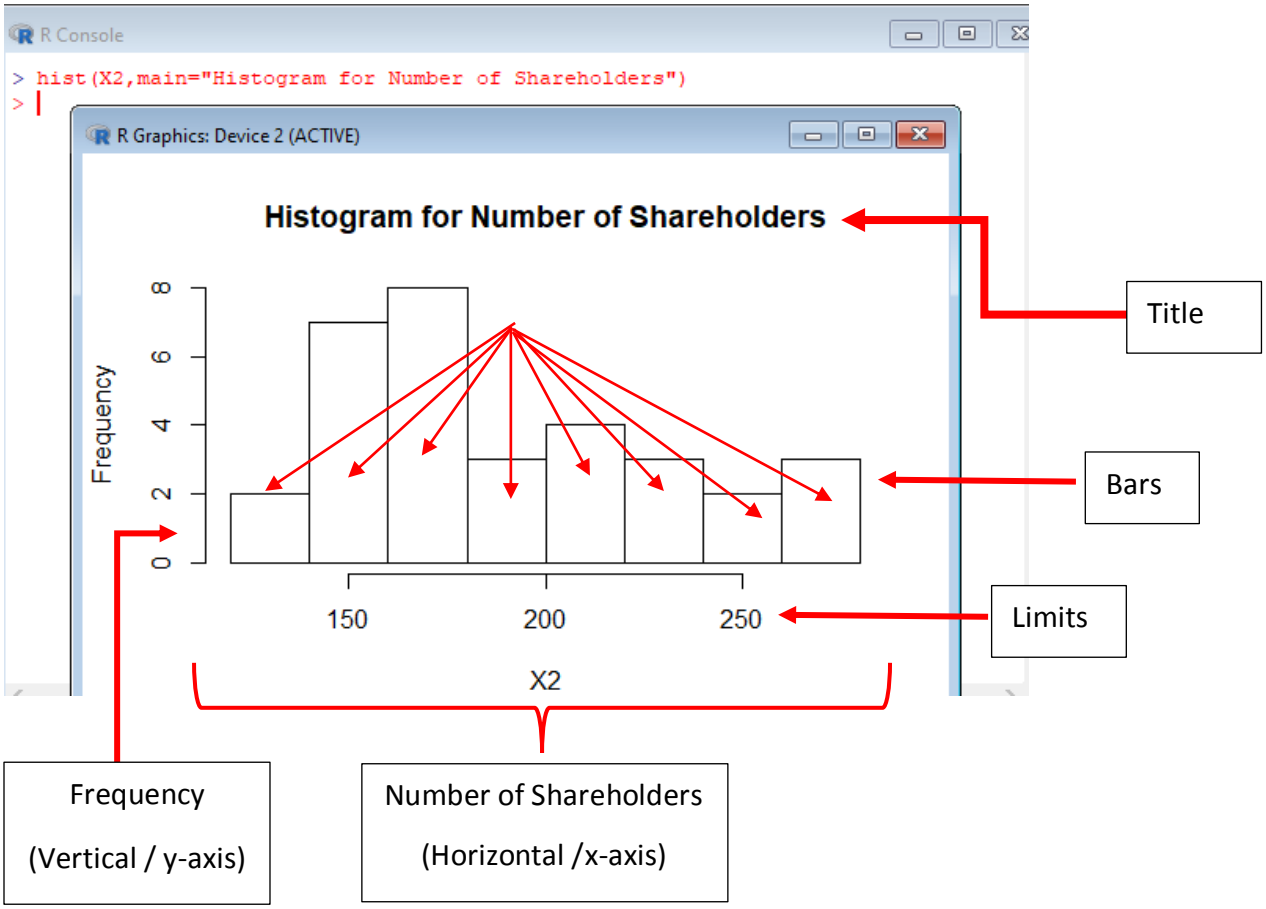
- To draw a histogram you can use **"hist()"** command.

```
##Part 01
hist(X2,main="Histogram for Number of Shareholders")
```

