UNIT-IV

1. A. How many types of linear models were there? What are they?

ANSWER:

There are 4 types of linear models. They are:

1. Simple Linear Regression
2. Multiple Regression
3. Logistic Regression
4. Poisson Regression

B. Write a short note on Regression Analysis.

ANSWER:

- Regression analysis is very widely used statistical tool to establish a relationship model between two variables.
- One of these variable is called predictor variable whose value is gathered through experiments.
- The other variable is called response variable whose value is derived from the predictor variable.

C. Write a short note on Linear Regression.

ANSWER:

- In linear regression these two variables are related through an equation, where exponent (power) of both these variables is 1.
- Mathematically a linear relationship represents a straight line when plotted as a graph.
- A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.
- The general mathematical equation for a linear regression is –
  $y = ax + b$

Following is the description of the parameters used:

- y is the response variable.
- x is the predictor variable.
- a and b are constants which are called the coefficients.

Steps to Establish a Regression

- A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is −

- Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
- Create a relationship model using the lm() functions in R.
- Find the coefficients from the model created and create the mathematical equation using these
- Get a summary of the relationship model to know the average error in prediction. Also called residuals.
- To predict the weight of new persons, use the predict()function in R.

Input Data:

Below is the sample data representing the observations −

- # Values of height 151, 174, 138, 186, 128, 136, 179, 163, 152, 131
- # Values of weight. 63, 81, 56, 91, 47, 57, 76, 72, 62, 48

lm() Function

- This function creates the relationship model between the predictor and the response variable.

Syntax

- The basic syntax for lm() function in linear regression is −
- lm(formula,data) Following is the description of the parameters used −
- formula is a symbol presenting the relation between x and y.
- data is the vector on which the formula will be applied.

Create relationship model and get the coefficients

```
>x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
>y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
>relation <- lm(y~x)
>print(relation)
```

predict()

Syntax

```
predict(object, newdata)
```

Following is the description of the parameters used:

- object is the formula which is already created using the lm() function.
- newdata is the vector containing the new value for predictor variable.

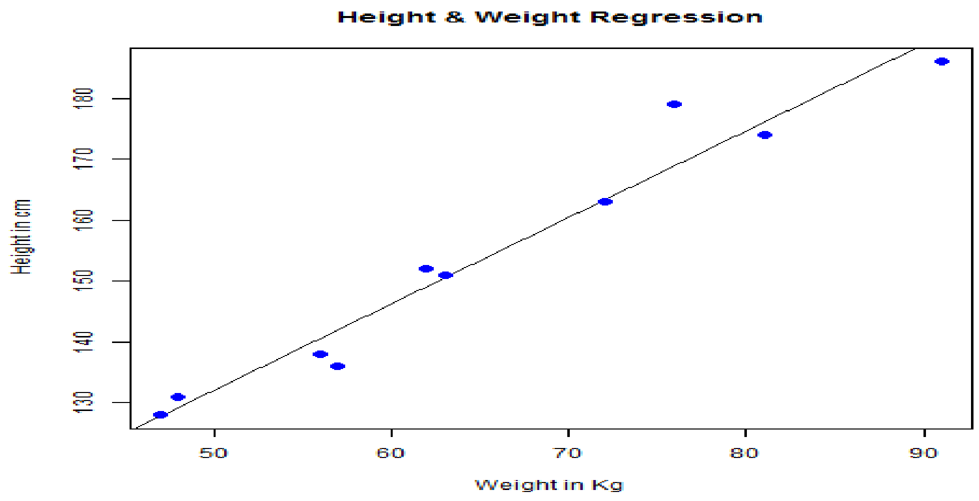
Predict the weight of new persons

```
# The predictor vector.

>x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

# The resposne vector.

>y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.

>relation <- lm(y~x)

# Find weight of a person with height 170.

>a <- data.frame(x = 170)

>result <-  predict(relation,a)

>print(result)

 1

76.22869
```

Visualize the Regression Graphically

```
# Create the predictor and response variable.

>x<- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

>y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

>relation <- lm(y~x)

# Give the chart file a name.

>png(file = "linearregression.png")

# Plot the chart.
```

```
>plot(y,x,col = "blue",main = "Height & Weight Regression",

abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")

# Save the file.

>dev.off()
```



D. Write a short note on Multiple Regression

ANSWER:

- Multiple regression is an extension of linear regression into relationship between more than two variables. In simple linear relation we have one predictor and one response variable, but in multiple regression we have more than one predictor variable and one response variable.

