3]
R code:

```
data("iris")          # load dataset iris
iris                  # view the dataset
dim(iris)             # shows we have 150 sample rows and 5 columns as dimensions
attributes(iris)      # to get an overall idea about classes, samples and their data types
summary(iris)         #Summarize data me

barplot(iris$Sepal.Width, xlab="Sample number",ylab="Sepal.Width")  #Bar plot for sepal width

hist(iris$Petal.Width,main="Histogram for Petal Width",
    xlab="Petal Width")                                              #Histogram for Petal length
```

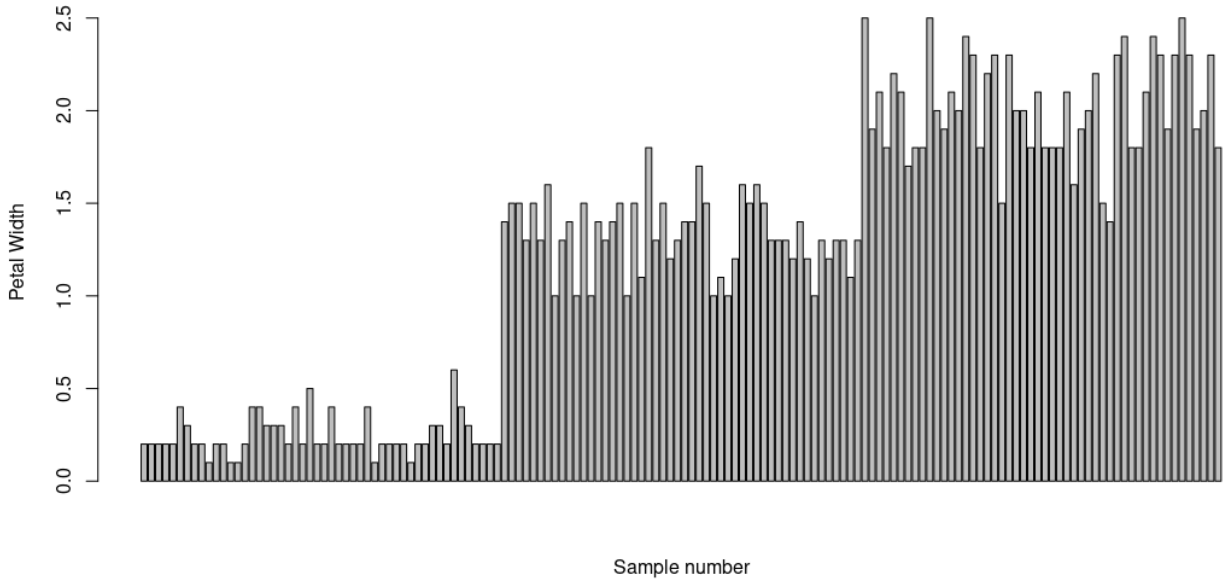
A **bar Plot** is a diagrammatic comparison of discrete variables.

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

Let us take **PETAL WIDTH** as the feature of interest.

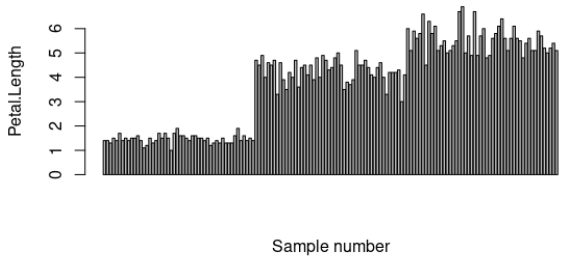
Plotting Bar Graph

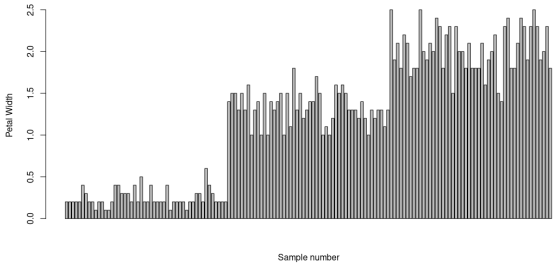
R code: **barplot**(iris$Sepal.Length, xlab="Sample number",ylab="Sepal.Length")