Set the **title** for the histogram

**main is a command that we can set a title**

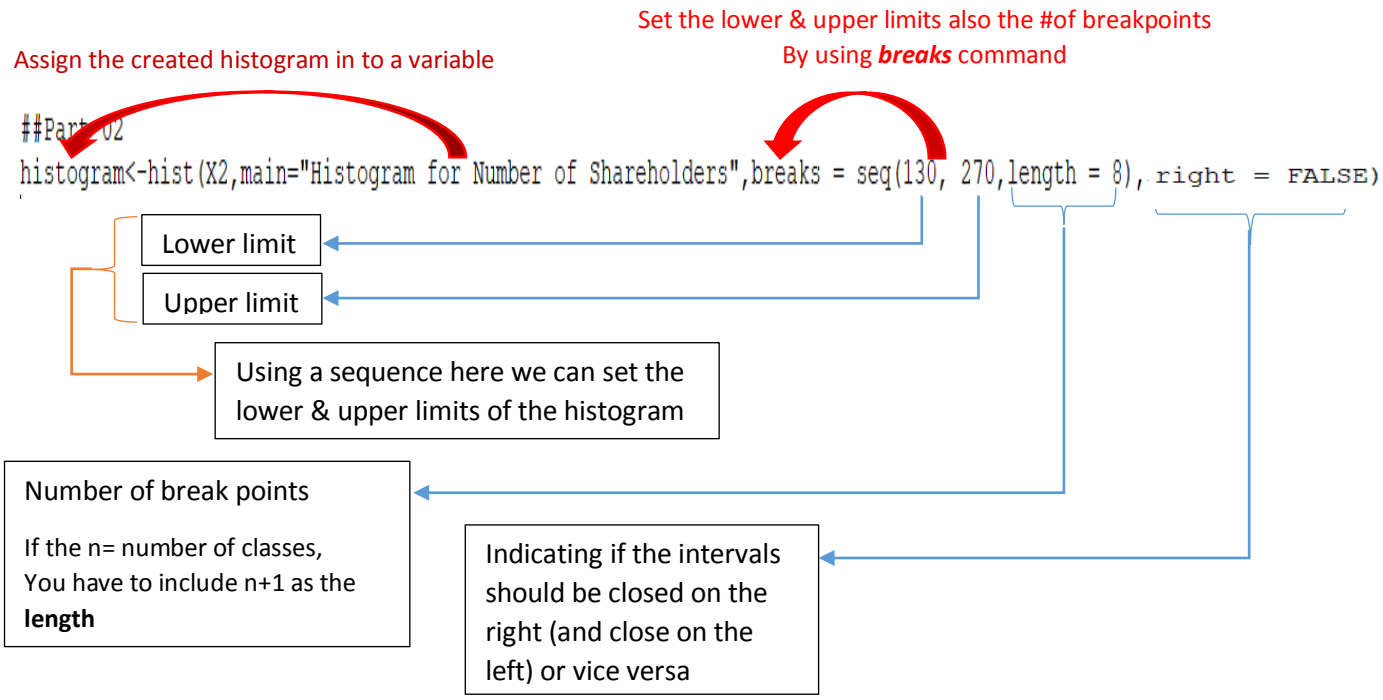
The **header name** of the data set that we are going to draw the histogram.

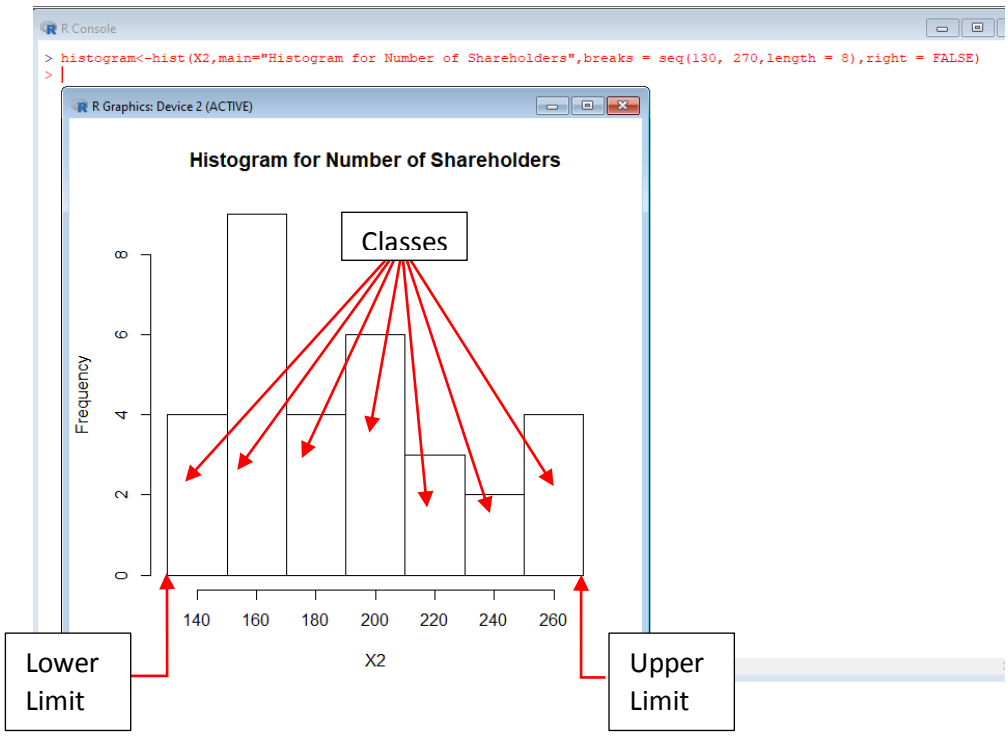


2. **Draw a histogram using seven classes where the lower limit is 130 and an upper limit of 270.**

- Now you know how to draw a histogram. According to this question you have to draw a histogram using seven classes, the lower limit should be 130 and upper limit should be 270.