The general mathematical equation for multiple regression is

y = a + b1x1 + b2x2 +...bnxn

Following is the description of the parameters used

- y is the response variable.
- a, b1, b2...bn are the coefficients.
- x1, x2, ...xn are the predictor variables.

We create the regression model using the lm() function in R. The model determines the value of the coefficients using the input data. Next we can predict the value of the response variable for a given set of predictor variables using these coefficients.

lm()

- This function creates the relationship model between the predictor and the response variable.

Syntax

lm(y ~ x1+x2+x3...,data)

Following is the description of the parameters used

- **formula** is a symbol presenting the relation between the response variable and predictor variables.
- **data** is the vector on which the formula will be applied.

Example

Input Data

- Consider the data set "mtcars" available in the R environment. It gives a comparison between different car models in terms of mileage per gallon (mpg), cylinder displacement("disp"), horse power("hp"), weight of the car("wt") and some more parameters.
- The goal of the model is to establish the relationship between "mpg" as a response variable with "disp","hp" and "wt" as predictor variables. We create a subset of these variables from the mtcars data set for this purpose.

```
>input <- mtcars[,c("mpg","disp","hp","wt")]

>print(head(input))

                    mpg    disp   hp     wt

Mazda RX4           21.0   160    110    2.620

Mazda RX4 Wag       21.0   160    110    2.875

Datsun 710          22.8   108     93    2.320

Hornet 4 Drive      21.4   258    110    3.215

Hornet Sportabout   18.7   360    175    3.440

Valiant             18.1   225    105    3.460
```

Create Relationship Model & get the Coeffients

```
input <- mtcars[,c("mpg","disp","hp","wt")]

# Create the relationship model.

model <- lm(mpg~disp+hp+wt, data = input)

# Show the model.

print(model)

# Get the Intercept and coefficients as vector elements.

cat("# # # # The Coefficient Values # # # ","\n")

a <- coef(model)[1]

print(a)

Xdisp <- coef(model)[2]

Xhp <- coef(model)[3]

Xwt <- coef(model)[4]

print(Xdisp)

print(Xhp)

print(Xwt)
```

Result

```
Call:

lm(formula = mpg ~ disp + hp + wt, data = input)

Coefficients:

(Intercept)        disp          hp          wt

  37.105505     -0.000937     -0.031157    -3.800891
```

E. Write a short note on Logistics Regression

ANSWER:

Logistic Regressions

- The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.

The general mathematical equation for logistic regression is

y = 1/(1+e^-(a+b1x1+b2x2+b3x3+...))

Following is the description of the parameters used

y is the response variable.

x is the predictor variable.

a and b are the coefficients which are numeric constants.

The function used to create the regression model is the glm() function.

Syntax

```
glm(formula,data,family)
```

Following is the description of the parameters used

- formula is the symbol presenting the relationship between the variables.
- data is the data set giving the values of these variables.

family is R object to specify the details of the model. It's value is binomial for logistic regression.

Example

- The in-built data set "mtcars" describes different models of a car with their various engine specifications. In "mtcars" data set, the transmission mode (automatic or manual) is described by the column am which is a binary value (0 or 1). We can create a logistic regression model between the columns "am" and 3 other columns - hp, wt and cyl.

```
# Select some columns form mtcars.

>input <- mtcars[,c("am","cyl","hp","wt")]

>print(head(input))
```

Result

```
 am     cyl       hp      wt
```

```
Mazda RX4            1     6     110    2.620
Mazda RX4 Wag        1     6     110    2.875
Datsun 710           1     4      93    2.320
Hornet 4 Drive       0     6     110    3.215
Hornet Sportabout    0     8     175    3.440
Valiant              0     6     105    3.460
```

Create Regression Model

- We use the **glm()** function to create the regression model and get its summary for analysis.

```
>input <- mtcars[,c("am","cyl","hp","wt")]

>am.data = glm(formula = am ~ cyl + hp + wt, data = input, family = binomial)

>print(summary(am.data))
```

