

# VPM's B.N. Bandodkar College Of Science

## Practical No.3

**AIM:** Practical of Principal Component Analysis(PCA).

### **Theory:**

PCA is a very popular method of dimensionality reduction because it provides a way to easily reduce the dimensions and is easy to understand. For this reason, PCA has been used in various applications from image compression to complex gene comparison. While using PCA, one should keep in mind its limitations well.

PCA is very sensitive to the scale of the data. It will create an initial basis in the direction of the largest variance in the data. Moreover, PCA applies a transformation over the data where all new components are orthogonal. The new features may not be interpretable in business.

Another limitation of PCA is the reliance on the mean and variance of the data. If the data has a relationship in higher dimensions such as kurtosis and skewness then PCA may not be the right technique to use on the data. In situations when the features are already orthogonal to each other and are uncorrelated, PCA will not produce any useful results except ordering the features in decreasing order of their variances.

PCA is very useful in situations when the data at hand is very large. Example, in case of image compression, PCA can be used to store the image in the first few hundred components and use less number of pixels.

We can implement the same in R programming language.

The **princomp()** function in R calculates the principal components of any data. We will also compare our results by calculating eigenvectors and eigenvalues separately. Let's use the **IRIS dataset**.

Let's start by loading the dataset.

# Taking the numeric part of the IRIS data

```
> data_iris <- iris[1:4]
```

The iris dataset having 150 observations (rows) with 4 features.

Let's use the **cov()** function to calculate the covariance matrix of the loaded iris data set.

# Calculating the covariance matrix

```
> Cov_data <- cov(data_iris )
```

The next step is to calculate the eigenvalues and eigenvectors.

We can use the **eigen()** function to do this automatically for us.

# Find out the eigenvectors and eigenvalues using the covariance matrix

```
> Eigen_data <- eigen(Cov_data)
```

We have calculated the Eigen values from the data. We will now look at the PCA function **princomp()** which automatically calculates these values.

Let's calculate the components and compare the values.

# Using the inbuilt function

## VPM's B.N. Bandodkar College Of Science

```
> PCA_data <- princomp(data_iris ,cor="False")
```

```
# Let's now compare the output variances
```

```
> Eigen_data$values
```

**Output:**

```
[1] 4.22824171 0.24267075 0.07820950 0.02383509
```

```
> PCA_data$sdev^2
```

```
  Comp.1  Comp.2  Comp.3  Comp.4  
4.20005343 0.24105294 0.07768810 0.02367619
```

There is a slight difference due to squaring in PCA\_data but the outputs are more or less similar. We can also compare the eigenvectors of both models.

```
> PCA_data$loadings[,1:4]
```

```
      Comp.1  Comp.2  Comp.3  Comp.4  
Sepal.Length 0.36138659 0.65658877 0.58202985 0.3154872  
Sepal.Width  -0.08452251 0.73016143 -0.59791083 -0.3197231  
Petal.Length 0.85667061 -0.17337266 -0.07623608 -0.4798390  
Petal.Width 0.35828920 -0.07548102 -0.54583143 0.7536574
```

```
> Eigen_data$vectors
```

```
      [,1]      [,2]      [,3]      [,4]  
[1,] 0.36138659 -0.65658877 -0.58202985 0.3154872  
[2,] -0.08452251 -0.73016143 0.59791083 -0.3197231  
[3,] 0.85667061 0.17337266 0.07623608 -0.4798390  
[4,] 0.35828920 0.07548102 0.54583143 0.7536574
```

This time the eigenvectors calculated are same and there is no difference.

Let us now understand our model. We transformed our 4 features into 4 new orthogonal components. To know the importance of the first component, we can view the summary of the model.

```
> summary(PCA_data)
```

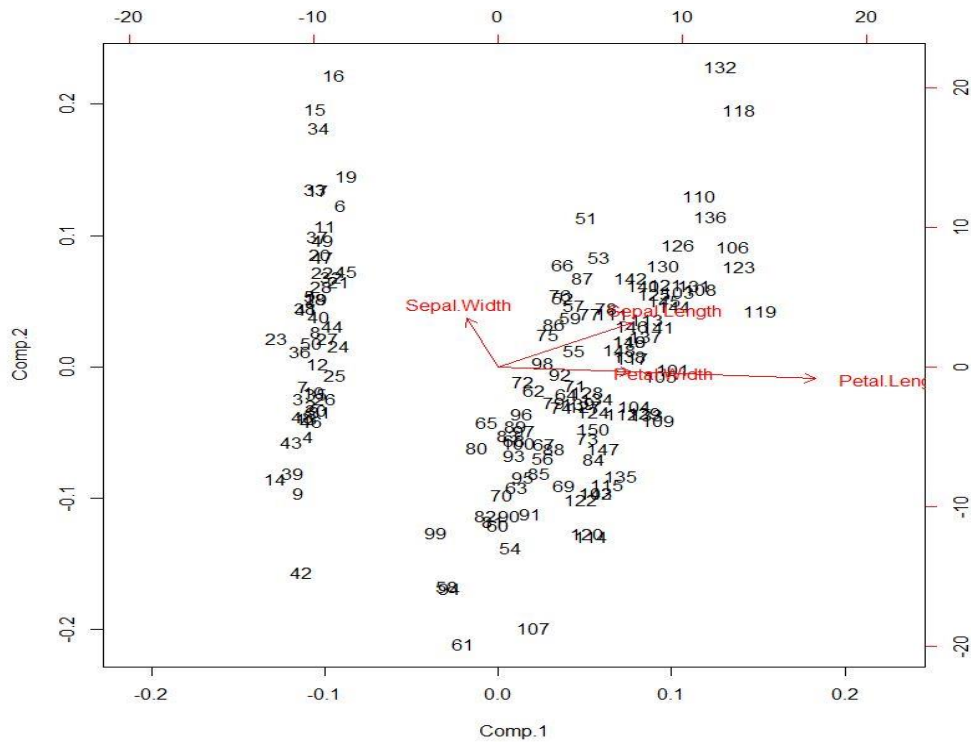
Importance of components:

```
      Comp.1  Comp.2  Comp.3  Comp.4  
Standard deviation 2.0494032 0.49097143 0.27872586 0.153870700  
Proportion of Variance 0.9246187 0.05306648 0.01710261 0.005212184  
Cumulative Proportion 0.9246187 0.97768521 0.99478782 1.000000000
```

From the Proportion of Variance, we see that the first component has an importance of **92.5%** in predicting the class while the second principal component has an importance of **5.3%** and so on. This means that using just the first component instead of all the 4 features will make our model accuracy to be about **92.5%** while we use only one-fourth of the entire set of features.

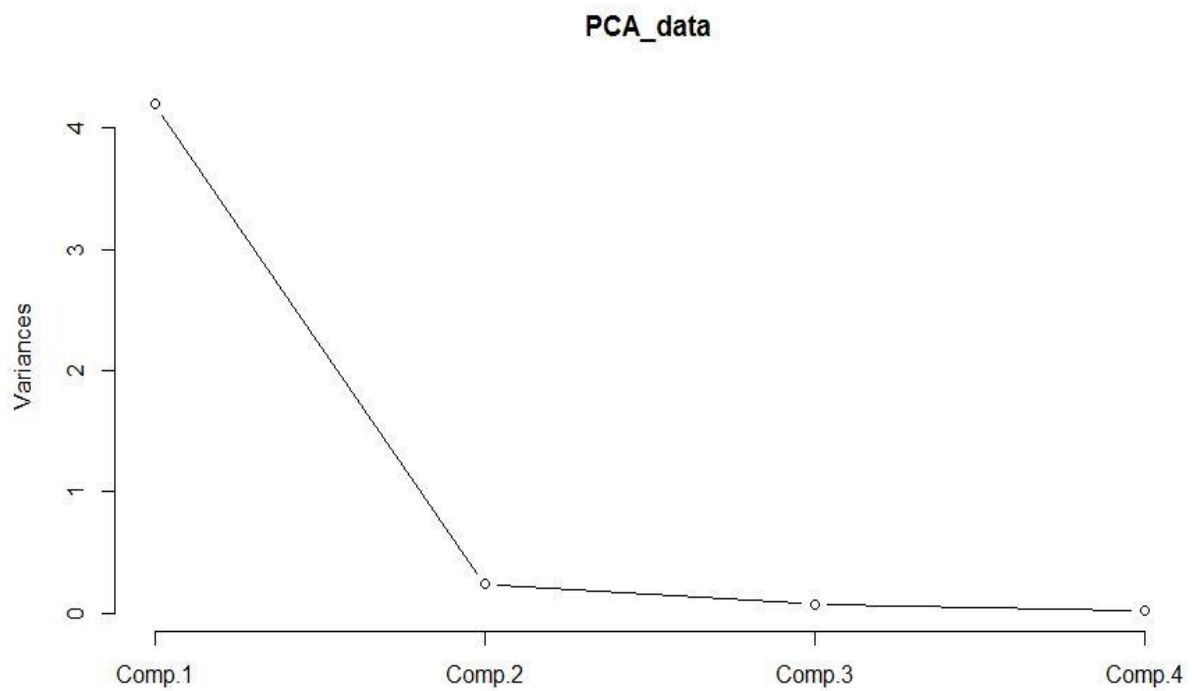
If we want the higher accuracy, we can take the first two components together and obtain a cumulative accuracy of up to **97.7%**. We can also understand how our features are transformed by using the biplot function on our model.

```
> biplot (PCA_data)
```



PCA feature transformation

```
> screplot(PCA_data, type="lines")
```



principle components

## VPM's B.N. Bhandodkar College Of Science

This plot shows the bend at the second principal component.

Let us now fit two naive Bayes models.

1. one over the entire data.
2. The second on the first principal component.

We will calculate the difference in accuracy between these two models.

```
#Select the first principal component for the second model
```

```
> model2 = PCA_data$loadings[,1]
```

```
#For the second model, we need to calculate scores by multiplying our loadings with the data
```

```
> model2_scores <- as.matrix(data_iris) %*% model2
```

```
#Loading libraries for naiveBayes model
```

```
> library(class)
```

```
> install.packages("e1071")
```

```
> library(e1071)
```

```
#Fitting the first model over the entire data
```

```
> mod1<-naiveBayes(iris[,1:4], iris[,5])
```

```
#Fitting the second model using the first principal component
```

```
> mod2<-naiveBayes(model2_scores, iris[,5])
```

```
# Accuracy for the first model
```

```
>table(predict(mod1, iris[,1:4]), iris[,5])
```

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	47	3
virginica	0	3	47

```
# Accuracy for the second model
```

```
>table(predict(mod2, model2_scores), iris[,5])
```

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	46	5
virginica	0	4	45

**All Command:**

```
data_iris <- iris[1:4]
Cov_data <- cov(data_iris )
# Find out the eigenvectors and eigenvalues using the covariance matrix
Eigen_data <- eigen(Cov_data)
# Using the inbuilt function
PCA_data <- princomp(data_iris ,cor="False")
# Let's now compare the output variances
Eigen_data$values
PCA_data$sdev^2
PCA_data$loadings[,1:4]
Eigen_data$vectors
summary(PCA_data)
biplot (PCA_data)

screplot(PCA_data, type="lines")
#Select the first principal component for the second model
model2 = PCA_data$loadings[,1]
#For the second model, we need to calculate scores by multiplying our loadings with the data
model2_scores <- as.matrix(data_iris) %*% model2

#Loading libraries for naiveBayes model
library(class)
install.packages("e1071")
library(e1071)

#Fitting the first model over the entire data
mod1<-naiveBayes(iris[,1:4], iris[,5])
#Fitting the second model using the first principal component
mod2<-naiveBayes(model2_scores, iris[,5])
# Accuracy for the first model
table(predict(mod1, iris[,1:4]), iris[,5])
# Accuracy for the second model
table(predict(mod2, model2_scores), iris[,5])
```

# VPM's B.N. Bandodkar College Of Science

## **Practical No.4**

**AIM:** Practical of Clustering.

**Theory:** This dataset is very commonly used for Overview of data, Data Visualization and Clustering model. It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The given columns in this dataset are:

i> Id

ii> SepalLength (Cm)

iii>SepalWidth (Cm)

iv> PetalLength (Cm)

v> PetalWidth (Cm)

vi> Species

**Lets visualize this dataSet and Cluster with kmeans**

**Solution approach –**

**IRIS Data, Basic Visualization before Clustering**

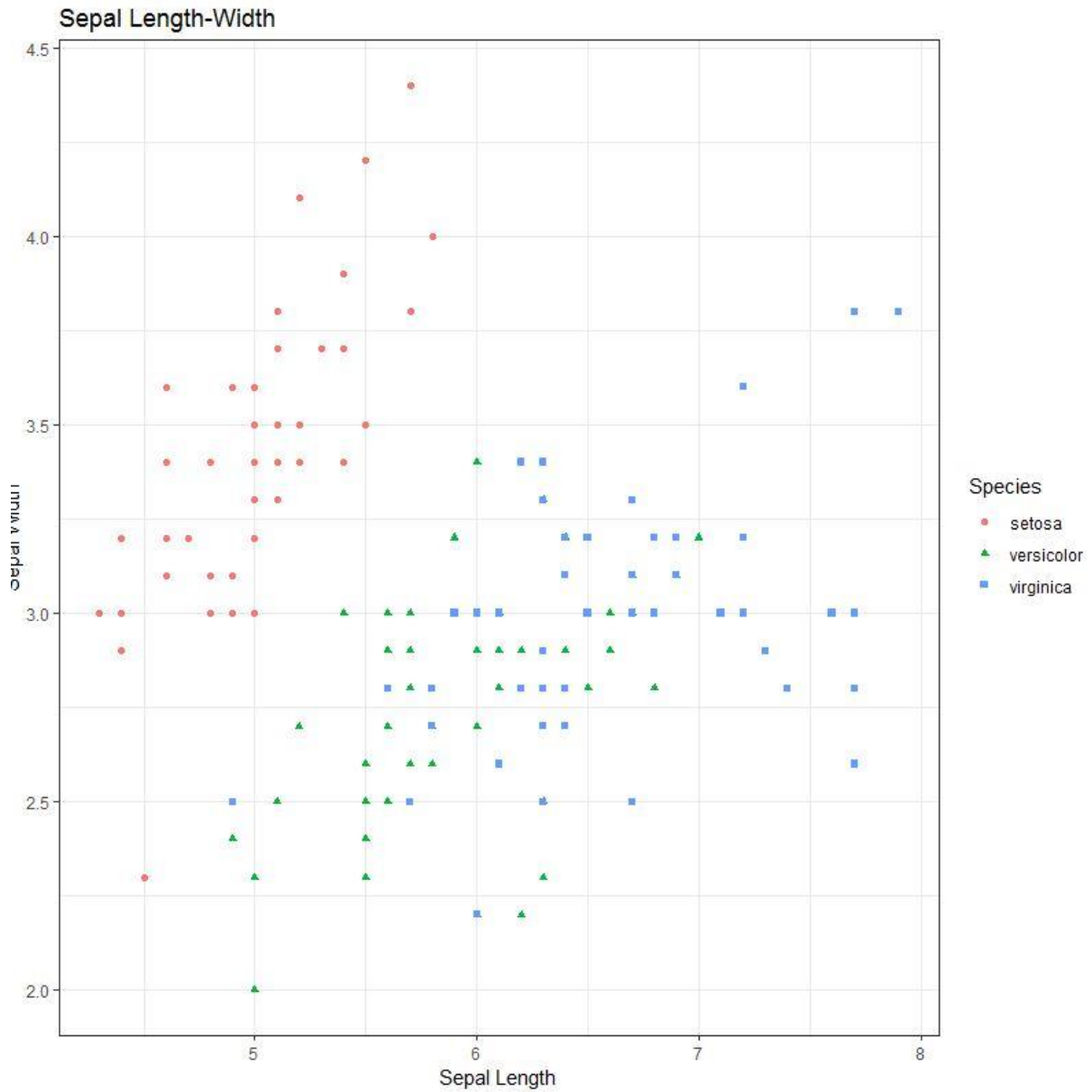
```
> install.packages("ggplot2")
```

```
> library(ggplot2)
```

```
> scatter <- ggplot(data=iris, aes(x = Sepal.Length, y = Sepal.Width))
```