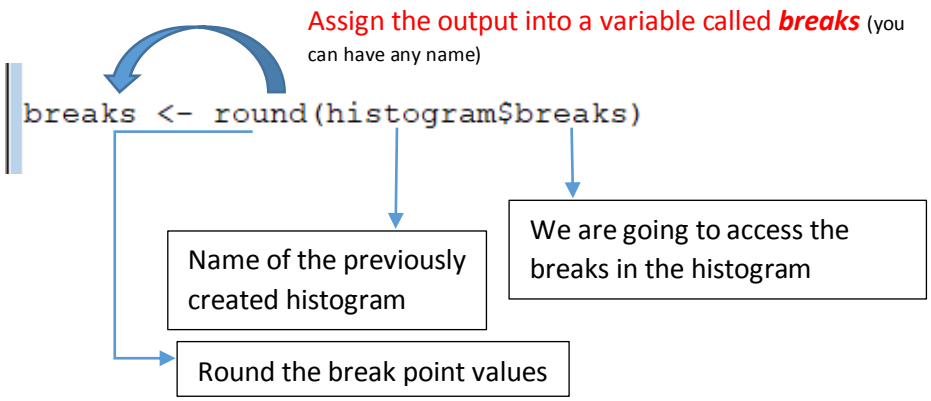
  For that purpose you can modify the previous command as below,



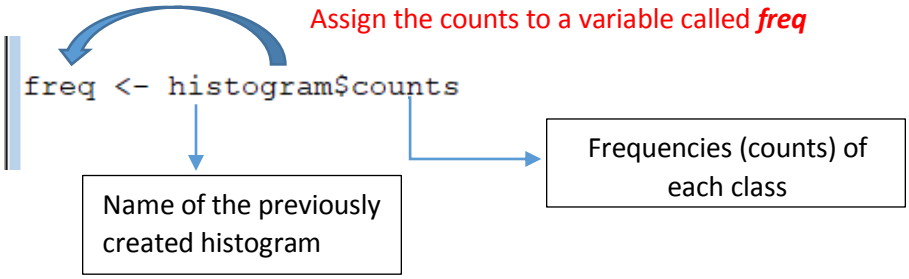



### 3. Construct the frequency distribution for the above specification.

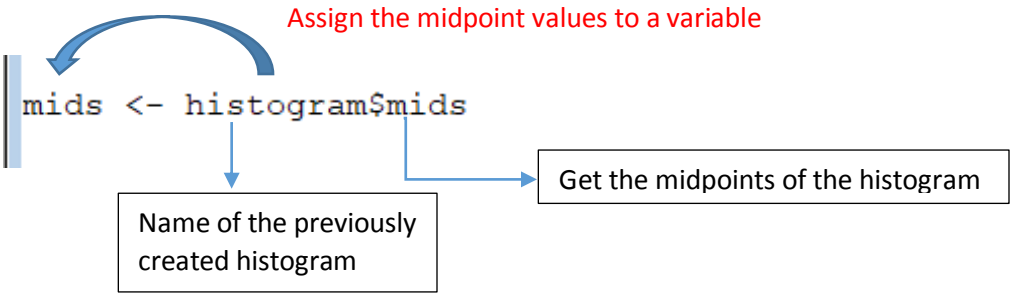
To construct the frequency distribution, we have to identify the break points correctly.



Then find the frequency of each class in the histogram.




After that find the mid points of each class




```
R Console
> breaks <- round(histogram$breaks)
> freq <- histogram$counts
> mids <- histogram$mids
> breaks
[1] 130 150 170 190 210 230 250 270
> freq
[1] 4 9 4 6 3 2 4
> mids
[1] 140 160 180 200 220 240 260
>
```

To display the frequency distribution table,

- As the first step create an empty vector called **classes**

```
classes <- c()
```

- Then have **for loop** which is repeat until the last break point.

```
for(i in 1:length(breaks)-1){
```

- Inside **for loop**, we are going to store the class boundaries in to the vector that we created at the beginning of this question.

  The output of the ***breaks*** variable contains the values as,

```
> breaks
[1] 130 150 170 190 210 230 250 270
>
```

By using for loop we are going to access the values of the ***breaks*** and store them inside the ***classes*** vector.