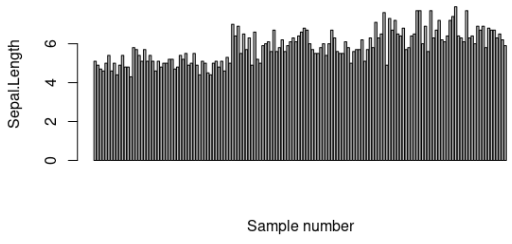


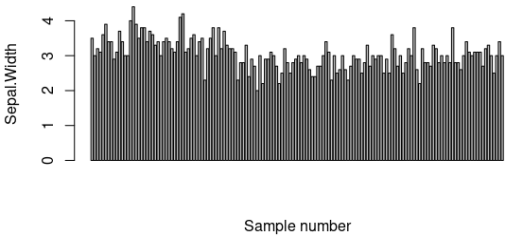
From the bar graph we can clearly see the **graph being segmented into 3 segments**, which is the difference in range of petal width with species, **thus it is a good feature to classify species upon.**

Bar Plot for each feature









As we notice sepal plots have no clear segments. While Petal plots show 3 segments

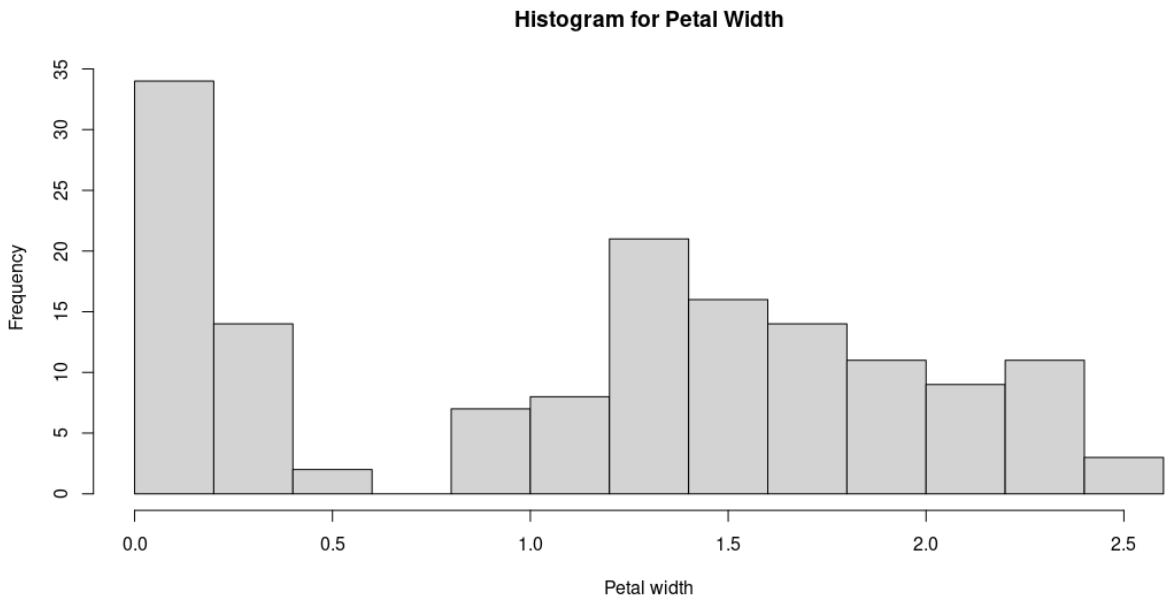
INFERENCE- This shows that PETAL (petal length or width) is a GOOD feature to classify Species while sepal is not.

- **Plotting Histogram**

A histogram represents the frequency distribution of continuous variables.

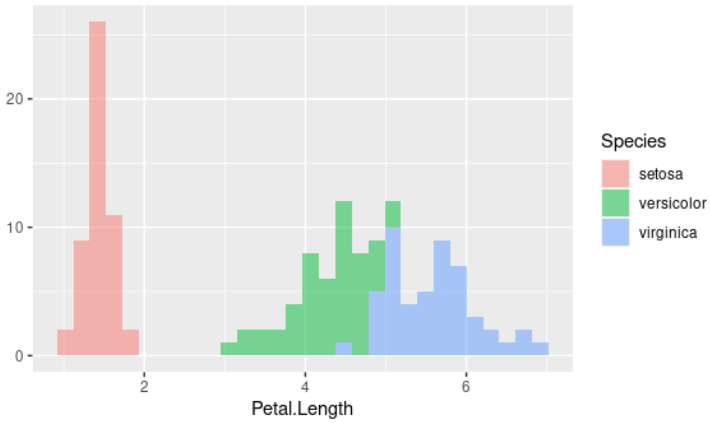
R CODE :

hist(iris$Petal.Width,main="Histogram for Petal Width", xlab="Petal Width") #plot histogram



require(ggplot2) #import ggplot2 package

qplot(Petal.Length, data=iris, geom='histogram', fill=Species, alpha=I(0.5))#plot color histogram