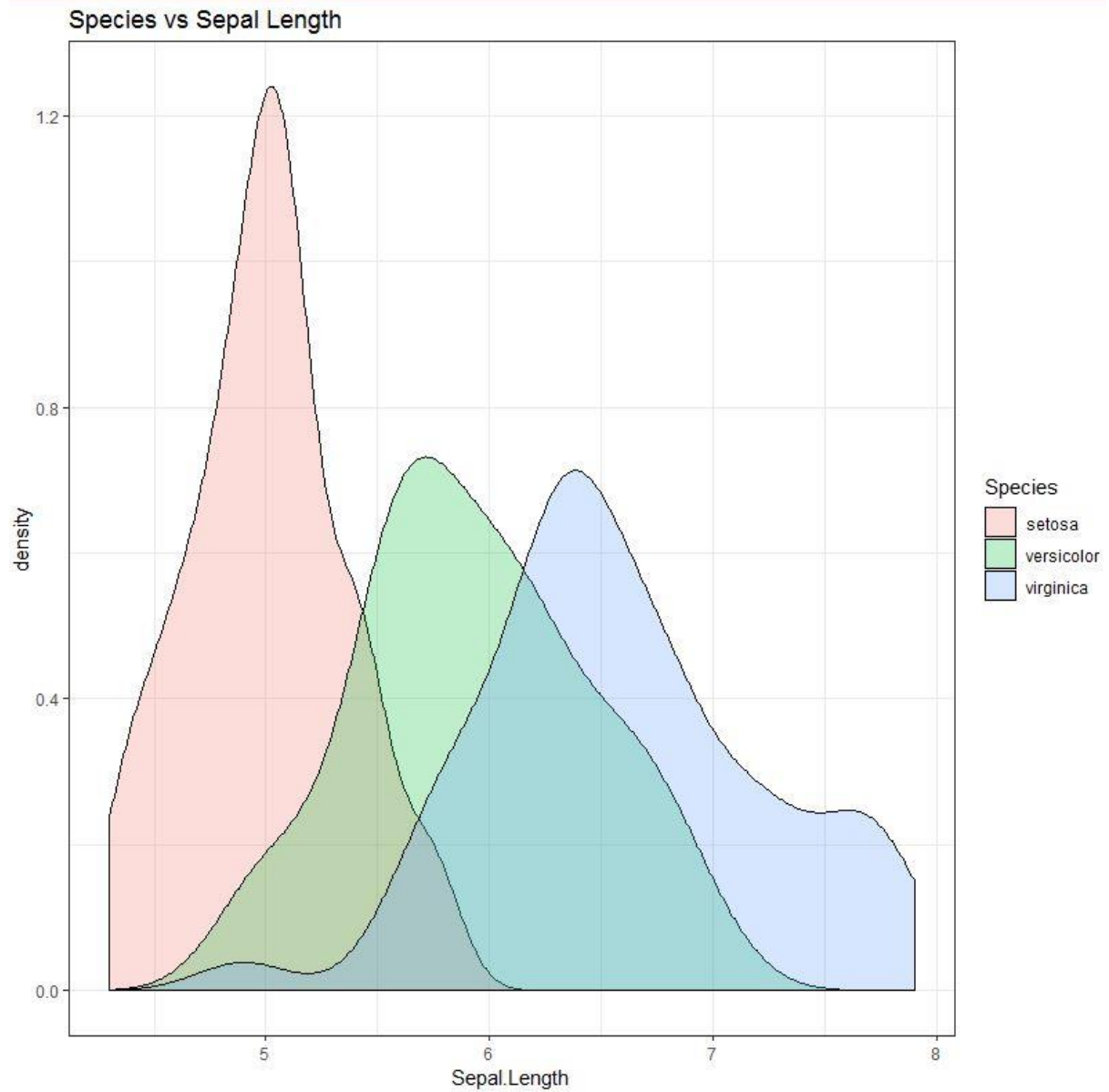
## VPM's B.N. Bandodkar College Of Science

```
> scatter + geom_point(aes(color=Species, shape=Species)) +  
+ theme_bw()+  
+ xlab("Sepal Length") + ylab("Sepal Width") +  
+ ggtitle("Sepal Length-Width")
```



## VPM's B.N. Bandodkar College Of Science

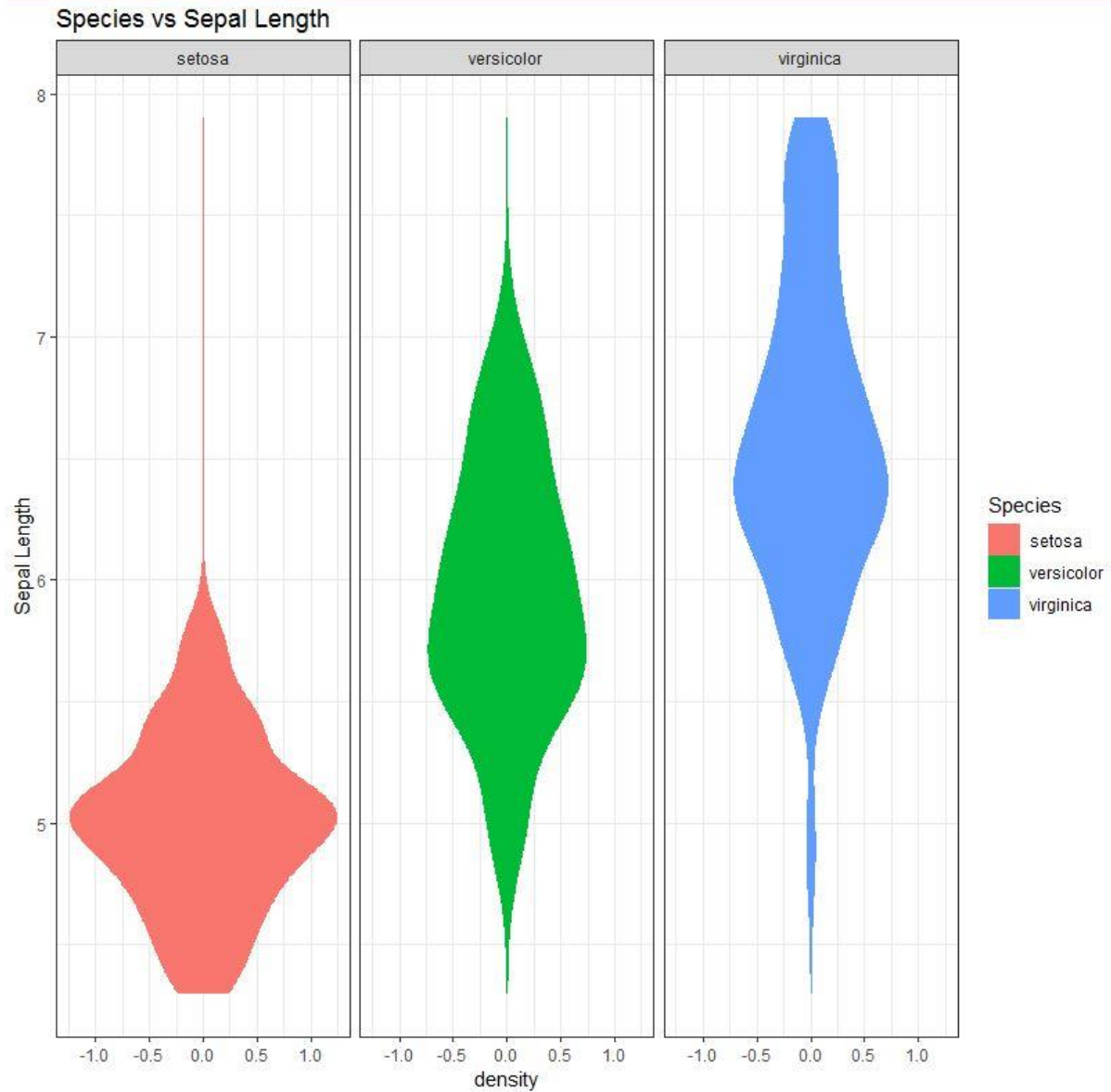
```
> ggplot(data=iris, aes(Sepal.Length, fill = Species))+  
+ theme_bw()+  
+ geom_density(alpha=0.25)+  
+ labs(x = "Sepal.Length", title="Species vs Sepal Length")
```





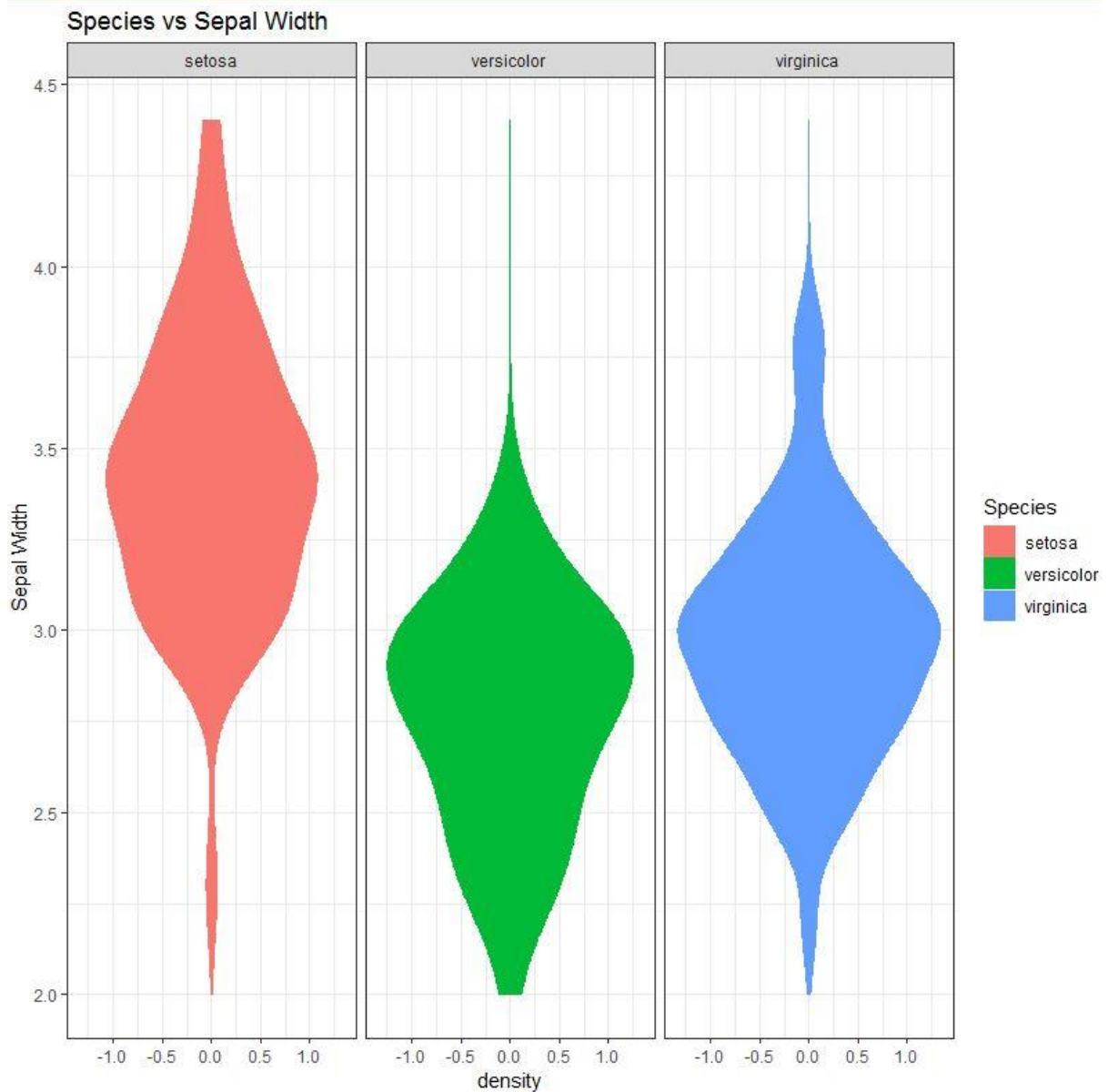
## VPM's B.N. Bandodkar College Of Science

```
> vol <- ggplot(data=iris, aes(x = Sepal.Length))  
  
> vol + stat_density(aes(ymax = ..density.., ymin = -..density..,  
+                         fill = Species, color = Species),  
+                   geom = "ribbon", position = "identity") +  
+   facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Length", title="Spec  
ies vs Sepal Length")
```



## VPM's B.N. Bandodkar College Of Science

```
> vol <- ggplot(data=iris, aes(x = Sepal.Width))  
> vol + stat_density(aes(ymax = ..density.., ymin = -..density..,  
+ fill = Species, color = Species),  
+ geom = "ribbon", position = "identity") +  
+ facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Width", title="Species  
vs Sepal Width")
```



### Clustering Data :: Method-1

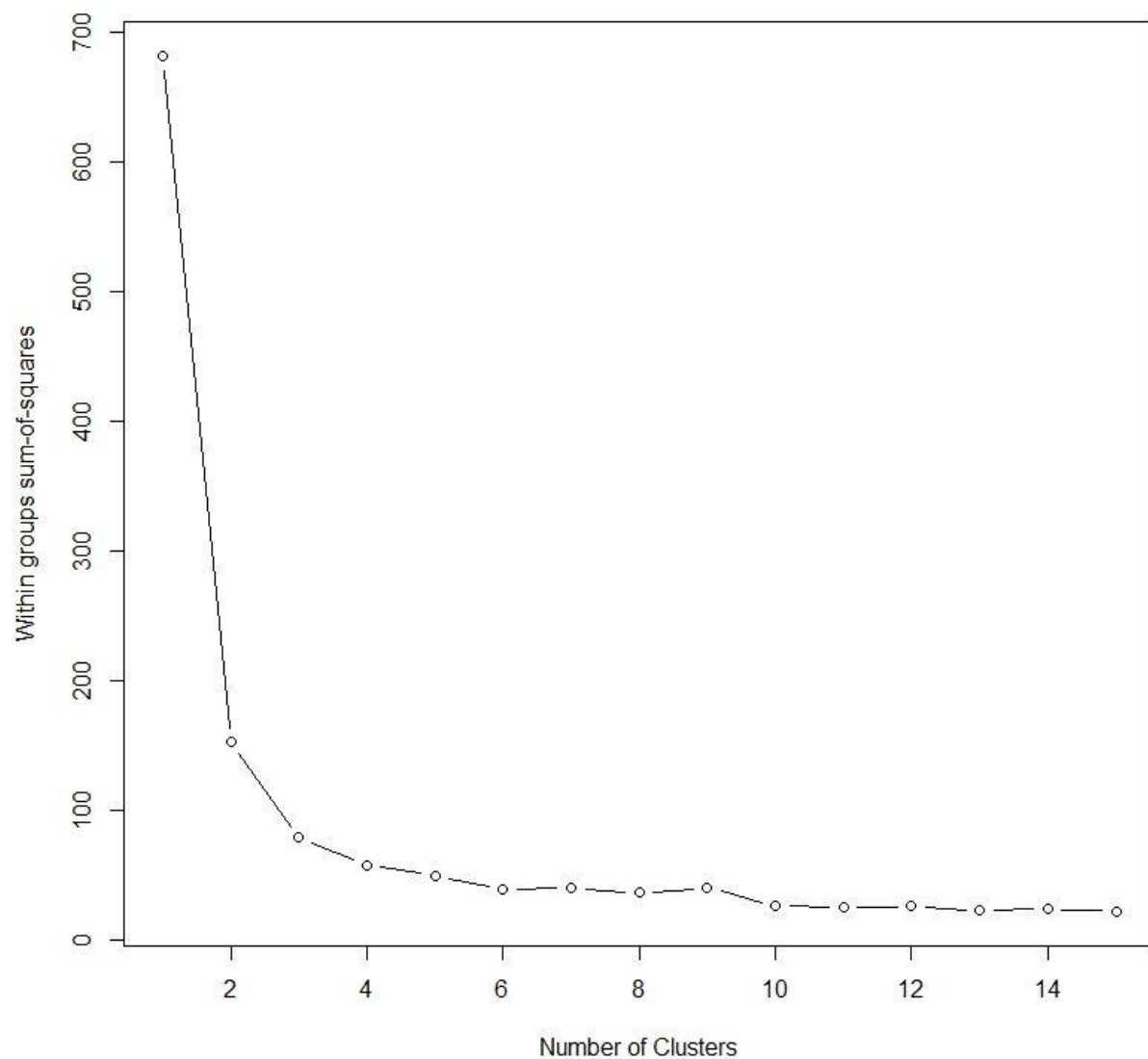
```
> irisData <- iris[,1:4]
> totalwSS<-c()
```

```
# kmeans clustering for 15 times in a loop
```

```
> for (i in 1:15)
+ {
+   clusterIRIS <- kmeans(irisData, centers=i)
+   totalwSS[i]<-clusterIRIS$tot.withinss
+ }
```

```
# Scree plot - Use plot function to plot values of tot_wss against no-of-clusters
```

```
> plot(x=1:15,                # x= No of clusters, 1 to 15
+      y=totalwSS,            # tot_wss for each
+      type="b",              # Draw both points as also connect them
+      xlab="Number of Clusters",
+      ylab="Within groups sum-of-squares")
```



## Clustering Data :: Method-2

### Using NbClust - Uses huge no of cluster suitability measuring criteria

```
> install.packages("NbClust")  
> library(NbClust)
```

```
# Set margins as: c(bottom, left, top, right)  
> par(mar = c(2,2,2,2))
```

*# NbClust measures appropriateness of cluster on a number of indices. # By default, it checks from 2 clusters to 15 clusters*

```
> nb <- NbClust(irisData, method = "kmeans") # Takes time
```

\*\*\* : The Hubert index is a graphical method of determining the number of clusters.  
In the plot of Hubert index, we seek a significant knee that corresponds to a significant increase of the value of the measure i.e the significant peak in Hubert index second differences plot.

\*\*\* : The D index is a graphical method of determining the number of clusters.  
In the plot of D index, we seek a significant knee (the significant peak in Dindex second differences plot) that corresponds to a significant increase of the value of the measure.