Let's consider about the first iteration,

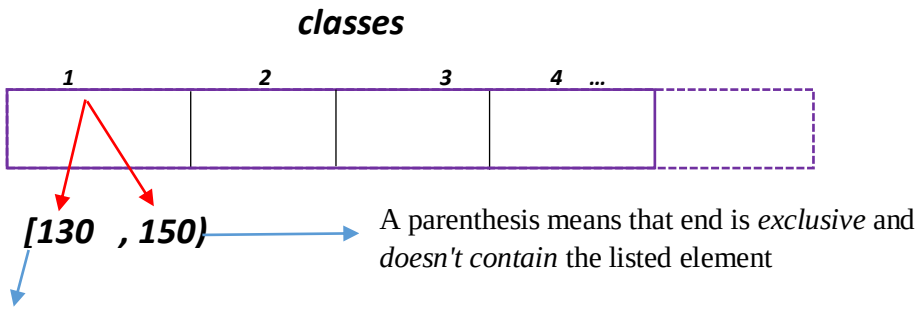
```
classes <- c()

for(i in 1:length(breaks)-1){
  classes[i] <- paste0("[", breaks[i], ",", breaks[i+1], ")")
}
```

Assign the output in to the **[i]th** index of the vector

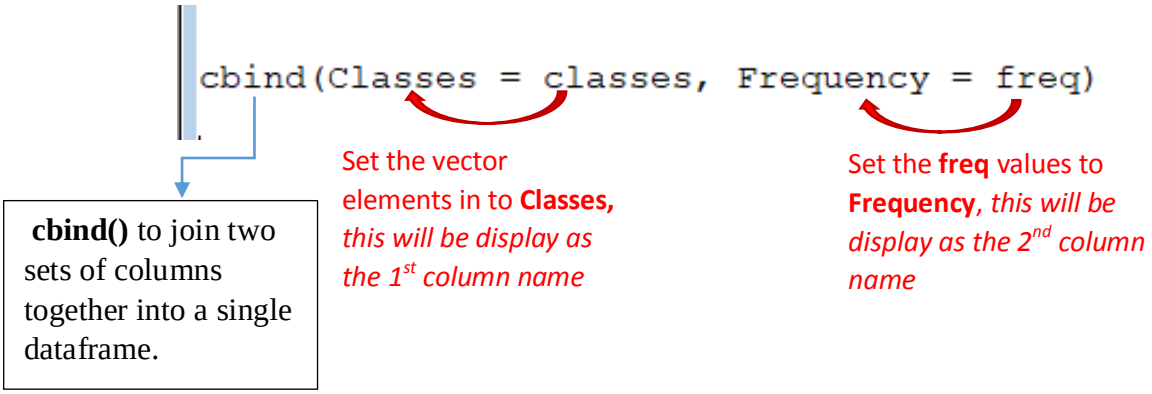
**paste** concatenates two strings **with a** space character between them. **paste0** does the same thing except **there's no space** character between the two strings being concatenated.

The **[i] th** value of the ***breaks*** variable

The **[i+1] th** value of the ***breaks*** variable



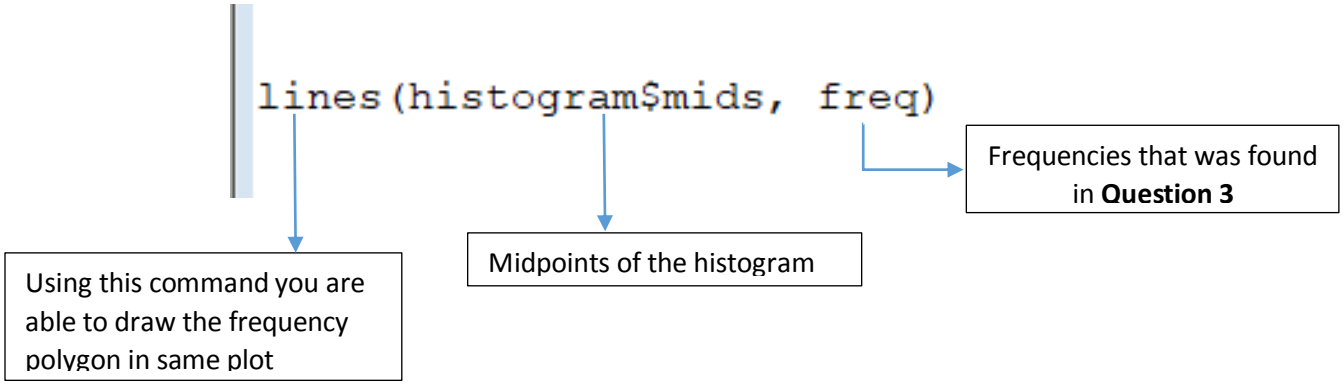
- To have the frequency table as the output,