Result

```
Call:

glm(formula = am ~ cyl + hp + wt, family = binomial, data = input)

Deviance Residuals:

     Min        1Q      Median       3Q       Max

-2.17272    -0.14907  -0.01464    0.14116   1.27641

Coefficients:

            Estimate Std. Error z value Pr(>|z|)

(Intercept) 19.70288    8.11637   2.428   0.0152 *

cyl          0.48760    1.07162   0.455   0.6491

hp           0.03259    0.01886   1.728   0.0840 .

wt          -9.14947    4.15332  -2.203   0.0276 *
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.2297  on 31  degrees of freedom

Residual deviance:  9.8415  on 28  degrees of freedom

AIC: 17.841

Number of Fisher Scoring iterations: 8
```

Conclusion

- In the summary as the p-value in the last column is more than 0.05 for the variables "cyl" and "hp", we consider them to be insignificant in contributing to the value of the variable "am". Only weight (wt) impacts the "am" value in this regression model.

F. Write a short note on Poisson Regression

ANSWER:

- Poisson Regression involves regression models in which the response variable is in the form of counts and not fractional numbers. For example, the count of number of births or number of wins in a football match series. Also the values of the response variables follow a Poisson distribution.

The general mathematical equation for Poisson regression is −

log(y) = a + b1x1 + b2x2 + bnxn.....

Following is the description of the parameters used −

- y is the response variable.
- a and b are the numeric coefficients.
- x is the predictor variable.

The function used to create the Poisson regression model is the glm() function.

Syntax:

glm(formula,data,family)

- formula is the symbol presenting the relationship between the variables.

- data is the data set giving the values of these variables.
- family is R object to specify the details of the model. It's value is 'Poisson' for Logistic Regression.

Example

- We have the in-built data set "warpbreaks" which describes the effect of wool type (A or B) and tension (low, medium or high) on the number of warp breaks per loom. Let's consider "breaks" as the response variable which is a count of number of breaks. The wool "type" and "tension" are taken as predictor variables.

Input Data

```
>input<-warpbreaks

>print(head(input))
```

Result:

```
  breaks      wool  tension
1 26          A     L
2 30          A     L
3 54          A     L
4 25          A     L
5 70          A     L
6 52          A     L
```

Create Regression Model

```
>output <-glm(formula = breaks ~ wool+tension, data = warpbreaks, family = poisson)

>print(summary(output))
```

Result:

```
Call:

glm(formula = breaks ~ wool + tension, family = poisson, data = warpbreaks)

Deviance Residuals:

   Min     1Q   Median    3Q    Max
```

```
  -3.6871  -1.6503  -0.4269     1.1902   4.2616

Coefficients:

            Estimate Std. Error z value Pr(>|z|)

(Intercept)  3.69196    0.04541  81.302  < 2e-16 ***

woolB       -0.20599    0.05157  -3.994 6.49e-05 ***

tensionM    -0.32132    0.06027  -5.332 9.73e-08 ***

tensionH    -0.51849    0.06396  -8.107 5.21e-16 ***

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 297.37  on 53  degrees of freedom

Residual deviance: 210.39  on 50  degrees of freedom

AIC: 493.06

Number of Fisher Scoring iterations: 4
```

---

2. A. Write a short note on Non-Linear Models

ANSWER:

- When modeling real world data for regression analysis, we observe that it is rarely the case that the equation of the model is a linear equation giving a linear graph.
- Most of the time, the equation of the model of real world data involves mathematical functions of higher degree like an exponent of 3 or a sin function.
- In such a scenario, the plot of the model gives a curve rather than a line. The goal of both linear and non-linear regression is to adjust the values of the model's parameters to find the line or curve that comes closest to your data.
- On finding these values we will be able to estimate the response variable with good accuracy.

B. How many types of Non-Linear Models were there? What are they?

ANSWER:

There are four types. they are:

1. Nonlinear least squares
2. Generalized additive models
3. Decision trees
4. Random forests

C. write a short note on Non-linear least squares.

ANSWER:

In Least Square regression, we establish a regression model in which the sum of the squares of the vertical distances of different points from the regression curve is minimized. We generally start with a defined model and assume some values for the coefficients. We then apply the **nls()** function of R to get the more accurate values along with the confidence intervals.

Syntax

The basic syntax for creating a nonlinear least square test in R is −

```
nls(formula, data, start)
```

Following is the description of the parameters used −

- **formula** is a nonlinear model formula including variables and parameters.
- **data** is a data frame used to evaluate the variables in the formula.
- **start** is a named list or named numeric vector of starting estimates.