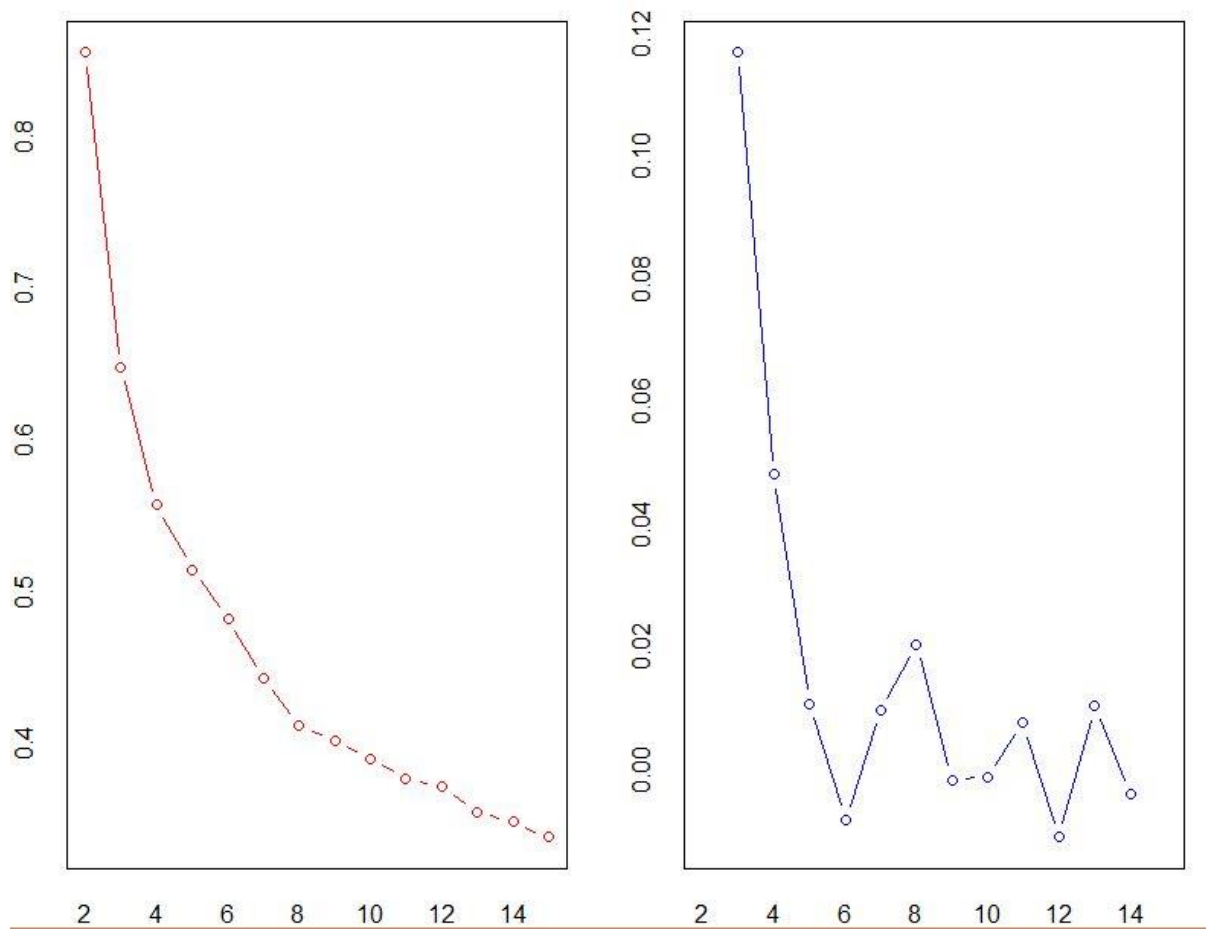
\*\*\*\*\*

- \* Among all indices:
- \* 10 proposed 2 as the best number of clusters
- \* 8 proposed 3 as the best number of clusters
- \* 2 proposed 4 as the best number of clusters
- \* 1 proposed 5 as the best number of clusters
- \* 1 proposed 8 as the best number of clusters
- \* 1 proposed 14 as the best number of clusters
- \* 1 proposed 15 as the best number of clusters

\*\*\*\*\* Conclusion \*\*\*\*\*

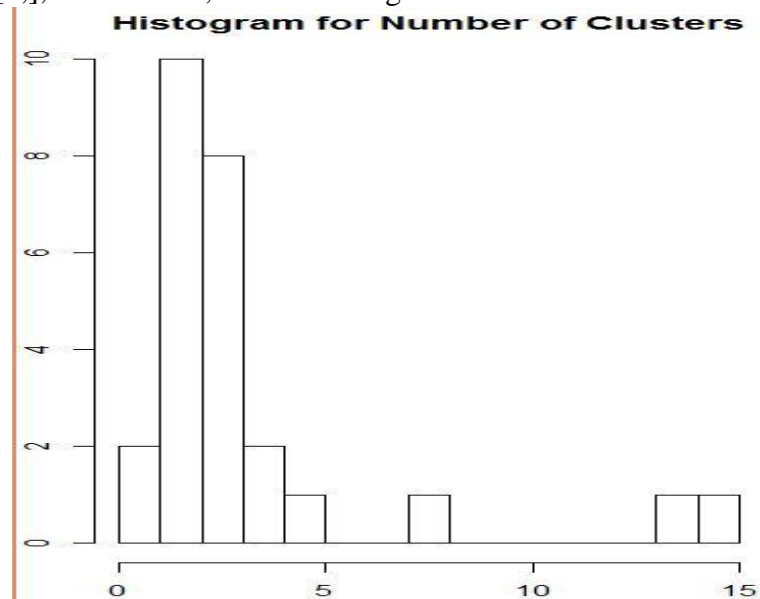
- \* According to the majority rule, the best number of clusters is 2

\*\*\*\*\*



# Draw a histogram denoting how various indices have voted for number of clusters.  
 # Out of 26 indicies, most (10) voted for 2 clusters, eight voted  
 # for 3 clusters and remaining eight (26-10-8) for other no of clusters  
 # Histogram, breaks =15 as our algorithm checks from 2 to 15 clusters

```
> hist(nb$Best.nc[1,], breaks = 15, main="Histogram for Number of Clusters")
```



### Clustering Data :: Method-3

calinski criterion is similar to finding ratio of between-cluster-variance/within-cluster variance

```
> install.packages("vegan")
```

```
> library(vegan)
```

Loading required package: permute

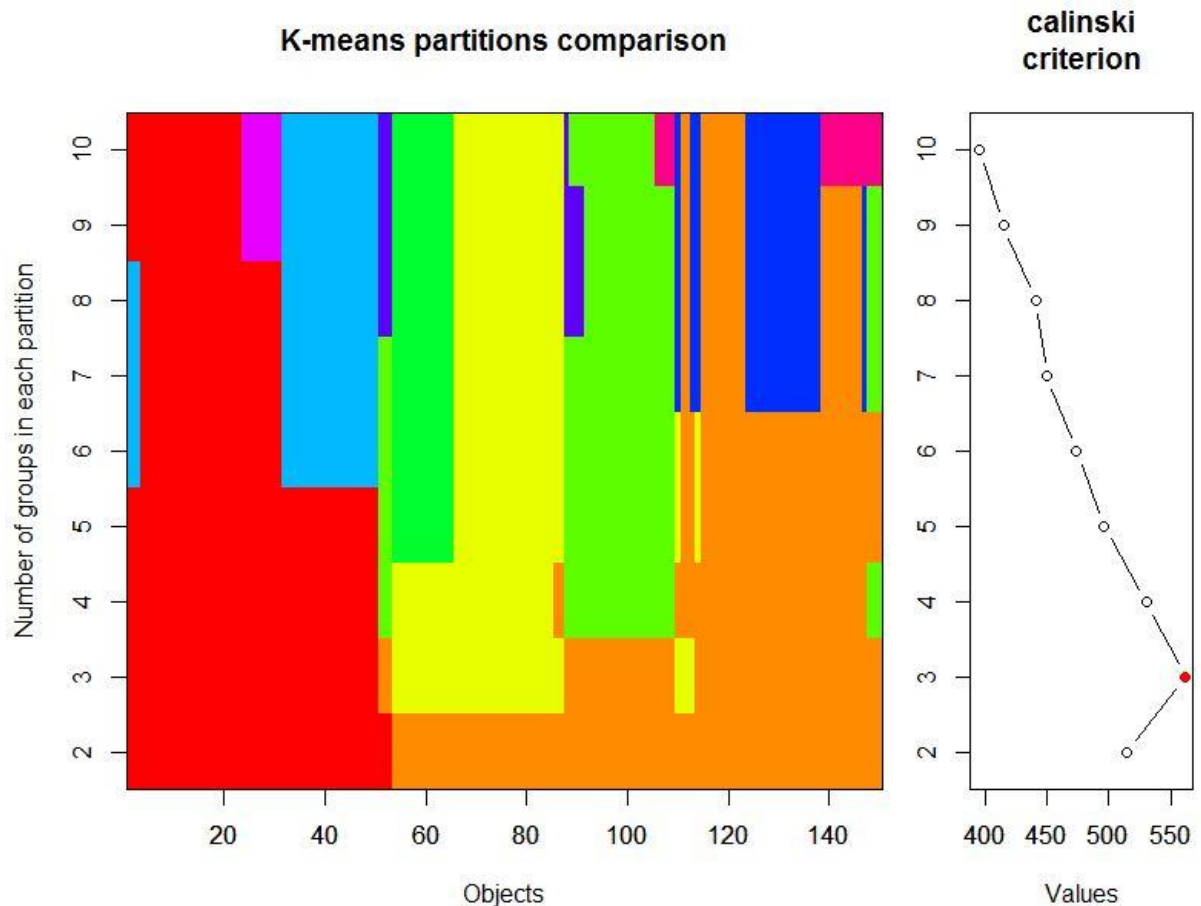
Loading required package: lattice

This is vegan 2.5-4

```
# Test for clusters 1 to 10
```

```
> modelData <- cascadeKM(irisData, 1, 10, iter = 100)
```

```
> plot(modelData, sortg = TRUE)
```



```
# Groups against BC/WC values
```

```
> modelData$results[2,]
```

1 groups 2 groups 3 groups 4 groups 5 groups 6 groups 7 groups 8 groups 9 groups 10 groups

NA 513.9245 561.6278 530.7658 495.5415 473.8506 449.6410 440.6205 414.5753 394.7207

```
> which.max(modelData$results[2,])
3 groups
3
```

#### Clustering Data with Silhouette plot :: Method-4

Try with 2 Clusters first —

**# For silhouette()**

```
> library(cluster)
```

```
> cl <- kmeans(iris[,-5], 2)
```

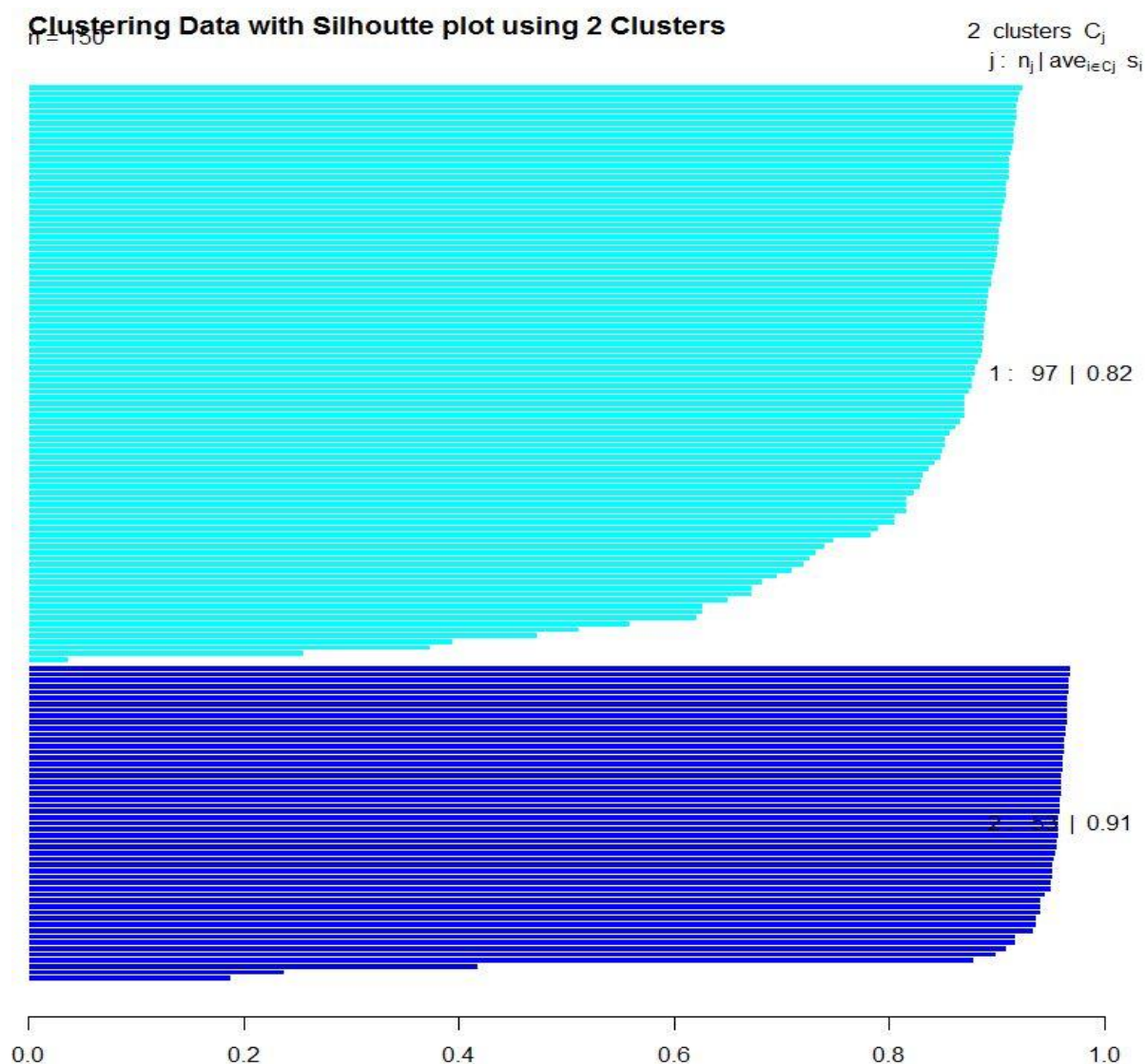
*# Compute and returns the distance matrix computed by using euclidean distance measure to compute # the distances between the rows of a data matrix.*

```
> dis <- dist(iris[,-5])^2
```

*# Get silhouette coefficient*

```
> sil = silhouette (cl$cluster, dis)
```

```
> plot(sil, main = "Clustering Data with Silhouette plot using 2 Clusters", col = c("cyan", "blue"))
```



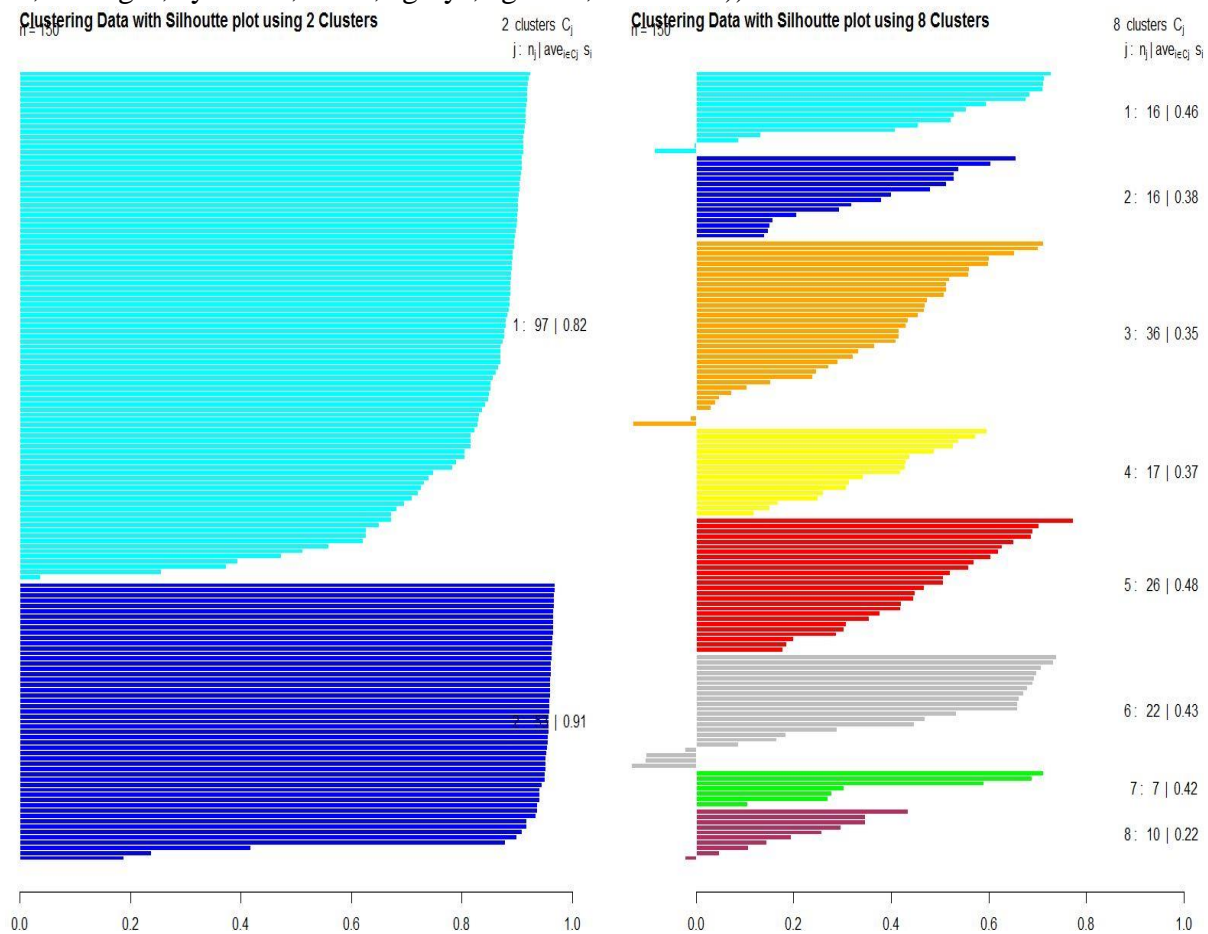
# VPM's B.N. Bandodkar College Of Science

```
> library(cluster) # For silhoutte()
> cl <- kmeans(iris[,-5], 8)
# Compute and returns the distance matrix computed by using euclidean distance measure to
compute

# the distances between the rows of a data matrix.
> dis <- dist(iris[,-5])^2

# Get silhoutte coefficient
> sil = silhouette (cl$cluster, dis)

> plot(sil, main = "Clustering Data with Silhoutte plot using 8 Clusters", col = c("cyan", "blue", "orange", "yellow", "red", "gray", "green", "maroon"))
```





# VPM's B.N. Bhandodkar College Of Science

## Analyze Clustering Tendency

Calculate Hopkin's statistic for iris and random dataset

```
# get_clust_tendency() assesses hopkins stat
```

```
> install.packages("factoextra")
```

```
> library(factoextra)
```

Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at <https://goo.gl/13EFCZ>

```
# 1. Given a vector of numbers or a column of a dataframe
```

```
# Generate uniform random numbers as per its min and max values
```

```
> install.packages("clustertend")
```

```
# Another package for hopkins() function
```

```
> library(clustertend)
```

```
# 1. Given a vector of numbers or a column of a dataframe
```

```
# Generate uniform random numbers as per its min and max values
```

```
> genx<-function(x){  
+   runif(length(x), min(x), (max(x)))  
+ }
```

```
# 2. Generate random data by applying function over each column
```

```
> random_df <- apply(iris[,-5], 2, genx)
```

```
> random_df <- as.data.frame(random_df)
```

```
# 3. Standardize both data sets
```

```
> iris[,-5] <- scale(iris[,-5]) # By default, center = T, scale = T
```

```
> random_df <- scale(random_df)
```

```
# 4. Compute Hopkins statistic for iris dataset
```

```
> res <- get_clust_tendency(iris[,-5],  
+                           n = nrow(iris) - 1 ,  
+                           graph = FALSE)
```

```
> res$hopkins_stat
```

```
[1] 0.1815219
```

```
# 5. Also calculate using function, hopkins(),
```

```
# of clustertend package
```

```
> hopkins(iris[,-5], n = nrow(iris) - 1)
```

```
$H
```

```
[1] 0.1903924
```

## VPM's B.N. Bhandodkar College Of Science

```
# 6. Compute Hopkins statistic for a random dataset
> res <- get_clust_tendency(random_df, n = nrow(random_df)-1,
+                           graph = FALSE)
> res$hopkins_stat
[1] 0.4980571
```

### All Command:

```
install.packages("ggplot2")
library(ggplot2)
scatter <- ggplot(data=iris, aes(x = Sepal.Length, y = Sepal.Width))
scatter + geom_point(aes(color=Species, shape=Species)) +
  theme_bw()+
  xlab("Sepal Length") + ylab("Sepal Width") +
  ggtitle("Sepal Length-Width")

ggplot(data=iris, aes(Sepal.Length, fill = Species))+
  theme_bw()+
  geom_density(alpha=0.25)+
  labs(x = "Sepal.Length", title="Species vs Sepal Length")

vol <- ggplot(data=iris, aes(x = Sepal.Length))

vol + stat_density(aes(ymax = ..density.., ymin = -..density..,
                      fill = Species, color = Species),
                  geom = "ribbon", position = "identity") +
  facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Length",
title="Species vs Sepal Length")

vol <- ggplot(data=iris, aes(x = Sepal.Width))

vol + stat_density(aes(ymax = ..density.., ymin = -..density..,
                      fill = Species, color = Species),
                  geom = "ribbon", position = "identity") +
  facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Width",
title="Species vs Sepal Width")

irisData <- iris[,1:4]
totalwSS<-c()

for (i in 1:15)
{
  clusterIRIS <- kmeans(irisData, centers=i)
  totalwSS[i]<-clusterIRIS$tot.withinss
}
```

## VPM's B.N. Bhandodkar College Of Science

```
plot(x=1:15,          # x= No of clusters, 1 to 15
     y=totalwSS,      # tot_wss for each
     type="b",        # Draw both points as also connect them
     xlab="Number of Clusters",
     ylab="Within groups sum-of-squares")
```

```
install.packages("NbClust")
library(NbClust)
par(mar = c(2,2,2,2))
nb <- NbClust(irisData, method = "kmeans")
hist(nb$Best.nc[1,], breaks = 15, main="Histogram for Number of Clusters")
```

```
install.packages("vegan")
library(vegan)
```

```
modelData <- cascadeKM(irisData, 1, 10, iter = 100) # Test for clusters 1 to 10
plot(modelData, sortg = TRUE)
```

```
modelData$results[2,]
```

```
which.max(modelData$results[2,])
```

```
library(cluster)
cl <- kmeans(iris[,-5], 2)
dis <- dist(iris[,-5])^2
sil = silhouette (cl$cluster, dis)
plot(sil, main = "Clustering Data with Silhouette plot using 2 Clusters", col = c("cyan",
"blue"))
library(cluster)
cl <- kmeans(iris[,-5], 8)
dis <- dist(iris[,-5])^2
sil = silhouette (cl$cluster, dis)
```

```
plot(sil, main = "Clustering Data with Silhouette plot using 8 Clusters", col = c("cyan",
"blue", "orange", "yellow", "red", "gray", "green", "maroon"))
install.packages("factoextra")
library(factoextra)
install.packages("clustertend")
library(clustertend)
genx<-function(x){
  runif(length(x), min(x), (max(x)))
}
random_df <- apply(iris[,-5], 2, genx)
```

## VPM's B.N. Bandodkar College Of Science

```
random_df <- as.data.frame(random_df)
```

```
iris[,-5] <- scale(iris[,-5])
```

```
random_df <- scale(random_df)
```

```
res <- get_clust_tendency(iris[,-5],  
                          n = nrow(iris) - 1 ,  
                          graph = FALSE)
```

```
res$hopkins_stat
```

```
hopkins(iris[,-5], n = nrow(iris) - 1)
```

```
res <- get_clust_tendency(random_df, n = nrow(random_df)-1,  
                          graph = FALSE)
```

```
res$hopkins_stat
```

# VPM's B.N. Bhandodkar College Of Science

## Practical No.5

**AIM:** Practical of Time-series forecasting.

### **Theory:**

Making predictions about the future is called extrapolation in the classical statistical handling of time series data.

More modern fields focus on the topic and refer to it as time series forecasting.

Forecasting involves taking models fit on historical data and using them to predict future observations.

Descriptive models can borrow for the future (i.e. to smooth or remove noise), they only seek to best describe the data.

An important distinction in forecasting is that the future is completely unavailable and must only be estimated from what has already happened.

The skill of a time series forecasting model is determined by its performance at predicting the future. This is often at the expense of being able to explain why a specific prediction was made, confidence intervals and even better understanding the underlying causes behind the problem.

Exploration of Time Series Data in R:

Here we'll learn to handle time series data on R. Our scope will be restricted to data exploring in a time series type of data set and not go to building time series models.

I have used an inbuilt data set of R called AirPassengers. The dataset consists of monthly totals of international airline passengers, 1949 to 1960.

Loading the Data Set

Following is the code which will help you load the data set and spill out a few top level metrics.

```
> data(AirPassengers)
```

```
> class(AirPassengers)
```

```
[1] "ts"
```

```
#This tells you that the data series is in a time series format
```

```
> start(AirPassengers)
```

```
[1] 1949  1
```

```
#This is the start of the time series
```

```
> end(AirPassengers)
```

```
[1] 1960 12
```

```
#This is the end of the time series
```

```
> frequency(AirPassengers)
```

```
[1] 12
```

```
#The cycle of this time series is 12months in a year
```

```
> summary(AirPassengers)
```

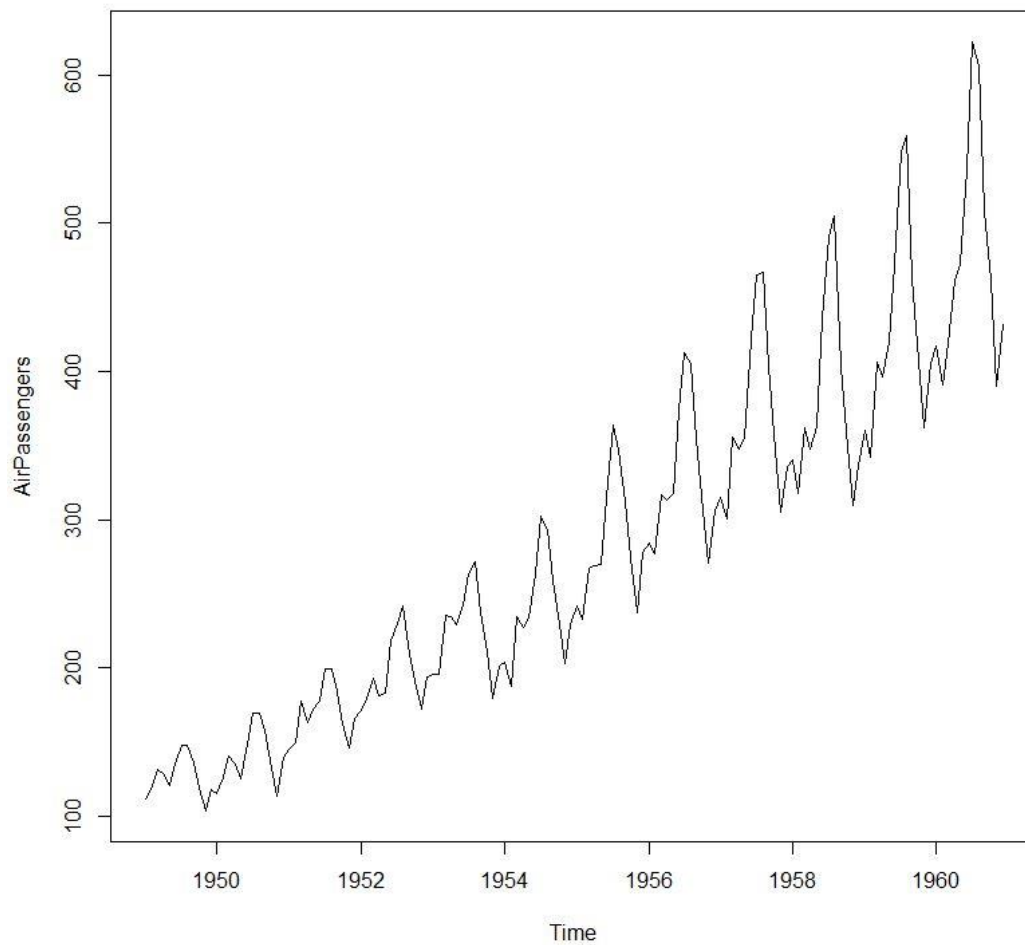
```
Min. 1st Qu. Median Mean 3rd Qu. Max.
```

```
104.0 180.0 265.5 280.3 360.5 622.0
```

## VPM's B.N. Bandodkar College Of Science

#The number of passengers are distributed across the spectrum

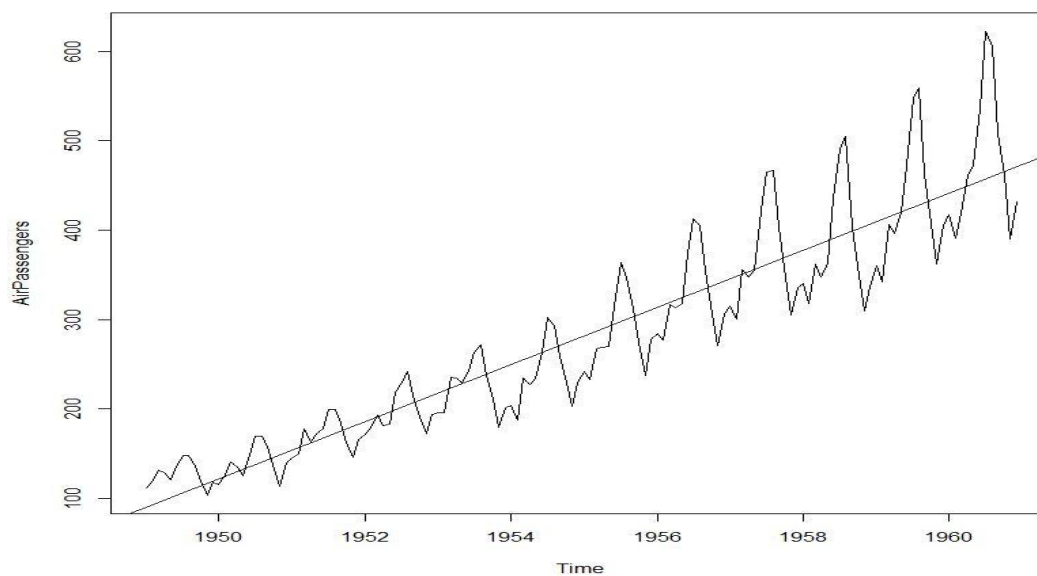
```
> plot(AirPassengers)
```



#This will plot the time series

```
> abline(reg=lm(AirPassengers~time(AirPassengers)))
```

# This will fit in a line



```
> cycle(AirPassengers)
```

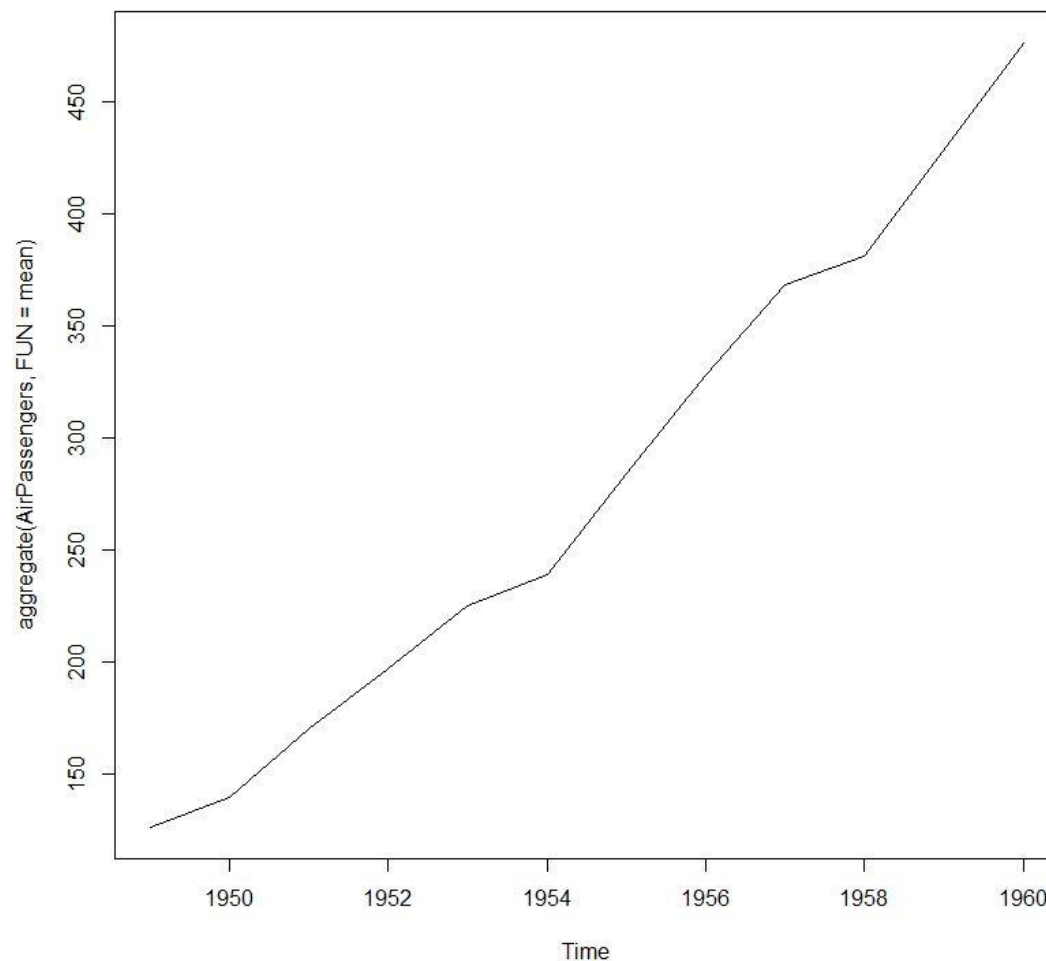
## VPM's B.N. Bandodkar College Of Science

```
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 1 2 3 4 5 6 7 8 9 10 11 12
1950 1 2 3 4 5 6 7 8 9 10 11 12
1951 1 2 3 4 5 6 7 8 9 10 11 12
1952 1 2 3 4 5 6 7 8 9 10 11 12
1953 1 2 3 4 5 6 7 8 9 10 11 12
1954 1 2 3 4 5 6 7 8 9 10 11 12
1955 1 2 3 4 5 6 7 8 9 10 11 12
1956 1 2 3 4 5 6 7 8 9 10 11 12
1957 1 2 3 4 5 6 7 8 9 10 11 12
1958 1 2 3 4 5 6 7 8 9 10 11 12
1959 1 2 3 4 5 6 7 8 9 10 11 12
1960 1 2 3 4 5 6 7 8 9 10 11 12
```

#This will print the cycle across years.

```
> plot(aggregate(AirPassengers,FUN=mean))
```

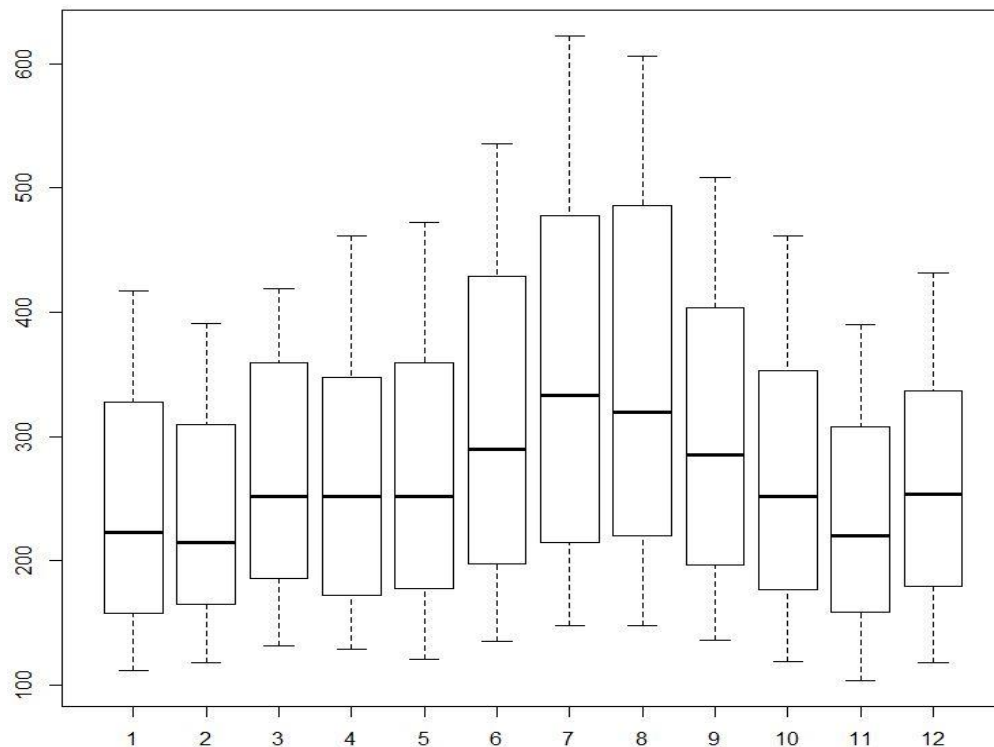
#This will aggregate the cycles and display a year on year trend



## VPM's B.N. Bandodkar College Of Science

```
> boxplot(AirPassengers~cycle(AirPassengers))
```

#Box plot across months will give us a sense on seasonal effect



### Important Inferences

1. The year on year trend clearly shows that the #passengers have been increasing without fail.
2. The variance and the mean value in July and August is much higher than rest of the months.
3. Even though the mean value of each month is quite different their variance is small. Hence, we have strong seasonal effect with a cycle of 12 months or less.