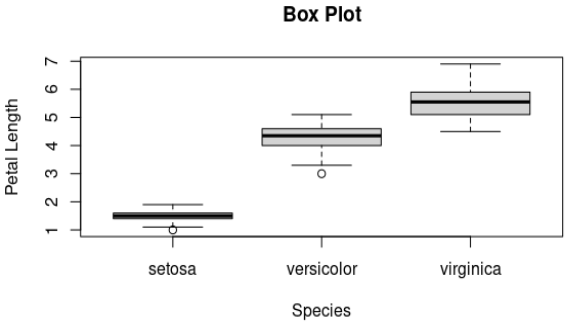
Plotting frequency of occurrence of petal width, we notice the plot segmenting into 2 parts.
For the whole dataset, Petal width varying roughly from 0.0 to 2.5.
we **notice a segmentation around 0.5**. Now from the BAR plot of Petal width we observe 0.5 is upper limit for SETOSA specie.

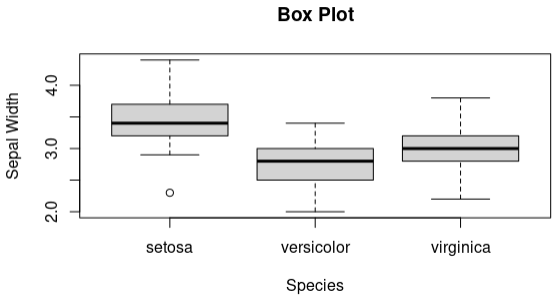
Hence this plot infers, most of SETOSA petal width is below 0.5 while for other two species it is greater than roughly 1 but it's non conclusive to differentiate them directly(due to high variance in data for them).

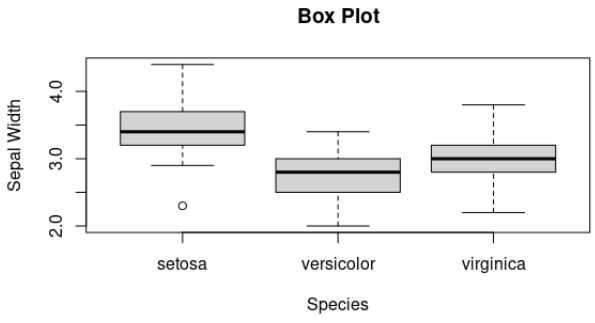
4]
To Identify OUTLIERS(an observation that lies an abnormal distance from other values in a random sample from a population)
We will use **BOXPLOT** to identify them.
Based on five parameter summary
(“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”)
we get a comprehensive glimpse of the dataset.

R code:

```
boxplot(Petal.Length ~ Species, data=iris,
        main="Box Plot",
        xlab="Species",
        ylab="Sepal Length")          #Box plotting Petal Length for each specie
```







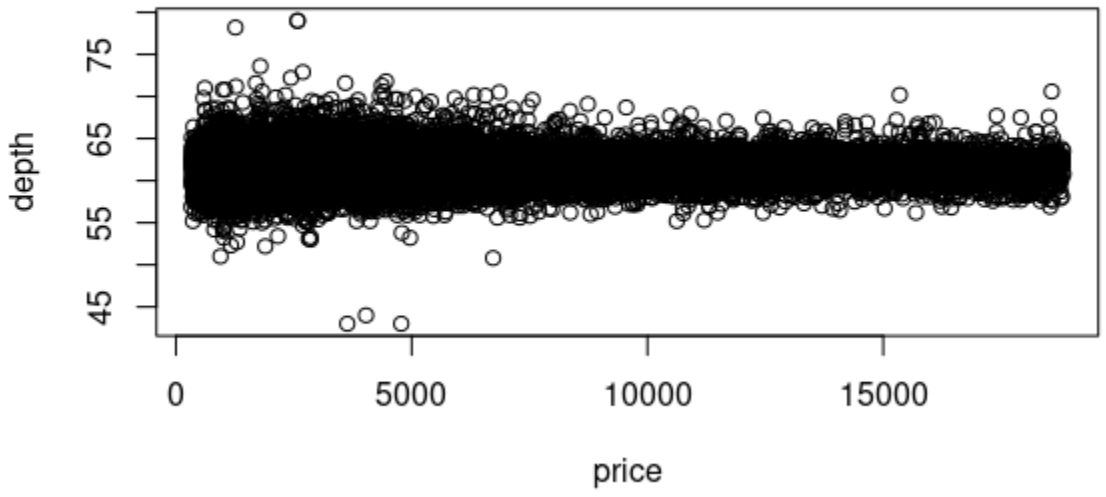
The circles outside the box plot represent the outliers.