Example

We will consider a nonlinear model with assumption of initial values of its coefficients. Next we will see what is the confidence intervals of these assumed values so that we can judge how well these values fir into the model.

So let's consider the below equation for this purpose −

```
a = b1*x^2+b2
```

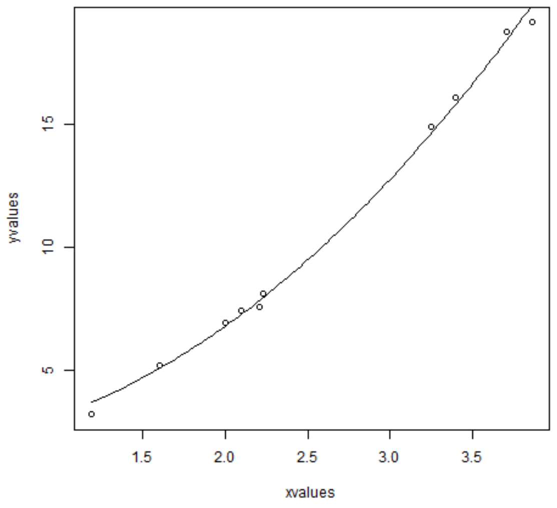
Let's assume the initial coefficients to be 1 and 3 and fit these values into nls() function.

```
xvalues <- c(1.6,2.1,2,2.23,3.71,3.25,3.4,3.86,1.19,2.21)
yvalues <- c(5.19,7.43,6.94,8.11,18.75,14.88,16.06,19.12,3.21,7.58)

# Give the chart file a name.
png(file = "nls.png")


# Plot these values.
```

```
plot(xvalues,yvalues)


# Take the assumed values and fit into the model.
model <- nls(yvalues ~ b1*xvalues^2+b2,start = list(b1 = 1,b2 = 3))

# Plot the chart with new data by fitting it to a prediction from
100 data points.
new.data <- data.frame(xvalues = seq(min(xvalues),max(xvalues),len
= 100))
lines(new.data$xvalues,predict(model,newdata = new.data))

# Save the file.
dev.off()

# Get the sum of the squared residuals.
print(sum(resid(model)^2))

# Get the confidence intervals on the chosen values of the
coefficients.
print(confint(model))
```

Output:

```
1] 1.081935
Waiting for profiling to be done...
       2.5%    97.5%
b1 1.137708 1.253135
b2 1.497364 2.496484
```



D. Write a short note on splines

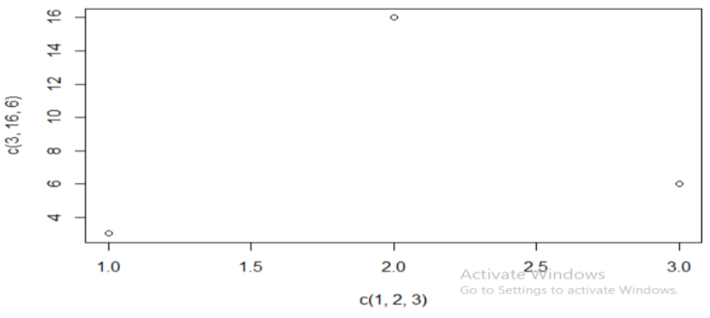
Answer:

- Polynomial regression only captures a certain amount of curvature in a nonlinear relationship. An alternative, and often superior, approach to modeling nonlinear relationships is to use splines (P. Bruce and Bruce 2017).
- Splines provide a way to smoothly interpolate between fixed points, called knots. Polynomial regression is computed between knots. In other words, splines are series of polynomial segments strung together, joining at knots (P. Bruce and Bruce 2017).
- The R package splines includes the function bs for creating a b-spline term in a regression model.
- You need to specify two parameters: the degree of the polynomial and the location of the knots. In our example, we'll place the knots at the lower quartile, the median quartile, and the upper quartile:

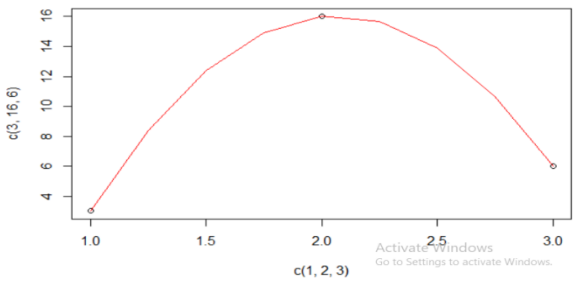
We'll create a model using a cubic spline (degree = 3):