Exploring data becomes most important in a time series model – without this exploration, you will not know whether a series is stationary or not. As in this case we already know many details about the kind of model we are looking out for.

Let's now take up a few time series models and their characteristics. We will also take this problem forward and make a few predictions.

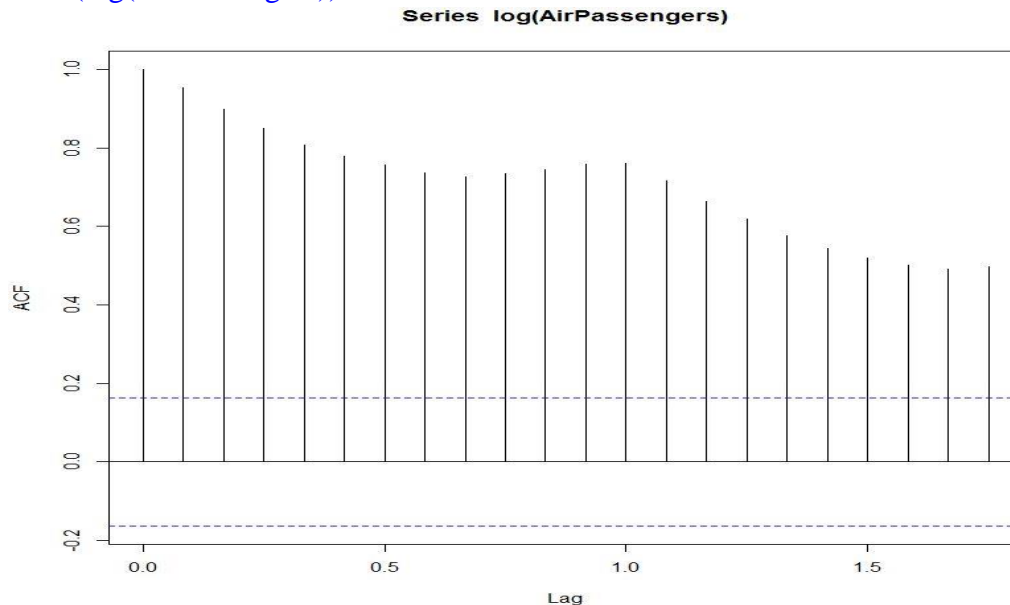


## VPM's B.N. Bhandarkar College Of Science

**Auto – correlation Function(ACF):** ACF is a plot of total correlation between different lag functions.

Following are the ACF plots for the series :

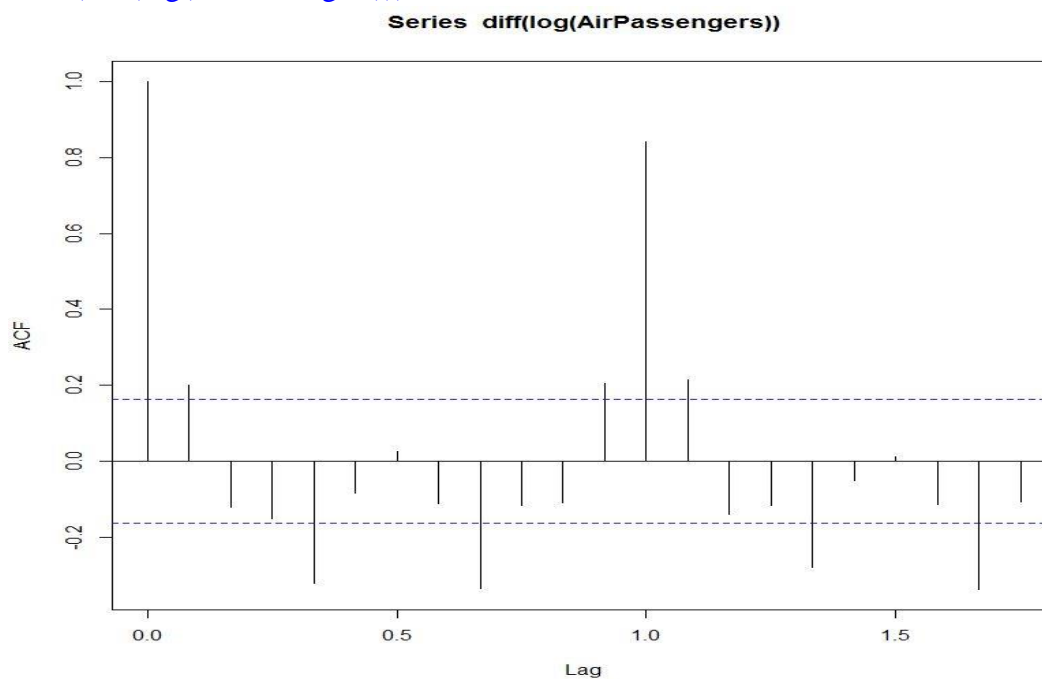
```
> acf(log(AirPassengers))
```



What do you see in the chart shown above?

Clearly, the decay of ACF chart is very slow, which means that the population is not stationary. We have already discussed above that we now intend to regress on the difference of logs rather than log directly. Let's see how ACF curve come out after regressing on the difference.

```
> acf(diff(log(AirPassengers)))
```



## VPM's B.N. Bandodkar College Of Science

```
> (fit <- arima(log(AirPassengers), c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12)))
```

Call:

```
arima(x = log(AirPassengers), order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12))
```

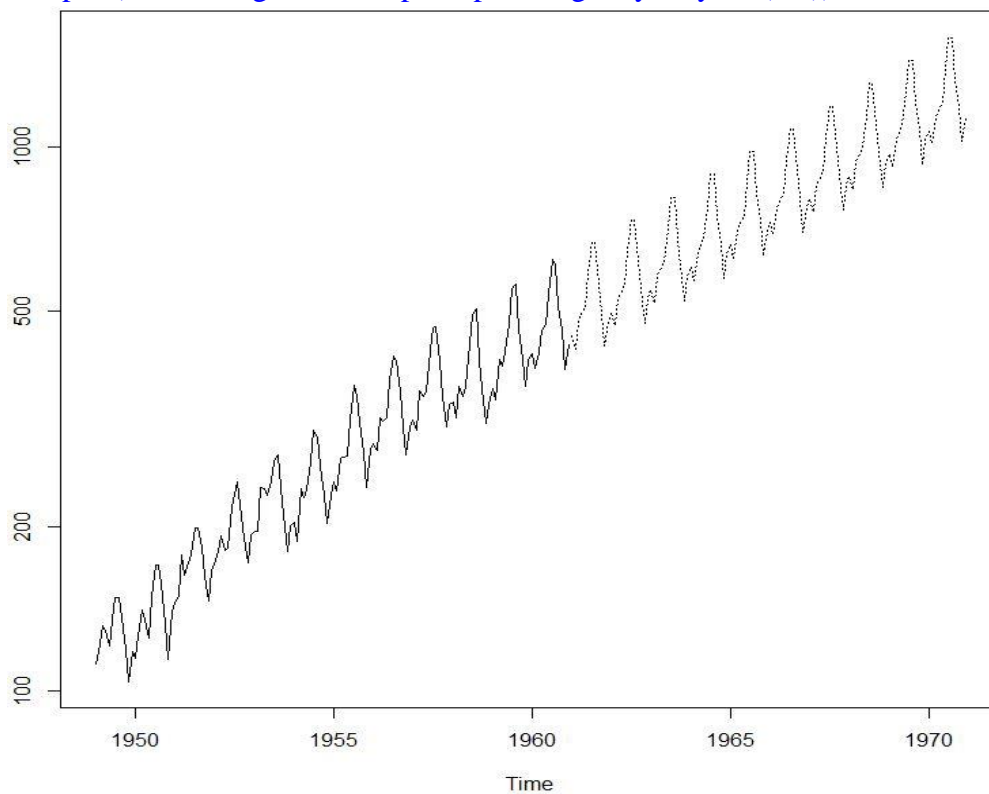
Coefficients:

```
      mal      sma1  
-0.4018 -0.5569  
s.e. 0.0896 0.0731
```

sigma^2 estimated as 0.001348: log likelihood = 244.7, aic = -483.4

```
> pred <- predict(fit, n.ahead = 10*12)
```

```
> ts.plot(AirPassengers, 2.718^pred$pred, log = "y", lty = c(1,3))
```



## VPM's B.N. Bandodkar College Of Science

### All Command:

```
data(AirPassengers)
class(AirPassengers)
start(AirPassengers)
end(AirPassengers)

frequency(AirPassengers)
summary(AirPassengers)
plot(AirPassengers)

abline(reg=lm(AirPassengers~time(AirPassengers)))

cycle(AirPassengers)

plot(aggregate(AirPassengers,FUN=mean))

boxplot(AirPassengers~cycle(AirPassengers))

acf(log(AirPassengers))

acf(diff(log(AirPassengers)))

(fit <- arima(log(AirPassengers), c(0, 1, 1),seasonal = list(order = c(0, 1, 1), period = 12)))

pred <- predict(fit, n.ahead = 10*12)

ts.plot(AirPassengers,2.718^pred$pred, log = "y", lty = c(1,3))
```

**Practical No. 6**

**AIM:** Practical of Simple/Multiple Linear Regression

**Theory:**

In statistics, **linear regression** is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called **multiple linear regression**. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quintile is used.

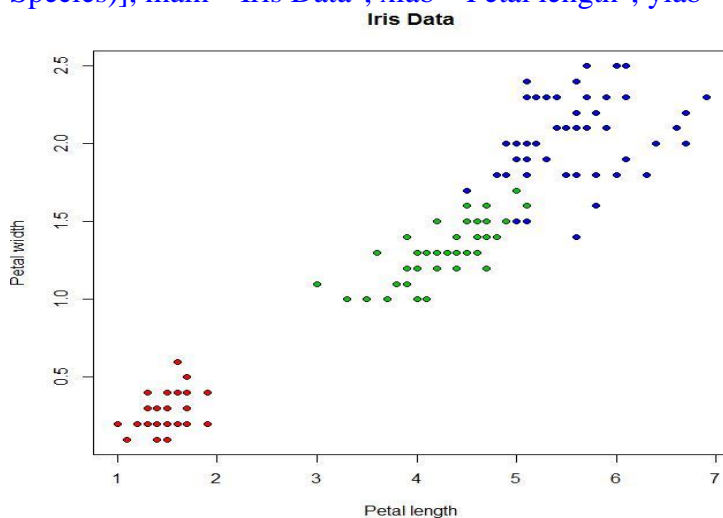
Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis. To summarise, the iris dataset consists of four measurements (length and width of the petals and sepals) of one hundred and fifty Iris flowers from three species:

You will have noticed on the iris dataset, that petal length and petal width are highly correlated over all species. How about running a linear regression? First of all, using the "least squares fit" function `lsfit` gives this:

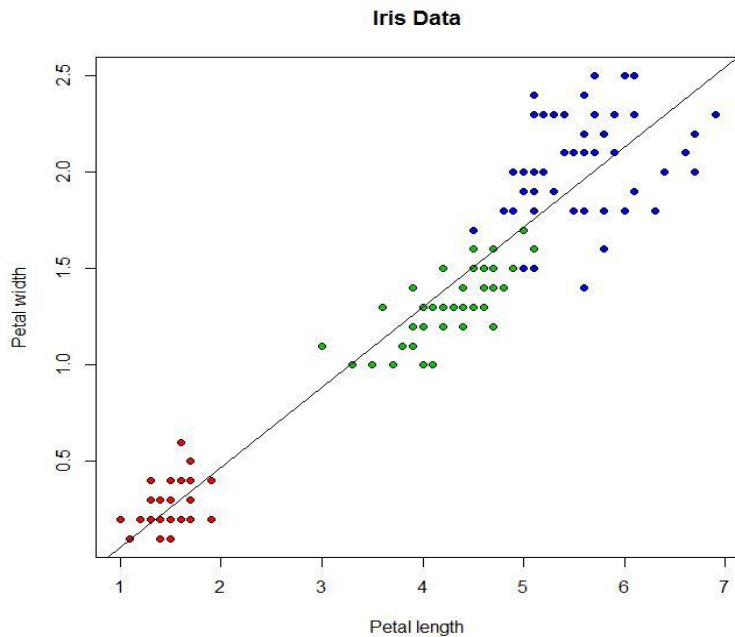
```
> lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients
```

```
Intercept      X  
-0.3630755  0.4157554
```

```
> plot(iris$Petal.Length, iris$Petal.Width, pch=21, bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length", ylab="Petal width")
```



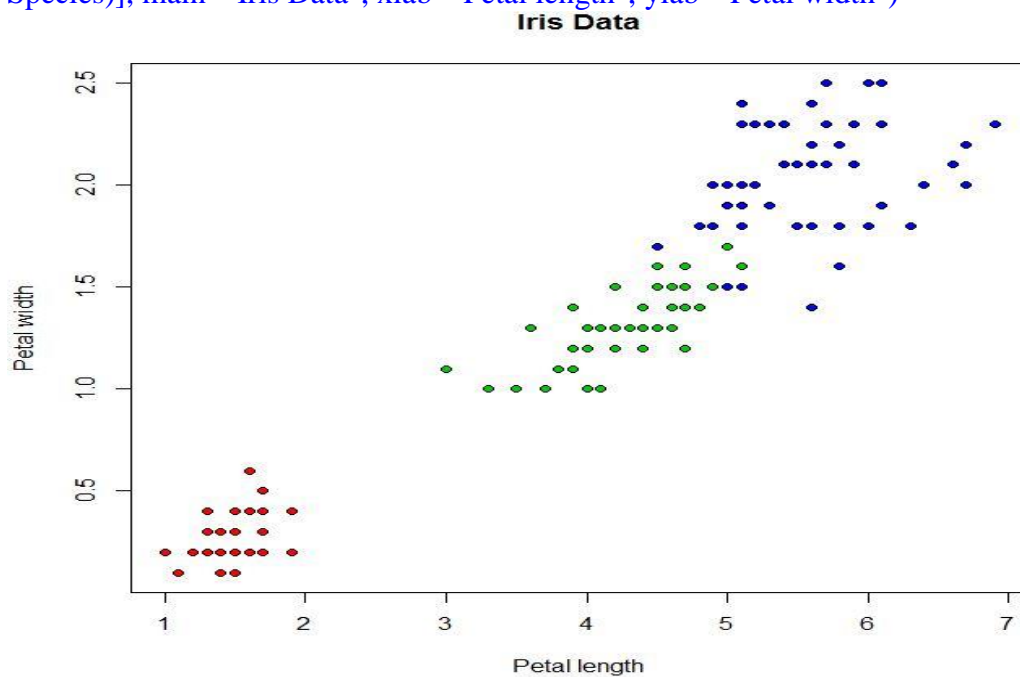
```
> abline(lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients, col="black")
```



The function `lsfit` is a bit of a "one trick pony" and its a lot more flexible to use a linear model instead (function `lm`). For this example you get exactly the same thing when we model petal width depending on petal length (written as `Petal.Width ~ Petal.Length` in R's model syntax):

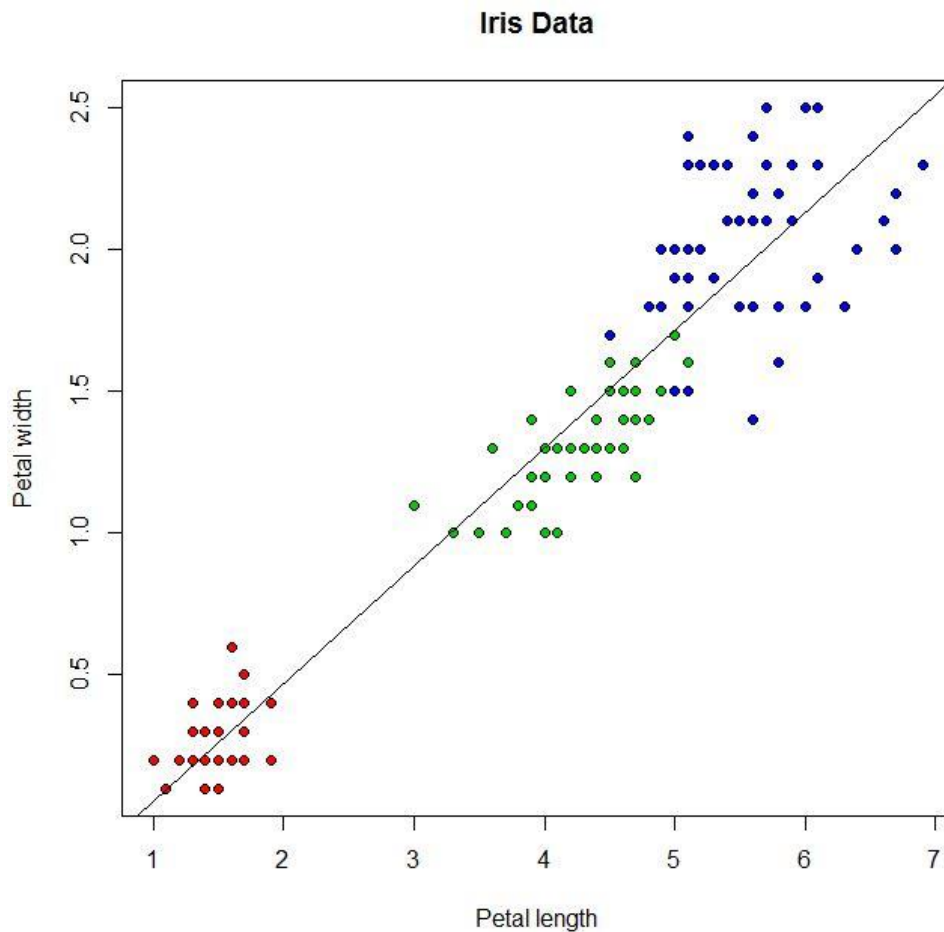
```
> lm(Petal.Width ~ Petal.Length, data=iris)$coefficients
(Intercept) Petal.Length
-0.3630755  0.4157554
```

```
> plot(iris$Petal.Length, iris$Petal.Width, pch=21, bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length", ylab="Petal width")
```