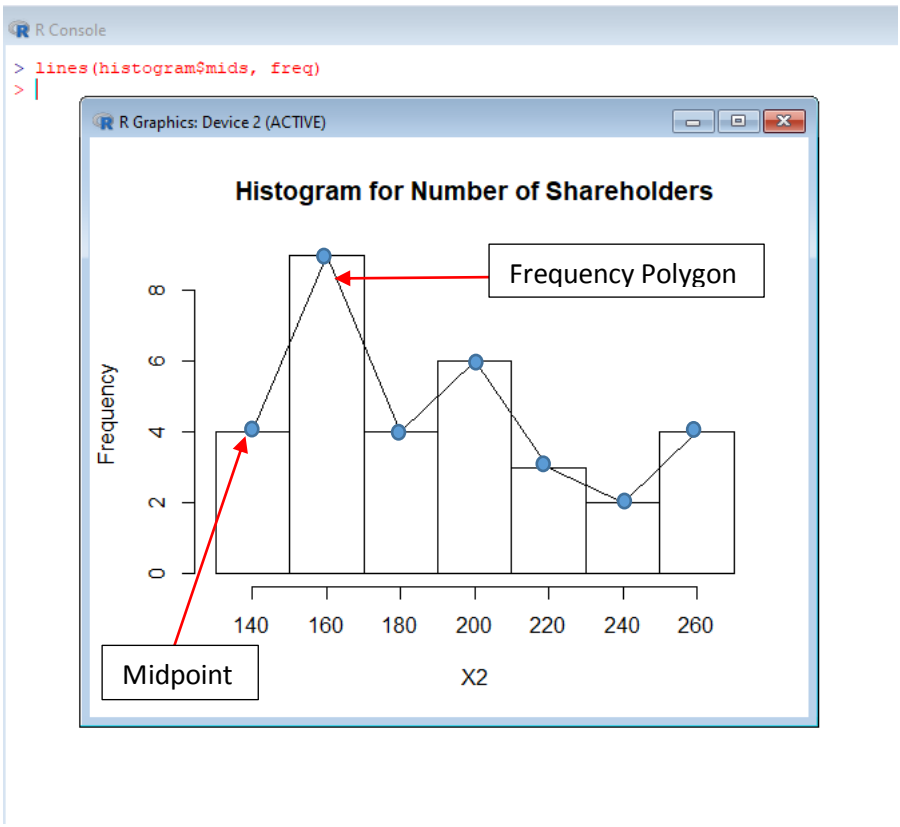



```
R Console
> classes <- c()
>
> for(i in 1:length(breaks)-1){
+   classes[i] <- paste("[", breaks[i], ",", breaks[i+1], ")")
+ }
> cbind(Classes = classes, Frequency = freq)
     Classes         Frequency
[1,] "[ 130 , 150 )" "4"
[2,] "[ 150 , 170 )" "9"
[3,] "[ 170 , 190 )" "4"
[4,] "[ 190 , 210 )" "6"
[5,] "[ 210 , 230 )" "3"
[6,] "[ 230 , 250 )" "2"
[7,] "[ 250 , 270 )" "4"
>
```

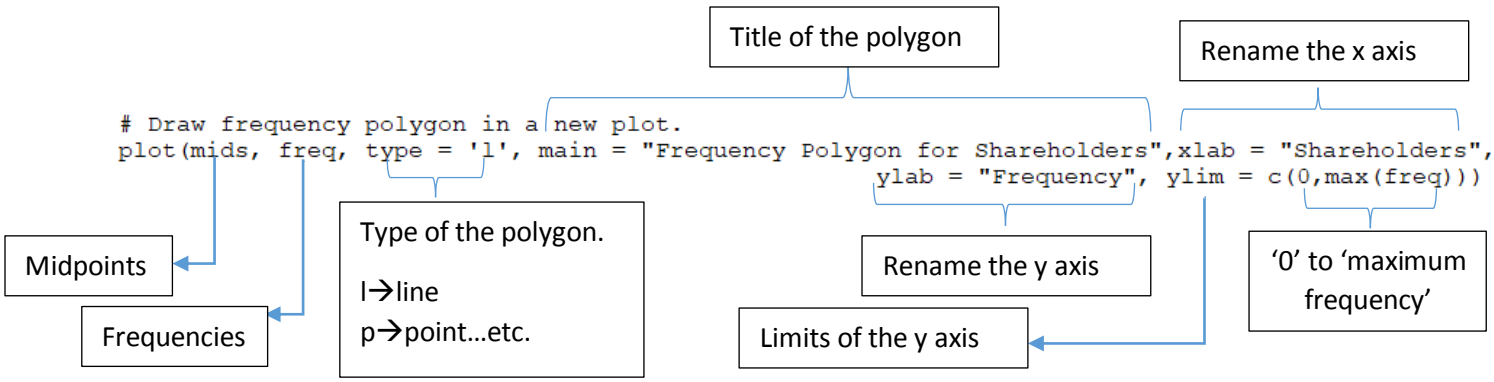
**4. Portray the distribution in the form of a frequency polygon.**