Code:

```
>spline(x=c(1,2,3),y=c(3,16,6),n=9)

$x

[1] 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00

$y

[1]  3.00000  8.40625 12.37500 14.90625 16.00000

[6] 15.65625 13.87500 10.65625  6.00000

>splineout<-spline(x=c(1,2,3),y=c(3,16,6),n=9)

>splineout$x

[1] 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00

>splineout$y

[1] 3.00000 8.40625 12.37500 14.90625 16.00000

[6] 15.65625 13.87500 10.65625 6.00000

>plot(c(1,2,3),c(3,16,6))
```



>points(splineout$x,splineout$y,type='l',col="red")



E. Write a short note on GAM(Generalized Additive Models)

Answer:

Generalized additive models

- Once you have detected a non-linear relationship in your data, the polynomial terms may not be flexible enough to capture the relationship, and spline terms require specifying the knots.
- Generalized additive models, or GAM, are a technique to automatically fit a spline regression. This can be done using the mgcv R package:

Code:

```
install.packages("faraway")

library(faraway)

data("ozone")

install.packages("mgcv")

library(mgcv)

gamobj<-gam(log(O3)~s(vh)+s(wind)+s(humidity)+s(temp)+s(ibh)+
```

s(dpg)+s(ibt)+s(vis)+s(doy),

family=gaussian(link=identity),data=ozone)

summary(gamobj)

pdf("GAMozone.pdf")

par(mfrow=c(3,3))

plot(gamobj)

dev.off()

F. Write a short note on decision trees.

Answer:

- Decision tree is a graph to represent choices and their results in form of a tree. The nodes in the graph represent an event or choice and the edges of the graph represent the decision rules or conditions.
- It is mostly used in Machine Learning and Data Mining applications using R.
- Examples of use of decision tress is − predicting an email as spam or not spam, predicting of a tumor is cancerous or predicting a loan as a good or bad credit risk based on the factors in each of these.
- Generally, a model is created with observed data also called training data. Then a set of validation data is used to verify and improve the model. R has packages which are used to create and visualize decision trees.
- For new set of predictor variable, we use this model to arrive at a decision on the category (yes/No, spam/not spam) of the data.

Syntax:

```
ctree(formula, data)
```

- **formula** is a formula describing the predictor and response variables.
- **data** is the name of the data set used.

Example:

```
# Load the party package. It will automatically load other
# dependent packages.
library(party)
```

```
# Create the input data frame.
input.dat <- readingSkills[c(1:105),]

# Give the chart file a name.
png(file = "decision_tree.png")

# Create the tree.
  output.tree <- ctree(
  nativeSpeaker ~ age + shoeSize + score,
  data = input.dat)

# Plot the tree.
plot(output.tree)

# Save the file.
dev.off()
```

G. Write a short note on Random Forests.

Answer:

- In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. A new observation is fed into all the trees and taking a majority vote for each classification model.
- An error estimate is made for the cases which were not used while building the tree. That is called an OOB (Out-of-bag) error estimate which is mentioned as a percentage.
- The R package "randomForest" is used to create random forests.

Syntax:

randomForest(formula,data)

- Formula : describing the predictor and response variables.
- Data is the name of the data set used.

Example:

```
# Load the party package. It will automatically load other
# required packages.
library(party)
library(randomForest)
```

```
# Create the forest.
output.forest <- randomForest(nativeSpeaker ~ age + shoeSize +
score,
           data = readingSkills)

# View the forest results.
print(output.forest)

# Importance of each predictor.
print(importance(fit,type = 2))
```

3. A. Write a short note on Time-Series and Auto-correlation.

Answer:

- A big part of statistics, particularly for financial and econometric data, is analyzing time series, data that are autocorrelated over time.
- That is, one observation depends on previous observations and the order matters.

B. Write a short note on Autoregressive Moving Objects.

Answer:

- One of the most common ways of fitting time series models is to use autoregressive (AR), moving average(MA) or both (ARMA).
- AR model can be thought of as linear regressions of the current value of the time series against previous values.
- MA models are, similarly, linear regressions of the current value of the time series against current and previous residuals.