## VPM's B.N. Bhandodkar College Of Science

```
> abline(lm(Petal.Width ~ Petal.Length, data=iris)$coefficients, col="black")
```



*(same graph again)*

You get more than just that with a linear model:

```
> summary(lm(Petal.Width ~ Petal.Length, data=iris))
```

Call:

```
lm(formula = Petal.Width ~ Petal.Length, data = iris)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.56515	-0.12358	-0.01898	0.13288	0.64272

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-0.363076	0.039762	-9.131	4.7e-16 ***
Petal.Length	0.415755	0.009582	43.387	< 2e-16 ***

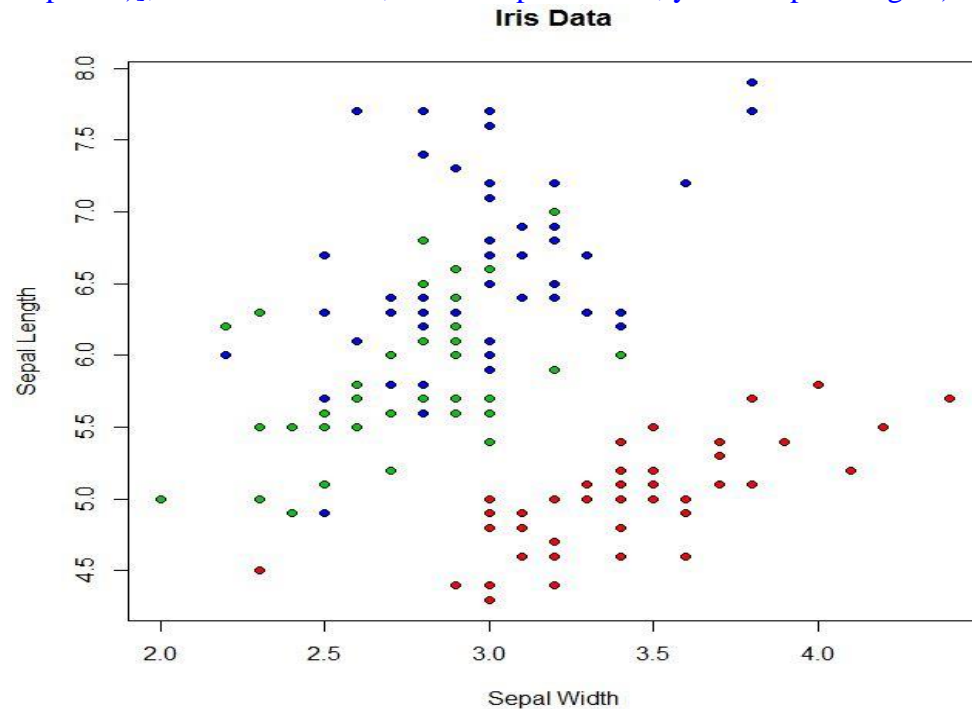
---Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2065 on 148 degrees of freedom  
Multiple R-squared: 0.9271, Adjusted R-squared: 0.9266  
F-statistic: 1882 on 1 and 148 DF, p-value: < 2.2e-16

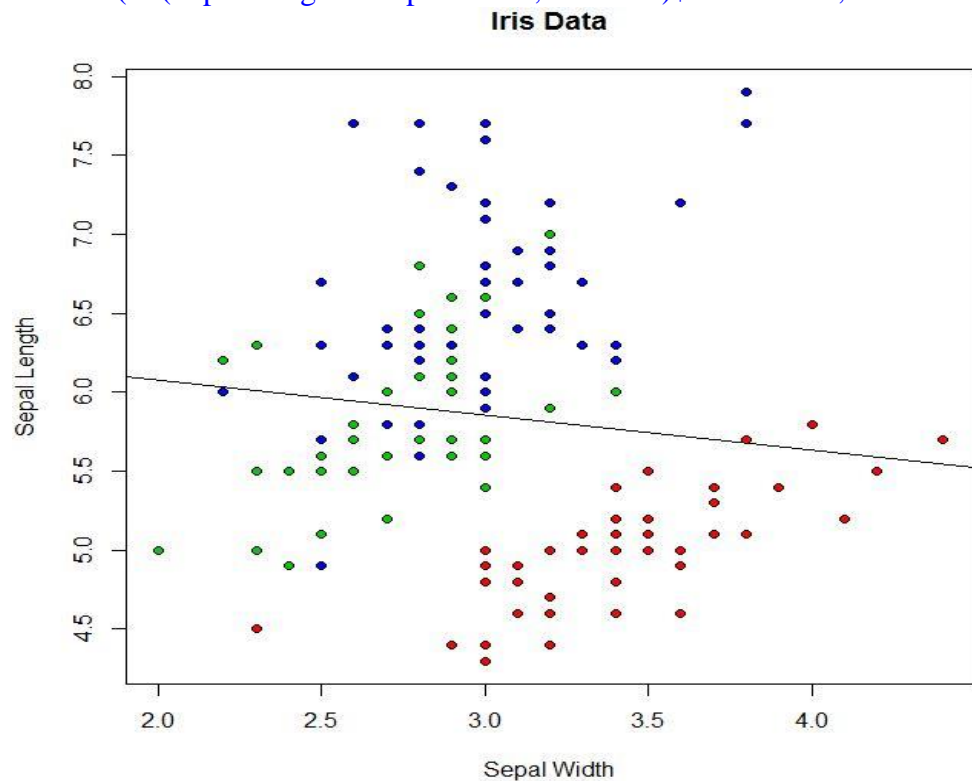
## VPM's B.N. Bandodkar College Of Science

The main point about using a linear model is we can consider more complicated examples.  
What about the sepal length as a function of the sepal width?

```
> plot(iris$Sepal.Width, iris$Sepal.Length, pch=21, bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Sepal Width", ylab="Sepal Length")
```



```
> abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")
```



## VPM's B.N. Bandodkar College Of Science

It very clear that the linear model Sepal.Length ~ Sepal.Width (black line) is not doing a very good job, even without looking at the statistics:

```
> summary(lm(Sepal.Length ~ Sepal.Width, data=iris))
```

Call:

```
lm(formula = Sepal.Length ~ Sepal.Width, data = iris)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.5561	-0.6333	-0.1120	0.5579	2.2226

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	6.5262	0.4789	13.63	<2e-16 ***
Sepal.Width	-0.2234	0.1551	-1.44	0.152

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8251 on 148 degrees of freedom

Multiple R-squared: 0.01382, Adjusted R-squared: 0.007159

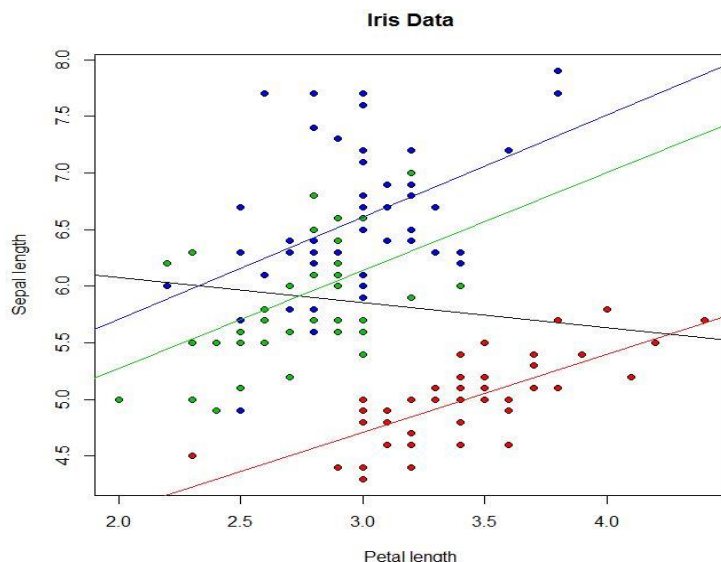
F-statistic: 2.074 on 1 and 148 DF, p-value: 0.1519

What happens if we divide the data up by species, and run three separate linear regressions?

```
> plot(iris$Sepal.Width, iris$Sepal.Length, pch=21, bg=c("red","green3","blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length", ylab="Sepal length")
> abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")
> abline(lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="setosa"),])$coefficients, col="red")
> abline(lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="versicolor"),])$coefficients, col="green3")
> abline(lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="virginica"),])$coefficients, col="blue")
```



## VPM's B.N. Bandodkar College Of Science



The coefficients doing separate per species regressions of Sepal.Length ~ Sepal.Width are:

```
> lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="setosa"),])$coefficients
```

(Intercept) Sepal.Width

2.6390012 0.6904897

```
> lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="versicolor"),])$coefficients
```

(Intercept) Sepal.Width

3.5397347 0.8650777

```
> lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="virginica"),])$coefficients
```

(Intercept) Sepal.Width

3.9068365 0.9015345

The equivalent linear model would be something like Sepal.Length ~ Petal.Length:Species + Species - 1, which gives identical coefficients (see later for why I did this):

```
> lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients
```

Speciessetosa

Speciesversicolor

Speciesvirginica

2.6390012

3.5397347

3.9068365

Sepal.Width:Speciessetosa

Sepal.Width:Speciesversicolor

Sepal.Width:Speciesvirginica

0.6904897

0.8650777

0.9015345

What are these new terms? Because Species is a categorical input variable (a factor in R's terminology) it can't be used directly in a linear model as they need actual numbers (a linear model is basically a matrix equation). So, the following "dummy variables" have been invented for each data point (which *are* just numbers)

Speciessetosa = 1 if Species is "setosa", 0 otherwise

Speciesversicolor = 1 if Species is "versicolor", 0 otherwise

Speciesvirginica = 1 if Species is "virginica", 0 otherwise

Sepal.Width:Speciessetosa = Sepal.Width if Species is "setosa", 0 otherwise

Sepal.Width:Speciesversicolor = Sepal.Width if Species is "versicolor", 0 otherwise

Sepal.Width:Speciesvirginica = Sepal.Width if Species is "virginica", 0 otherwise

## VPM's B.N. Bandodkar College Of Science

Using the summary command on the linear model object gives:

```
> summary(lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris))
```

Call:

```
lm(formula = Sepal.Length ~ Sepal.Width:Species + Species - 1,
    data = iris)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.26067	-0.25861	-0.03305	0.18929	1.44917

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
Speciessetosa	2.6390	0.5715	4.618	8.53e-06 ***
Speciesversicolor	3.5397	0.5580	6.343	2.74e-09 ***
Speciesvirginica	3.9068	0.5827	6.705	4.25e-10 ***
Sepal.Width:Speciessetosa	0.6905	0.1657	4.166	5.31e-05 ***
Sepal.Width:Speciesversicolor	0.8651	0.2002	4.321	2.88e-05 ***
Sepal.Width:Speciesvirginica	0.9015	0.1948	4.628	8.16e-06 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4397 on 144 degrees of freedom

Multiple R-squared: 0.9947, Adjusted R-squared: 0.9944

F-statistic: 4478 on 6 and 144 DF, p-value: < 2.2e-16

Just look at those p-values! Every single term has an excellent p-value, as does the model as a whole. And the residual standard error has also been halved.

In this case, the Sepal.Length ~ Sepal.Width:Species + Species - 1 model is clearly much better than just Sepal.Length ~ Sepal.Width.

### Simplify with AIC

On the other hand, what about this choice instead: Sepal.Length ~ Sepal.Width + Species. In fact, this is what the AIC (Akaike Information Criterion) step function gives you if you start with all possible interactions between sepal width and species, which is written Sepal.Length ~ Sepal.Width \* Species (using a asterix instead of a plus or colon) in R:

```
> summary(step(lm(Sepal.Length ~ Sepal.Width * Species, data=iris)))
```

Start: AIC=-240.59

Sepal.Length ~ Sepal.Width \* Species

	Df	Sum of Sq	RSS	AIC
- Sepal.Width:Species	2	0.15719	28.004	-243.75
<none>		27.846		-240.59

Step: AIC=-243.74

Sepal.Length ~ Sepal.Width + Species

## VPM's B.N. Bhandodkar College Of Science

	Df	Sum of Sq	RSS	AIC
<none>		28.004	243.75	
- Sepal.Width	1	10.953	38.956	-196.23
- Species	2	72.752	100.756	-55.69

Call:

lm(formula = Sepal.Length ~ Sepal.Width + Species, data = iris)

Residuals:

Min	1Q	Median	3Q	Max
-1.30711	-0.25713	-0.05325	0.19542	1.41253

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	2.2514	0.3698	6.089	9.57e-09 ***
Sepal.Width	0.8036	0.1063	7.557	4.19e-12 ***
Speciesversicolor	1.4587	0.1121	13.012	< 2e-16 ***
Speciesvirginica	1.9468	0.1000	19.465	< 2e-16 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.438 on 146 degrees of freedom

Multiple R-squared: 0.7259, Adjusted R-squared: 0.7203

F-statistic: 128.9 on 3 and 146 DF, p-value: < 2.2e-16

I just introduced a model of the form  $\text{Sepal.Length} \sim \text{Sepal.Width}:\text{Species} + \text{Species} - 1$ , which gave identical coefficients to those found doing species specific regressions:

> `lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients`

Speciessetosa	Speciesversicolor	Speciesvirginica
2.6390012	3.5397347	3.9068365
Sepal.Width:Speciessetosa	Sepal.Width:Speciesversicolor	Sepal.Width:Speciesvirginica
0.6904897	0.8650777	0.9015345

The use of the "- 1" in the model above told R not to automatically include a default intercept term. The alternative is the following:

> `lm(Sepal.Length ~ Sepal.Width:Species + Species, data=iris)$coefficients`

(Intercept)	Speciesversicolor	Speciesvirginica
2.6390012	0.9007335	1.2678352
Sepal.Width:Speciessetosa	Sepal.Width:Speciesversicolor	Sepal.Width:Speciesvirginica
0.6904897	0.8650777	0.9015345

## VPM's B.N. Bhandodkar College Of Science

### All Command:

```
lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients
```

```
plot(iris$Petal.Length, iris$Petal.Width, pch=21,  
bg=c("red", "green3", "blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",  
ylab="Petal width")
```

```
abline(lsfit(iris$Petal.Length, iris$Petal.Width)$coefficients, col="black")
```

```
lm(Petal.Width ~ Petal.Length, data=iris)$coefficients
```

```
plot(iris$Petal.Length, iris$Petal.Width, pch=21,  
bg=c("red", "green3", "blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",  
ylab="Petal width")
```

```
abline(lm(Petal.Width ~ Petal.Length, data=iris)$coefficients, col="black")
```

```
summary(lm(Petal.Width ~ Petal.Length, data=iris))
```

```
plot(iris$Sepal.Width, iris$Sepal.Length, pch=21,  
bg=c("red", "green3", "blue")[unclass(iris$Species)], main="Iris Data", xlab="Sepal Width",  
ylab="Sepal Length")
```

```
abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")
```

```
summary(lm(Sepal.Length ~ Sepal.Width, data=iris))
```

```
plot(iris$Sepal.Width, iris$Sepal.Length, pch=21,  
bg=c("red", "green3", "blue")[unclass(iris$Species)], main="Iris Data", xlab="Petal length",  
ylab="Sepal length")
```

```
abline(lm(Sepal.Length ~ Sepal.Width, data=iris)$coefficients, col="black")
```

```
abline(lm(Sepal.Length ~ Sepal.Width,  
data=iris[which(iris$Species=="setosa"),])$coefficients, col="red")
```

```
abline(lm(Sepal.Length ~ Sepal.Width,  
data=iris[which(iris$Species=="versicolor"),])$coefficients, col="green3")
```

```
abline(lm(Sepal.Length ~ Sepal.Width,  
data=iris[which(iris$Species=="virginica"),])$coefficients, col="blue")
```

```
lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="setosa"),])$coefficients
```

```
lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="versicolor"),])$coefficients
```

```
lm(Sepal.Length ~ Sepal.Width, data=iris[which(iris$Species=="virginica"),])$coefficients
```

```
lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients
```

## VPM's B.N. Bandodkar College Of Science

```
summary(lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris))
```

```
summary(step(lm(Sepal.Length ~ Sepal.Width * Species, data=iris)))
```

```
lm(Sepal.Length ~ Sepal.Width:Species + Species - 1, data=iris)$coefficients
```

```
lm(Sepal.Length ~ Sepal.Width:Species + Species, data=iris)$coefficients
```

**Practical No.7**

**AIM:** Practical of Logistics Regression.

**Theory:**

**Logistic regression** is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

**Data Exploration:**

```
> library(datasets)
> ir_data<- iris
> head(ir_data)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1         5.1         3.5          1.4          0.2  setosa
2         4.9         3.0          1.4          0.2  setosa
3         4.7         3.2          1.3          0.2  setosa
4         4.6         3.1          1.5          0.2  setosa
5         5.0         3.6          1.4          0.2  setosa
6         5.4         3.9          1.7          0.4  setosa
> str(ir_data)
'data.frame':      150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