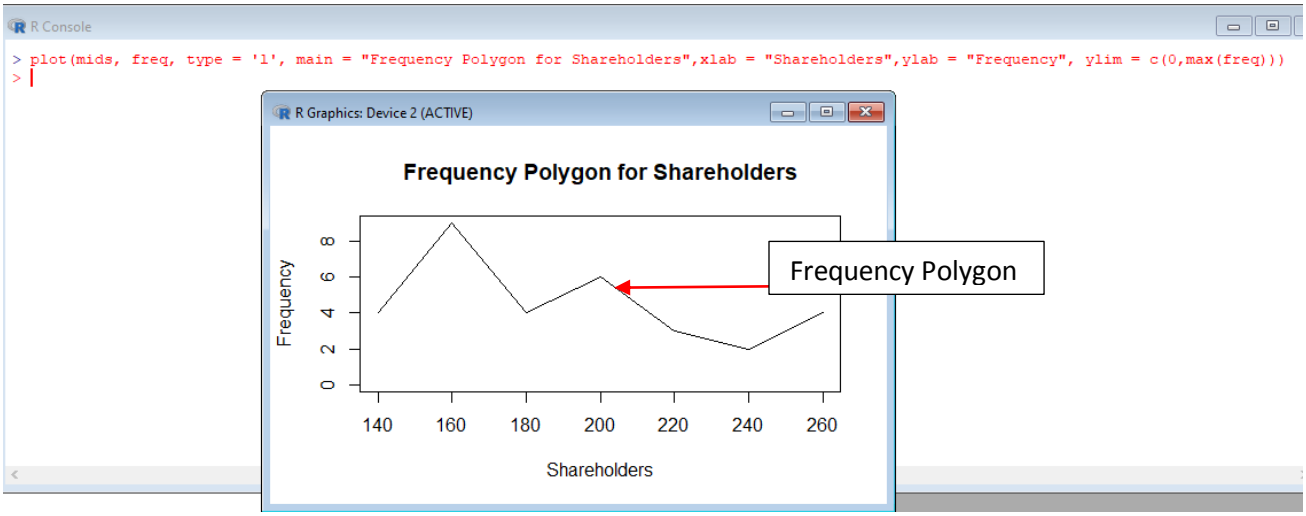
Add frequency polygon to the same plot you can use the **"lines()"** command as follows,






To draw the frequency polygon in a new plot you can use the **"plot()"** command.





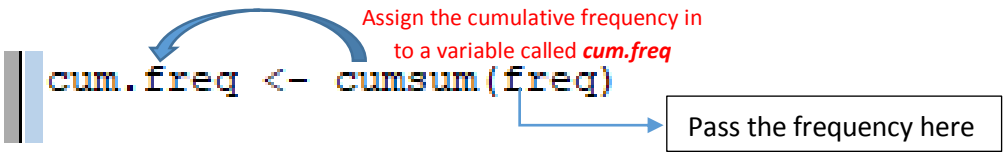

5. **Portray the distribution in a less-than cumulative frequency polygon.**

Cumulative frequency: The cumulative frequency is the running total of the frequencies.

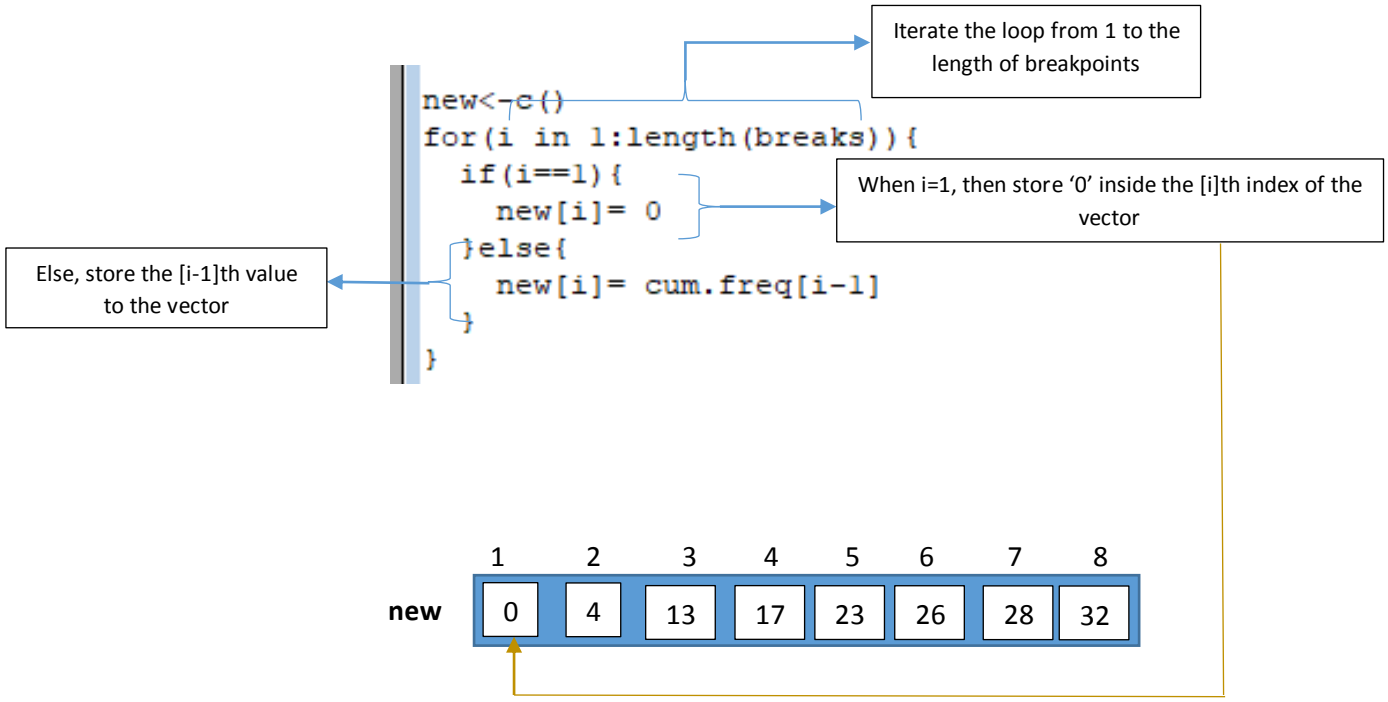
Cumulative frequency will be calculated as follows,