5]

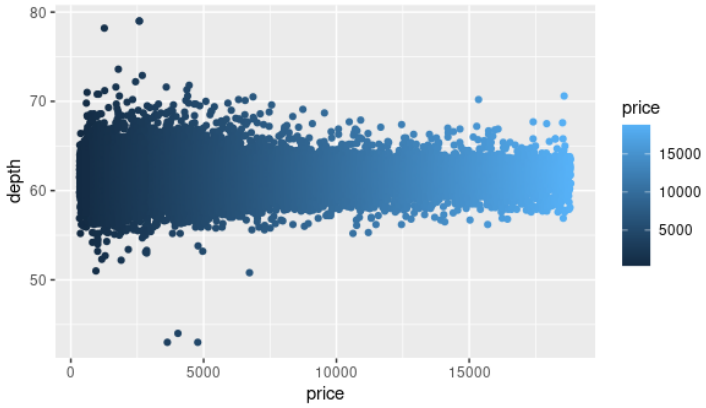
R code:

```
library(ggplot2)                    #import ggplot package
data(diamonds)                      #load dataset
summary(diamonds)                   #summarize the statistics of the dataset
dim(diamonds)                       #point out dimensions of dataset

plot(diamonds$price,diamonds$depth,xlab="price",ylab="depth")    #scatter plot price vs depth
```

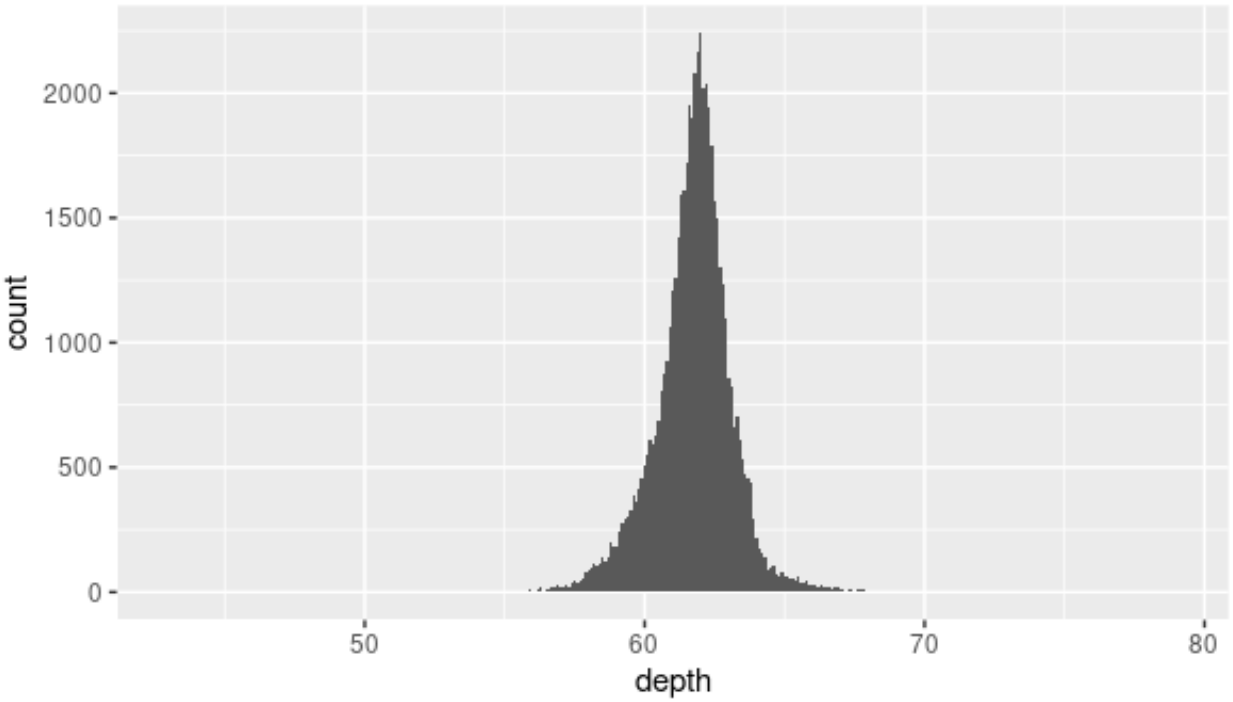


Another form,
R CODE: **ggplot**(diamonds, aes(x=price, y=depth, color=price)) + geom_point()