So, we have a dataframe with 150 observations of 5 variables. The first 4 variables give information about plant attributes in centimeters and the last one give us the name of plant species. Species are given as Factor variable with 3 levels:

```
> levels(ir_data$Species)
[1] "setosa" "versicolor" "virginica"
```

We should check whether we have any NA values in our dataset:

```
> sum(is.na(ir_data))
[1] 0
```

So, we are dealing with a complete dataset here. As we want to use Logistic Regression in this post, let's subset the data so that we have to deal with 2 species of plants rather than 3 (because logistic regression will be built on binary outcomes)

```
> ir_data<-ir_data[1:100,]
```

Also we will randomly define our Test and Control groups:

```
> set.seed(100)
```

# VPM's B.N. Bandodkar College Of Science

```
> samp<-sample(1:100,80)
```

```
> ir_test<-ir_data[samp,]
```

```
> ir_ctrl<-ir_data[-samp,]
```

We will use the test set to create our model and control set to check our model. Now, lets explore the dataset a little bit more with the help of plots:

```
> install.packages("ggplot2")
```

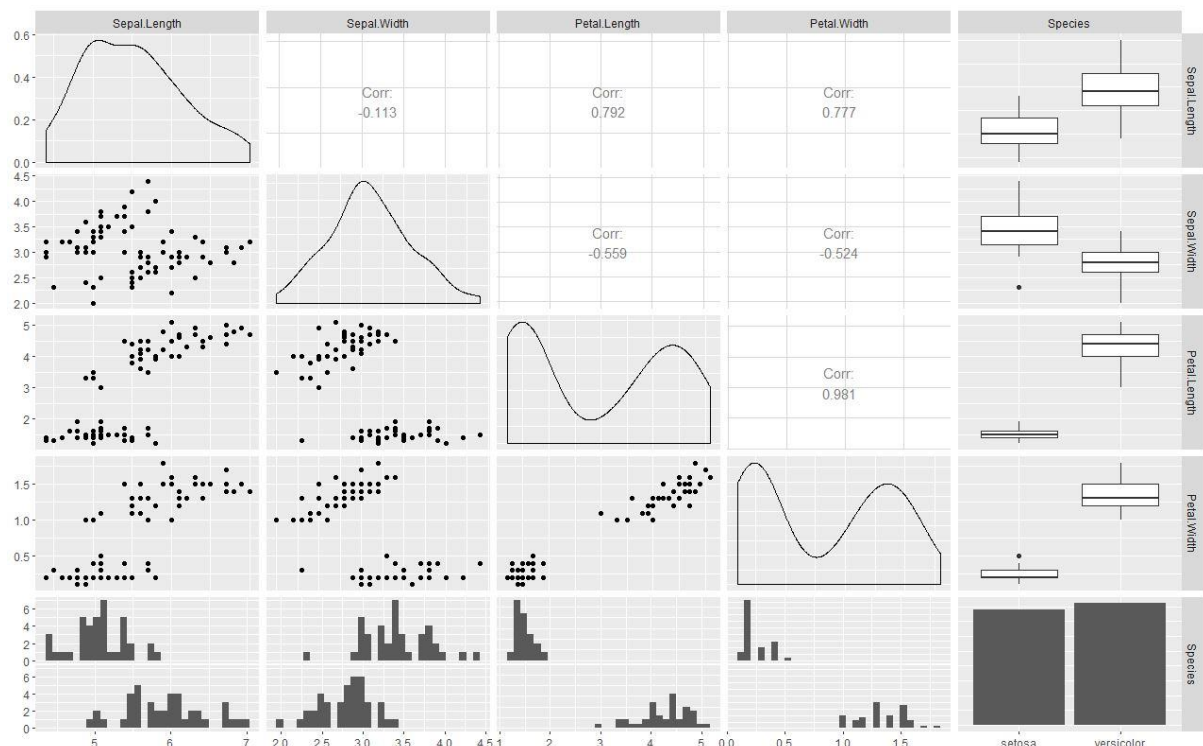
```
> library(ggplot2)
```

```
> install.packages("GGally")
```

```
> library(GGally)
```

```
> ggpairs(ir_test)
```

```
plot: [5,1] [=====
=====] 84% est
: 2s `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
plot: [5,2] [=====
=====] 88%
est: 2s `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
plot: [5,3] [=====
=====] 9
2% est: 1s `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
plot: [5,4] [=====
=====] 96%
96% est: 1s `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Model Fitting

## VPM's B.N. Bandodkar College Of Science

Now, we will try to model this data using Logistic Regression. Here, we will keep it simple and will use only a single variable:

```
> y<-ir_test$Species; x<-ir_test$Sepal.Length  
> glfit<-glm(y~x, family = 'binomial')
```

The default link function for above model is 'logit', which is what we want. We can use this simple model to get the probability of whether a given plant is 'setosa' or 'versicolor' based on its 'sepal length'. Before jumping to predictions using, let us have a look at the model itself.

```
> summary(glfit)
```

Call:

```
glm(formula = y ~ x, family = "binomial")
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.94538	-0.50121	0.04079	0.45923	2.26238

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-25.386	5.517	-4.601	4.20e-06 ***
x	4.675	1.017	4.596	4.31e-06 ***

---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 110.854 on 79 degrees of freedom  
Residual deviance: 56.716 on 78 degrees of freedom  
AIC: 60.716

Number of Fisher Scoring iterations: 6

We can see that the P-Values indicate highly significant results for this model. Although, we should check any model deeply, but right now we will move to the prediction part.

####Checking Model's Predictions Let us use our Control set which we defined earlier to predict using this model:

```
> newdata<- data.frame(x=ir_ctrl$Sepal.Length)  
> predicted_val<-predict(glfit, newdata, type="response")  
> prediction<-data.frame(ir_ctrl$Sepal.Length, ir_ctrl$Species,predicted_val)  
> prediction
```

	ir_ctrl.Sepal.Length	ir_ctrl.Species	predicted_val
1	5.1	setosa	0.176005274
2	4.7	setosa	0.031871367
3	4.6	setosa	0.020210042
4	5.0	setosa	0.118037011
5	4.6	setosa	0.020210042
6	4.3	setosa	0.005048194
7	4.6	setosa	0.020210042

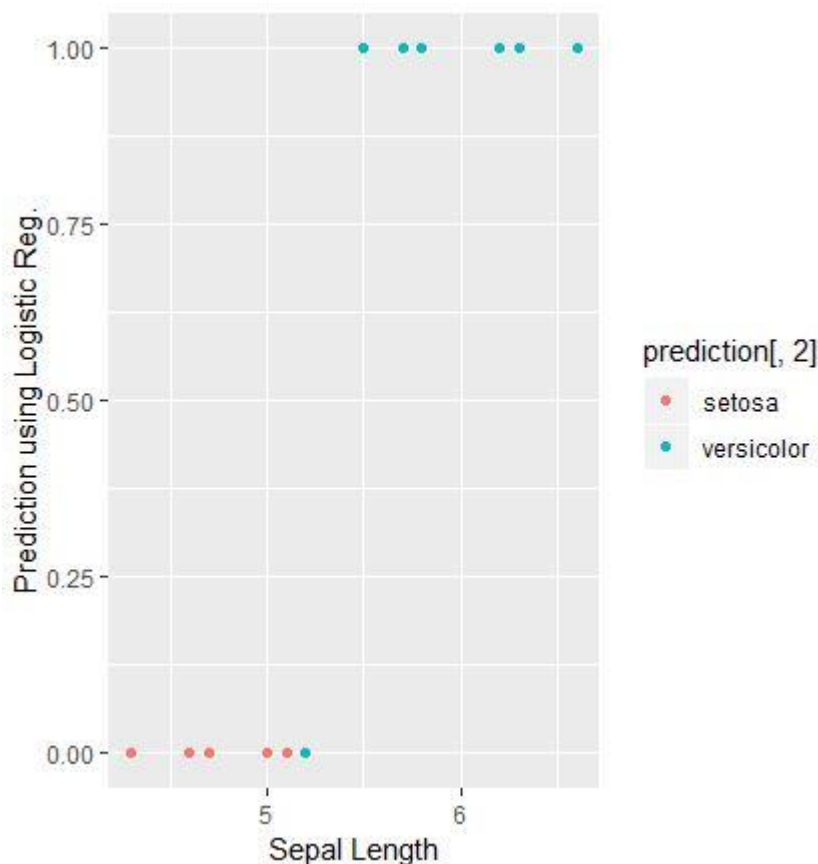


## VPM's B.N. Bandodkar College Of Science

8	5.2	setosa	0.254235573
9	5.2	setosa	0.254235573
10	5.0	setosa	0.118037011
11	5.0	setosa	0.118037011
12	6.6	versicolor	0.995801728
13	5.2	versicolor	0.254235573
14	5.8	versicolor	0.849266756
15	6.2	versicolor	0.973373695
16	6.6	versicolor	0.995801728
17	5.5	versicolor	0.580872616
18	6.3	versicolor	0.983149322
19	5.7	versicolor	0.779260130
20	5.7	versicolor	0.779260130

We can see in the table above that what probability our model is predicting for a given plant to be 'versicolor' based on its 'sepal length'. Let's visualize this result using a simple plot. Let's say that we will consider any plant to be 'versicolor' if its probability for the same is more than 0.5:

```
> qqplot(prediction[,1], round(prediction[,3]), col=prediction[,2], xlab = 'Sepal Length', ylab  
+       = 'Prediction using Logistic Reg.')
```



So, from the above plot, we can see that our simple model is doing a fairly good prediction for plant species. We can also see a blue dot in the bottom cluster. This blue dot is showing that although correct specie of this plant is 'versicolor' but our model is predicting it as 'setosa'.

## VPM's B.N. Bandodkar College Of Science

### All Command

```
library(datasets)
ir_data<- iris
head(ir_data)
str(ir_data)
levels(ir_data$Species)
sum(is.na(ir_data))
ir_data<-ir_data[1:100,]
set.seed(100)
samp<-sample(1:100,80)
ir_test<-ir_data[samp,]
ir_ctrl<-ir_data[-samp,]
install.packages("ggplot2")
library(ggplot2)
install.packages("GGally")
library(GGally)
ggpairs(ir_test)
y<-ir_test$Species; x<-ir_test$Sepal.Length
glfit<-glm(y~x, family = 'binomial')
summary(glfit)
newdata<- data.frame(x=ir_ctrl$Sepal.Length)
predicted_val<-predict(glfit, newdata, type="response")
prediction<-data.frame(ir_ctrl$Sepal.Length, ir_ctrl$Species,predicted_val)
prediction

qplot(prediction[,1], round(prediction[,3]), col=prediction[,2], xlab = 'Sepal Length', ylab
      = 'Prediction using Logistic Reg.')
```

# VPM's B.N. Bhandodkar College Of Science

## Practical No.8

**AIM:** Practical of Hypothesis testing.

### **Theory:**

Hypothesis Tests, or Statistical Hypothesis Testing, is a technique used to compare two datasets, or a sample from a dataset. It is a **statistical inference method** so, in the end of the test, you'll **draw a conclusion**—you'll infer something—about the characteristics of what you're comparing.

A hypothesis test is usually composed by

- *Null Hypothesis* ( $H_0$ , read "H zero"): states that all things remain equal. No phenomena is observed or there is not relationship between what you are comparing;
- *Alternative Hypothesis* ( $H_1$ , read "H one"): states the opposite of the Null Hypothesis. That there was some change, or observed relationship between what you are comparing.

### **Hypothesis Testing with R:**

Hypothesis tests for population means are done in R using the command "**t.test**".

#### **One-sample hypothesis test :**

Let  $x$  represents a sample collected from a normal population with unknown mean and standard deviation. We want to test if the population mean is equal to 9, at significance level 5%.

The hypotheses are:

```
> x= c(6.2, 6.6, 7.1, 7.4, 7.6, 7.9, 8, 8.3, 8.4, 8.5, 8.6,  
+ 8.8, 8.8, 9.1, 9.2, 9.4, 9.4, 9.7, 9.9, 10.2, 10.4, 10.8,  
+ 11.3, 11.9)          #Entering the data  
  
> t.test(x-9,alternative="two.sided",conf.level=0.95)  
#Performing the t-test
```

#### One Sample t-test

```
data: x - 9  
t = -0.35687, df = 23, p-value = 0.7244  
alternative hypothesis: true mean is not equal to 0  
95 percent confidence interval:  
-0.7079827 0.4996494  
sample estimates:  
mean of x  
-0.1041667
```

Interpretation of the result:

The P-value (0.3622) is greater than the significance level 5% (1-0.95), so we conclude that the null hypothesis that the mean of this population is 9 is plausible.

## VPM's B.N. Bandodkar College Of Science

### Two-sample hypothesis test:

If we are interested in finding the confidence interval for the difference of two population means, the R-command "**t.test**" is also to be used.

We are interested in testing observations middle range and higher viscosity are from populations with different means, at significance level 5%.

The hypotheses are:

```
> x=c(418,421,421,422,425,427,431,434,437,439,446,447,448,453,454,463,465)
# Entering the data into the R-workspace
> y=c(429,430,430,431,36,437,440,441,445,446,447)
> test2<-t.test(x,y,alternative="two.sided",mu=0,var.equal=F,conf.level=0.95)
> test2          #performing a t-test procedure, containing a confidence
                  interval computation
```

### Welch Two Sample t-test

data: x and y

t = 1.0123, df = 10.202, p-value = 0.3348

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-44.46343 118.86984

sample estimates:

mean of x mean of y

438.2941 401.0909

### Interpretation of the result:

The P-value (0.3348) is greater than the significance level 5% (1-0.95), so we conclude that the null hypothesis that the population means are equal is plausible.

## VPM's B.N. Bandodkar College Of Science

### All Command

```
x= c(6.2, 6.6, 7.1, 7.4, 7.6, 7.9, 8, 8.3, 8.4, 8.5, 8.6,  
      8.8, 8.8, 9.1, 9.2, 9.4, 9.4, 9.7, 9.9, 10.2, 10.4, 10.8,  
      11.3, 11.9)
```

```
t.test(x-9,alternative="two.sided",conf.level=0.95)
```

```
x=c(418,421,421,422,425,427,431,434,437,439,446,447,448,453,454,463,465)
```

```
y=c(429,430,430,431,36,437,440,441,445,446,447)
```

```
test2<-t.test(x,y,alternative="two.sided",mu=0,var.equal=F,conf.level=0.95)
```

```
test2
```

# VPM's B.N. Bandodkar College Of Science

## Practical No.9

**AIM:** Practical of Analysis of Variance.

### **Theory:**

**Analysis of variance (ANOVA)** is a collection of statistical models and their associated estimation procedures (such as the "variation" among and between groups) used to analyse the differences among group means in a sample.

ANOVA was developed by statistician and evolutionary biologist Ronald Fisher. In the ANOVA setting, the observed variance in a particular variable is partitioned into components attributable to different sources of variation. In its simplest form, ANOVA provides a statistical test of whether the population means of several groups are equal, and therefore generalizes the *t*-test to more than two groups.

ANOVA is useful for comparing (testing) three or more group means for statistical significance. It is conceptually similar to multiple two-sample *t*-tests, but is more conservative, resulting in fewer type I errors, and is therefore suited to a wide range of practical problems.

### **An Example of ANOVA using R:**

There are three groups with seven observations per group. We denote group *i* values by *y<sub>i</sub>*:

```
> y1 = c(18.2, 20.1, 17.6, 16.8, 18.8, 19.7, 19.1)
> y2 = c(17.4, 18.7, 19.1, 16.4, 15.9, 18.4, 17.7)
> y3 = c(15.2, 18.8, 17.7, 16.5, 15.9, 17.1, 16.7)
```

Now we combine them into one long vector, with a second vector, group, identifying group membership:

```
> y = c(y1, y2, y3)
> n = rep(7, 3)
> n
[1] 7 7 7
> group = rep(1:3, n)
> group
[1] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3
```

Here are summaries by group and for the combined data. First we show stem-leaf diagrams.

```
> tmp = tapply(y, group, stem)
```

The decimal point is at the |

```
16 | 8
17 | 6
18 | 28
19 | 17
20 | 1
```

The decimal point is at the |

## VPM's B.N. Bandodkar College Of Science

```
15 | 9
16 | 4
17 | 47
18 | 47
19 | 1
```

The decimal point is at the |

```
15 | 29
16 | 57
17 | 17
18 | 8
```

```
> stem(y)
```

The decimal point is at the |

```
15 | 299
16 | 4578
17 | 14677
18 | 24788
19 | 117
20 | 1
```

Now we show summary statistics by group and overall. We locally define a temporary function, tmpfn, to make this easier.

```
> tmpfn = function(x) c(sum = sum(x), mean = mean(x), var = var(x),
+                        n = length(x))
> tapply(y, group, tmpfn)
$`1`
```

sum	mean	var	n
130.300000	18.614286	1.358095	7.000000

```
$`2`
```

sum	mean	var	n
123.600000	17.657143	1.409524	7.000000

```
$`3`
```

sum	mean	var	n
117.900000	16.842857	1.392857	7.000000

```
> tmpfn(y)
```

sum	mean	var	n
371.800000	17.704762	1.798476	21.000000

While we could show you how to use R to mimic the computation of SS by hand, it is

## VPM's B.N. Bandodkar College Of Science

more natural to go directly to the ANOVA table.

```
> data = data.frame(y = y, group = factor(group))
> fit = lm(y ~ group, data)
> anova(fit)
```

Analysis of Variance Table

Response: y

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
group	2	11.007	5.5033	3.9683	0.03735 *
Residuals	18	24.963	1.3868		

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The anova(fit) object can be used for other computations on the handout and in class. For instance, the tabled F values can be found by the following. First we extract the treatment and error degrees of freedom. Then we use qt to get the tabled F values.

```
> df = anova(fit)[, "Df"]
> names(df) = c("trt", "err")
> df
trt err
 2  18
> alpha = c(0.05, 0.01)
> qf(alpha, df["trt"], df["err"], lower.tail = FALSE)
[1] 3.554557 6.012905
```