| Classes | frequancy | | Cum.Frequancy |
|---|---|---|---|
| 130-150 | 4 | | 4 |
| 150-170 | 9 | | 13 |
| 170-190 | 4 | | 17 |
| 190-210 | 6 | | 23 |
| 210-230 | 3 | | 26 |
| 230-250 | 2 | | 28 |
| 250-270 | 4 | | 32 |
| | | | |

- To find the cumulative frequency in R you can use "**cumsum()**" command.



- To draw the cumulative frequency polygon,
  - First you have to store the values of ***cum.freq*** in to a vector called ***new.***
  - Then to store the cumulative frequencies in the ***cum.freq*** in to the ***new*** vector, we can use for loop.

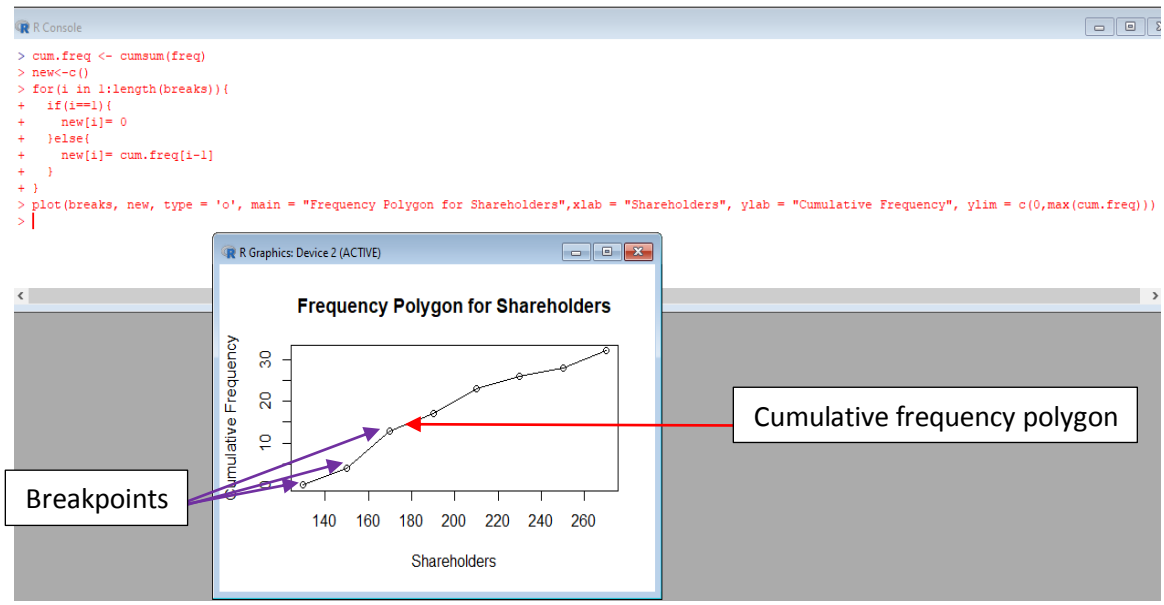



To plot the cumulative frequency again you can use the "**plot()**" command. Here *instead* of *midpoints* and *frequency* you have to pass the **breakpoints** and the **vector** which contains the cumulative frequency values.
Rest of the parameters are as previous.

```
plot(breaks, new, type = 'o', main = "Frequency Polygon for Shareholders",
xlab = "Shareholders", ylab = "Cumulative Frequency", ylim = c(0,max(cum.freq)))
```