Now to find the depth where most diamonds are found:
We plot a HISTOGRAM to measure frequency of diamonds with depth

R CODE:
ggplot(diamonds, aes(x=depth)) + geom_histogram(binwidth=0.1)
binwidth adjusts the plot thinness lowering it makes each class range thinner and plot finer.



The depth where most diamonds are found is the x value corresponding to the the PEAK

6]
R CODE:

data(USArrests) #load data
summary(USArrests) #summarize statistics of the data
dim(USArrests) #print dimensions

To abbreviate state names, we take use of the function:

state.abb[which(state.name==”Ohio”)] returns “OH”

df <- USArrests #make a copy of dataset as df
rownames(df) #display default names of state

```
$row.names
 [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"       "California"
 [6] "Colorado"       "Connecticut"    "Delaware"       "Florida"        "Georgia"
[11] "Hawaii"         "Idaho"          "Illinois"       "Indiana"        "Iowa"
[16] "Kansas"         "Kentucky"       "Louisiana"      "Maine"          "Maryland"
[21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"    "Missouri"
[26] "Montana"        "Nebraska"       "Nevada"         "New Hampshire"  "New Jersey"
[31] "New Mexico"     "New York"       "North Carolina" "North Dakota"   "Ohio"
[36] "Oklahoma"       "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
[41] "South Dakota"   "Tennessee"      "Texas"          "Utah"           "Vermont"
[46] "Virginia"       "Washington"     "West Virginia"  "Wisconsin"      "Wyoming"
```

abbr <- state.abb[which(state.name==rownames(df))]
#passing array of default names to the function then saving them to array abbr

```
Abbr -->  "AL" "AK" "AZ" "AR" "CA" "CO" "CT" "DE" "FL" "GA" "HI"......
```

for (x in 1:50) {
  rownames(df)[x] <- abbr[x] #for each i(state) replace new abbreviation with old name
}
rownames(df) #display new row names

```
> rownames(df)
 [1] "AL" "AK" "AZ" "AR" "CA" "CO" "CT" "DE" "FL" "GA" "HI" "ID" "IL" "IN" "IA" "KS" "KY" "LA"
[19] "ME" "MD" "MA" "MI" "MN" "MS" "MO" "MT" "NE" "NV" "NH" "NJ" "NM" "NY" "NC" "ND" "OH" "OK"
[37] "OR" "PA" "RI" "SC" "SD" "TN" "TX" "UT" "VT" "VA" "WA" "WV" "WI" "WY"
>
```

b)

To Select the states that contain the letter ‘b’
We use function grep which returns all elements which contain the search term.
grep("search term", state.name)

R CODE:

data(USArrests)
grep("b", rownames(USArrests))

Returns:

```
1 27
Which correspond to "Alabama" and "Nebraska"
```

c)

R CODE:

```
data(USArrests)
rownames(USArrests)

b = c(0,0,0,0,0)                # this vector is our counter b[1] is count for a, b[2] for b etc...
M <- c("a","e","i","o","u")     # corresponding label to b[i]


for(i in 1:50){                                                    # loop through each state
character_array <- unlist(strsplit(rownames(USArrests)[i], "")) # convert string to character array
for(j in character_array ){                                        #loop for each character in the array
  if(j=="a" || j=="A")  b[1] <- b[1] + 1                  # if character is a or A, increment its counter b[1] by 1
  if(j=="e" || j=="E")  b[2] <- b[2] + 1
  if(j=="i"  || j=="I")    b[3] <- b[3] + 1                 #similarly for each vowel
  if(j=="o" || j=="O")  b[4] <- b[4] + 1
  if(j=="u" || j=="U")  b[5] <- b[5] + 1
}
}
b                                                  # return b to see the net counts of vowel

barplot(b,names.arg=M,,xlab="Vowel",ylab="Count",col="blue",
        main="Vowel count"
)                                                  #plot bar graph showing b as the count for each vowel
```

RESULT:

```
61 28 44 36 8
A  E  I  O  U
```