Syntax:

```
timeseries.object.name <-  ts(data, start, end, frequency)
```

Following is the description of the parameters used −

- **data** is a vector or matrix containing the values used in the time series.
- **start** specifies the start time for the first observation in time series.
- **end** specifies the end time for the last observation in time series.
- **frequency** specifies the number of observations per unit time.

Example:

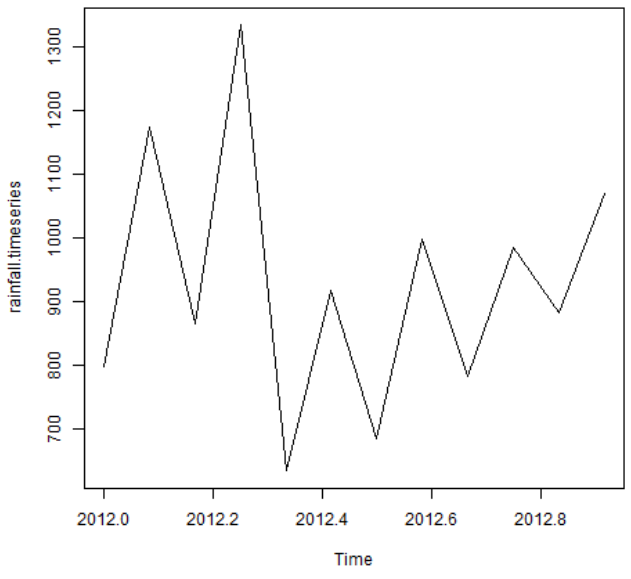
```
# Get the data points in form of a R vector.
```

```
rainfall <-
c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1
071)

# Convert it to a time series object.
rainfall.timeseries <- ts(rainfall,start = c(2012,1),frequency =
12)

# Print the timeseries data.
print(rainfall.timeseries)

# Give the chart file a name.
png(file = "rainfall.png")

# Plot a graph of the time series.
plot(rainfall.timeseries)

# Save the file.
dev.off()
```

When we execute the above code, it produces the following result and chart −

```
Jan     Feb     Mar     Apr     May      Jun     Jul     Aug     Sep
2012  799.0  1174.8  865.1  1334.6  635.4  918.5  685.5  998.6
784.2
        Oct     Nov     Dec
2012  985.0  882.8 1071.0
```

The Time series chart −



C. Write a short note on VAR

ANSWER:

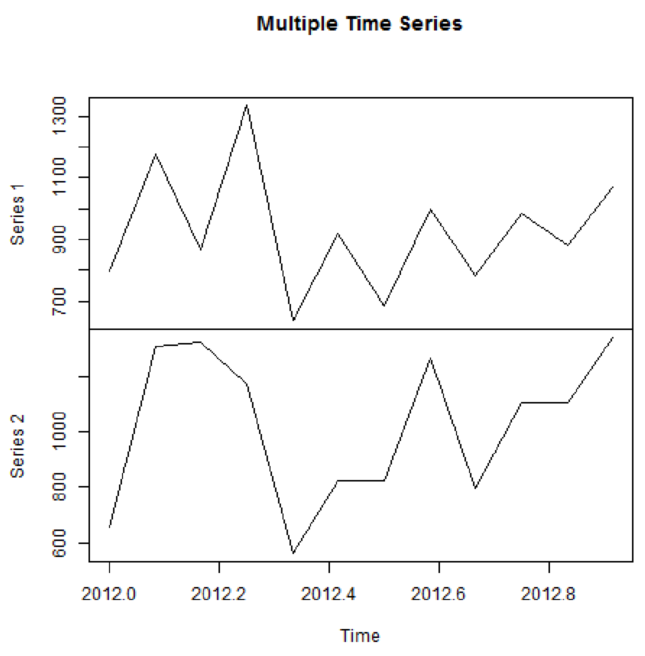
- When dealing with multiple time series where each depends on its own past, others pasts and other presents, things gets more complicated.
- The first thing we will do is convert all of the GDP data into a multivariate time series.

To do this we first cast the data.frame to wide format then call ts to convert it.