Breakpoints

Vector




- To get the cumulative frequency table as the output, you can use the **"cbind()"** command.

```
cbind(Upper = breaks, CumFreq = new)
```

Set the breakpoints in to **Upper**, *this will display as the 1st column name*

Set the vector in to **CumFreq**, *this will display as the 2nd column name*

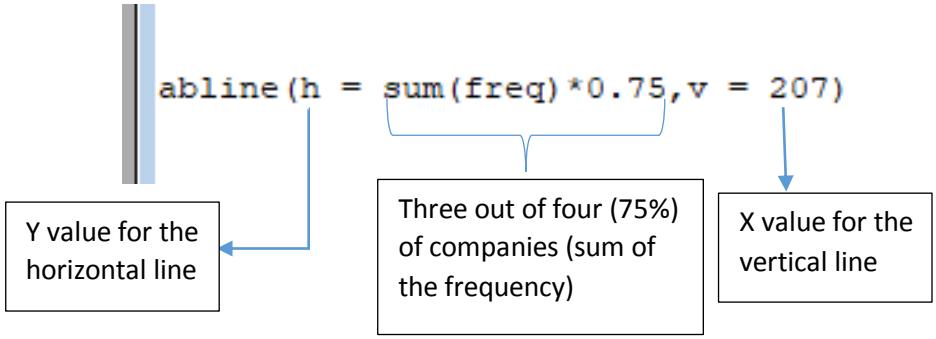
```
R Console
> cbind(Upper = breaks, CumFreq = new)
     Upper CumFreq
[1,]   130       0
[2,]   150       4
[3,]   170      13
[4,]   190      17
[5,]   210      23
[6,]   230      26
[7,]   250      28
[8,]   270      32
>
```

6. **Based on the polygon, three out of four (75%) of the companies have how many shareholders or less?**

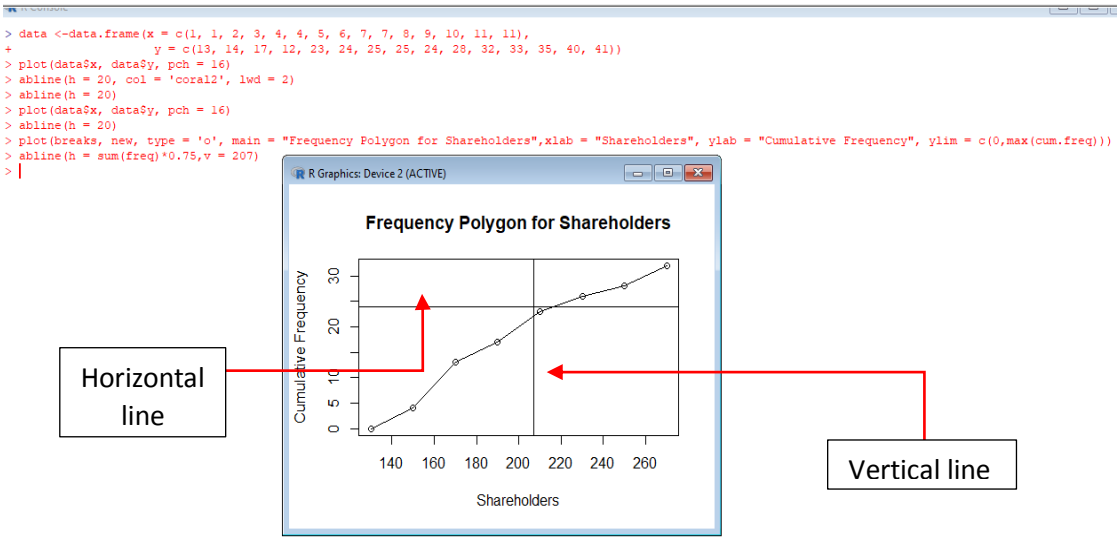
Here you can use **"abline()"** command ,it can be used to add one or more straight lines to a plot in R. The basic syntax is of **"abline()"** is as follows:

**abline(a=NULL, b=NULL, h=NULL, v=NULL, …)**

- **a, b:** single values that specify the intercept and slope of the line
- **h:** the y-value for the horizontal line
- **v:** the x-value for the vertical line




**NOTE:** before you execute this line, you have to execute the command that plot the cumulative frequency polygon previously. Also you need to find "v" value before giving to the command.



**NOTE**: As the final step, you have to detach the data set. For that you can use "**detach()**" command.

- The detach function can be used to:

Remove the attachment of a data, which was previously attached with the attach function.