A confidence interval on the pooled variance can be computed as well using the anova(fit) object. First we get the residual sum of squares, SSTrt, then we divide by the appropriate chi-square tabled values.

```
> anova(fit)["Residuals", "Sum Sq"]
[1] 24.96286

> anova(fit)["Residuals", "Sum Sq"]/qchisq(c(0.025, 0.975), 18,
+                                           lower.tail = FALSE)
[1] 0.7918086 3.0328790
```



## VPM's B.N. Bandodkar College Of Science

### All Command

```
y1 = c(18.2, 20.1, 17.6, 16.8, 18.8, 19.7, 19.1)
y2 = c(17.4, 18.7, 19.1, 16.4, 15.9, 18.4, 17.7)
y3 = c(15.2, 18.8, 17.7, 16.5, 15.9, 17.1, 16.7)
```

```
y = c(y1, y2, y3)
n = rep(7, 3)
n
```

```
group = rep(1:3, n)
group
```

```
tmp = tapply(y, group, stem)
```

```
stem(y)
```

```
tmpfn = function(x) c(sum = sum(x), mean = mean(x), var = var(x),
                      n = length(x))
tapply(y, group, tmpfn)
```

```
tmpfn(y)
```

```
data = data.frame(y = y, group = factor(group))
fit = lm(y ~ group, data)
anova(fit)
```

```
df = anova(fit)[, "Df"]
names(df) = c("trt", "err")
df
```

```
alpha = c(0.05, 0.01)
qf(alpha, df["trt"], df["err"], lower.tail = FALSE)
```

```
anova(fit)["Residuals", "Sum Sq"]
```

```
anova(fit)["Residuals", "Sum Sq"]/qchisq(c(0.025, 0.975), 18,
lower.tail = FALSE)
```

**Practical No. 10**

**AIM:** Practical of Decision Tree.

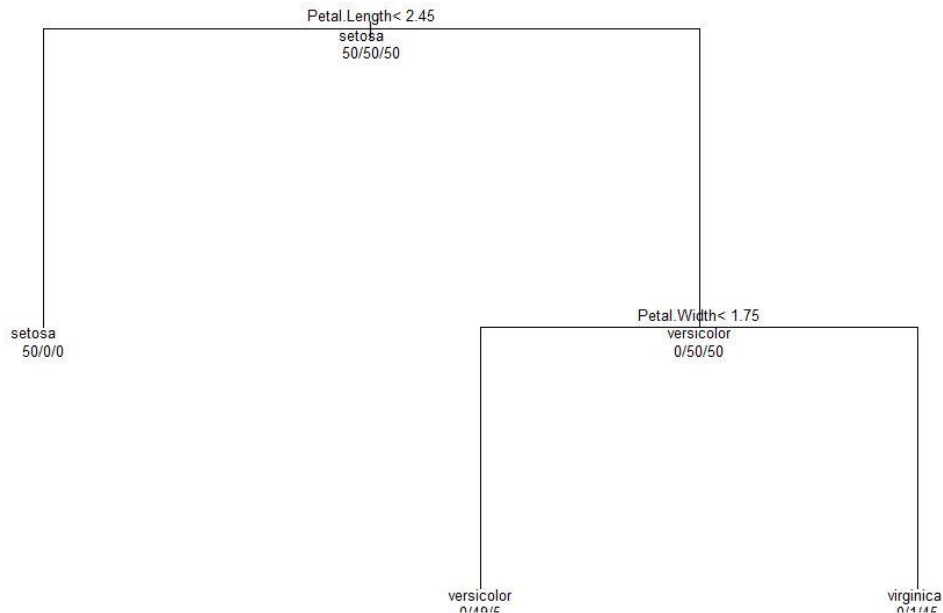
**Theory:** A **decision tree** is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

- Used for classifying data by partitioning attribute space.
- Tries to find axis-parallel decision boundaries for specified optimality criteria.
- Leaf nodes contain class labels, representing classification decisions.
- Keeps splitting nodes based on split criterion, such as GINI index, information gain or entropy.
- Pruning necessary to avoid over fitting.

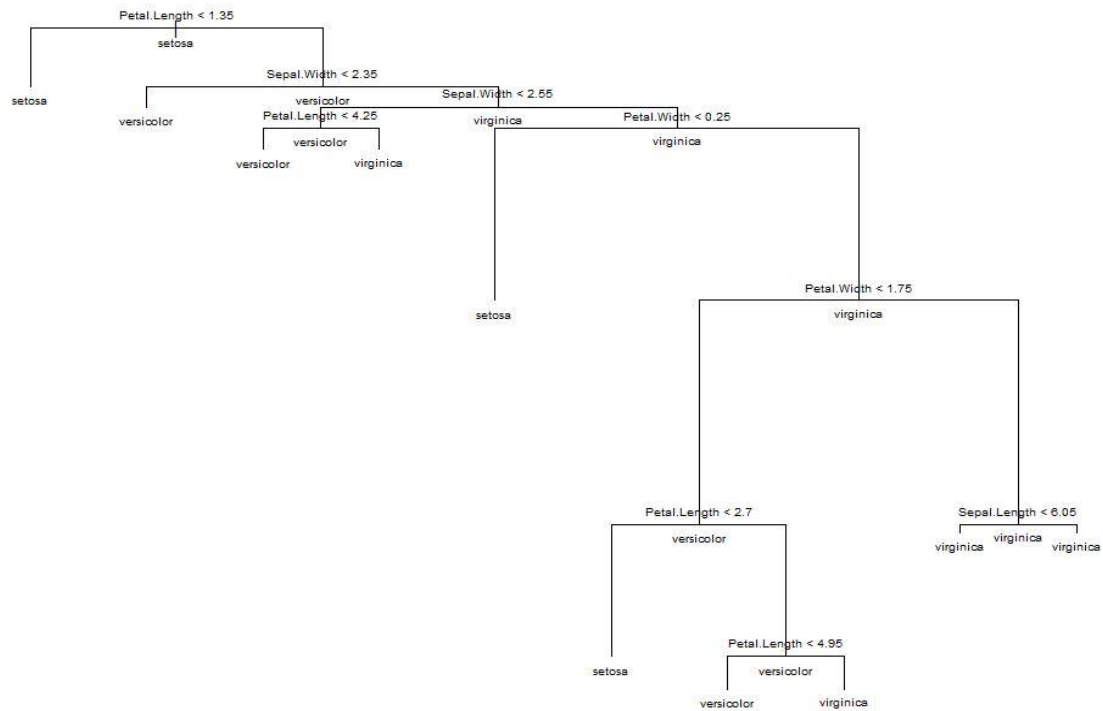
**Decision Tree using R:**

```
> mydata<-data.frame(iris)
> attach(mydata)
> install.packages("rpart")
> library(rpart)
> model<-rpart(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
+             data=mydata,
+             method="class")
> plot(model)
> text(model,use.n=TRUE,all=TRUE,cex=0.8)
```



## VPM's B.N. Bhandodkar College Of Science

```
> install.packages("tree")
> library(tree)
> model1<-tree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
+             data=mydata,
+             method="class",
+             split="gini")
> plot(model1)
> text(model1,all=TRUE,cex=0.6)
```



## VPM's B.N. Bhandodkar College Of Science

```
> install.packages("party")  
> library(party)  
Loading required package: grid  
Loading required package: mvtnorm  
Loading required package: modeltools  
Loading required package: stats4  
Loading required package: strucchange  
Loading required package: zoo
```

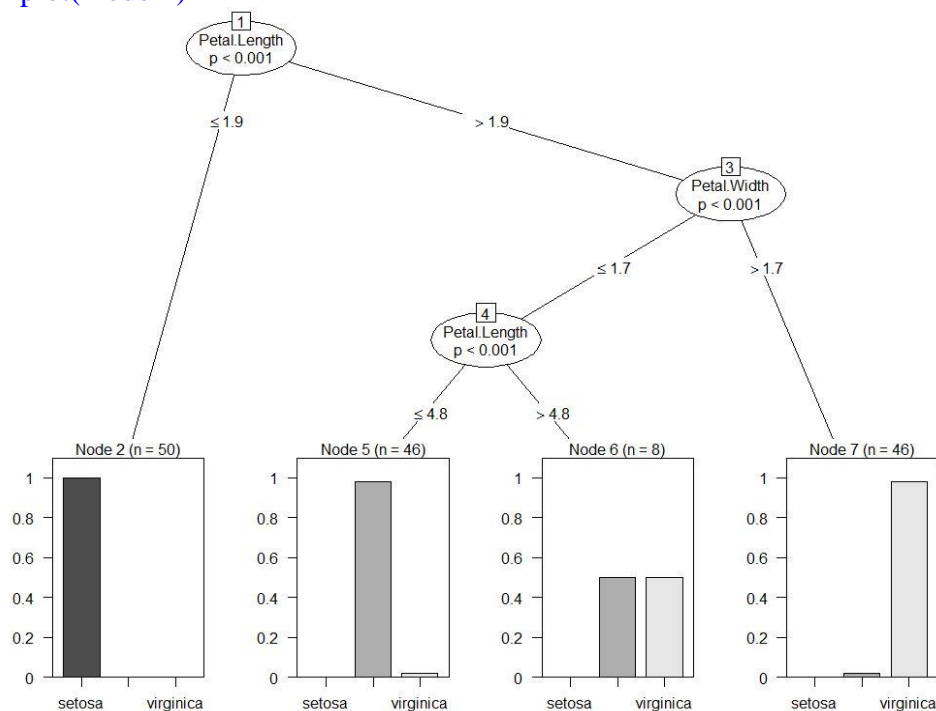
Attaching package: 'zoo'

The following objects are masked from 'package:base':

as.Date, as.Date.numeric

Loading required package: sandwich

```
> model2<-ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,  
+               data=mydata)  
> plot(model2)
```



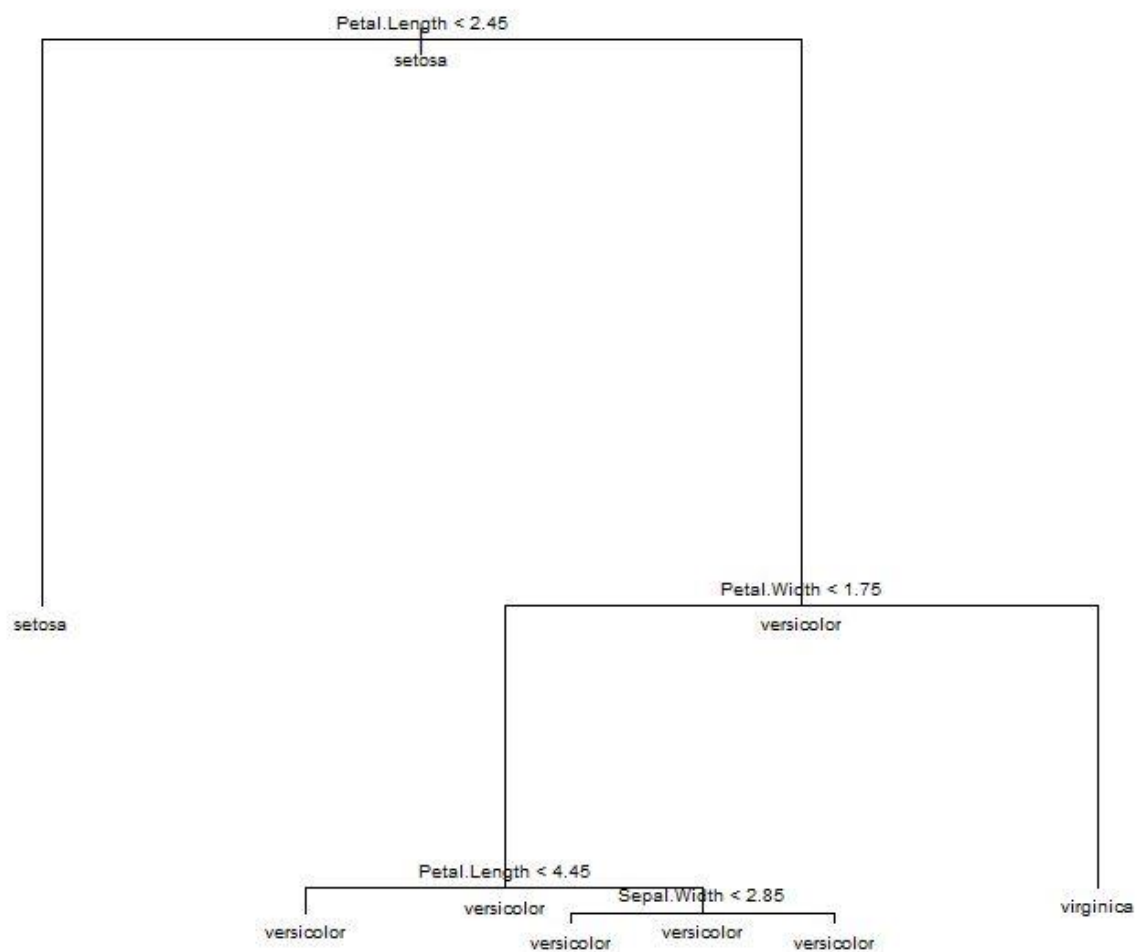
## VPM's B.N. Bandodkar College Of Science

```
> library(tree)
> mydata<-data.frame(iris)
> attach(mydata)
```

The following objects are masked from mydata (pos = 13):

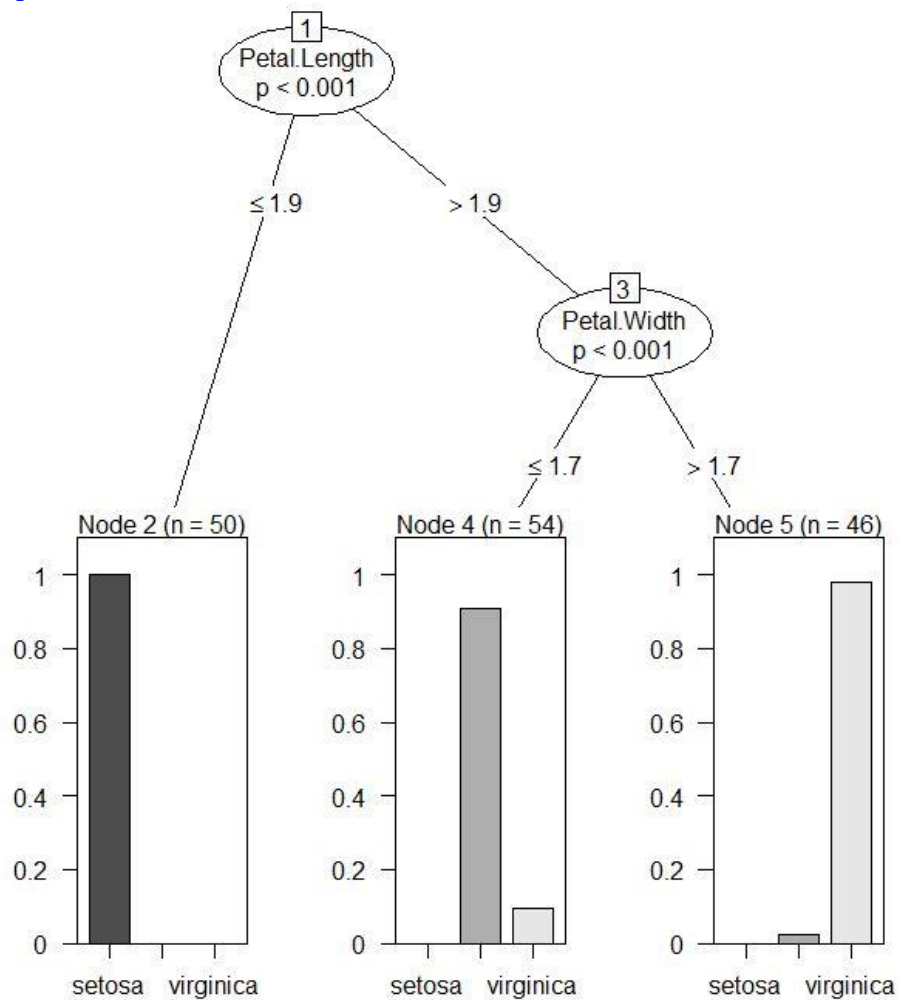
Petal.Length, Petal.Width, Sepal.Length, Sepal.Width, Species

```
> model1<-tree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
+             data=mydata,
+             method="class",
+             control = tree.control(nobs = 150, mincut = 10))
> plot(model1)
> text(model1,all=TRUE,cex=0.6)
```



## VPM's B.N. Bhandodkar College Of Science

```
> predict(model1,iris)
      setosa versicolor virginica
1      1 0.00000000 0.00000000
2      1 0.00000000 0.00000000
3      1 0.00000000 0.00000000
4      1 0.00000000 0.00000000
5      1 0.00000000 0.00000000
.
.
.
148    0 0.02173913 0.97826087
149    0 0.02173913 0.97826087
150    0 0.02173913 0.97826087
> model2<-ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
+               data = mydata, controls = ctree_control(maxdepth=2))
> plot(model2)
```



## VPM's B.N. Bhandodkar College Of Science

### All Command

```
mydata<-data.frame(iris)
attach(mydata)
install.packages("rpart")
library(rpart)
model<-rpart(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
             data=mydata,
             method="class")
plot(model)
text(model,use.n=TRUE,all=TRUE,cex=0.8)

install.packages("tree")
library(tree)
model1<-tree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
             data=mydata,
             method="class",
             split="gini")
plot(model1)
text(model1,all=TRUE,cex=0.6)

install.packages("party")
library(party)
model2<-ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
             data=mydata)
plot(model2)

library(tree)
mydata<-data.frame(iris)
attach(mydata)
model1<-tree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
             data=mydata,
             method="class",
             control = tree.control(nobs = 150, mincut = 10))
plot(model1)
text(model1,all=TRUE,cex=0.6)
predict(model1,iris)

model2<-ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
             data = mydata, controls = ctree_control(maxdepth=2))
plot(model2)
```