Example:

Live Demo

```
# Get the data points in form of a R vector.
rainfall1 <-
c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1
071)
rainfall2 <-

c(655,1306.9,1323.4,1172.2,562.2,824,822.4,1265.5,799.6,1105.6,1106
.7,1337.8)

# Convert them to a matrix.
combined.rainfall <-  matrix(c(rainfall1,rainfall2),nrow = 12)

# Convert it to a time series object.
rainfall.timeseries <- ts(combined.rainfall,start =
c(2012,1),frequency = 12)

# Print the timeseries data.
print(rainfall.timeseries)

# Give the chart file a name.
png(file = "rainfall_combined.png")

# Plot a graph of the time series.
plot(rainfall.timeseries, main = "Multiple Time Series")

# Save the file.
dev.off()
```

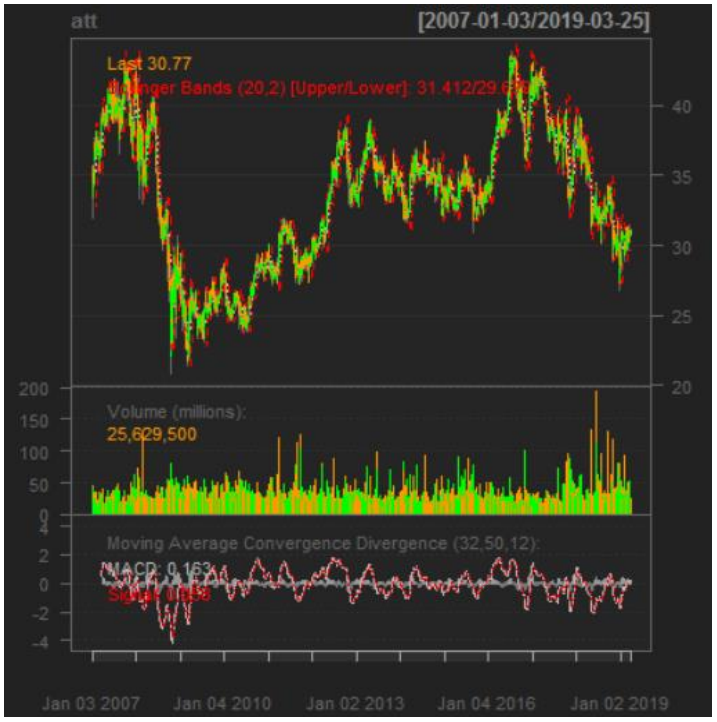


D. Write a short note on GARCH

ANSWER:

- A problem with ARMA models is that they do not handle extreme events or high volatility well.
- To overcome this a good tool to use is generalized autoregressive conditional heteroskedasticity or the GARCH family of models, which in addition to modeling the mean of the process also model the variance.

```
>require(quantmod)

>att<-getSymbols("T",auto.assign=FALSE)

>att

>require(xts)

>head(att)

>plot(att)

>chartSeries(att)

>addBBands()

>addMACD(32,50,12)
```



4. A. Write a short note on clustering

Answer:

- Play's a big role in modern machine learning, is the partitioning of data into groups.
- This can be done in a number of ways, the two most popular being K-means and hierarchical clustering.
- In terms of a data.frame, a clustering algorithm finds out which rows are similar to each other.

Rows that are grouped together are supposed to have high similarity to each other and low similarity with rows outside the grouping

B. Write a short note on k-means clustering

ANSWER:

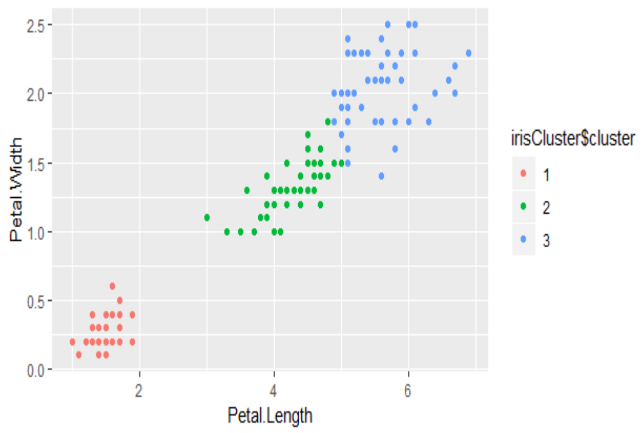
- One of the more popular algorithms for clustering is K-means.
- It divides the observations into discrete groups based on some distance metric.
- For K-means we need to specify the number of clusters, and then the algorithm assigns observations into that many clusters.

- In R, K-means is done with the aptly named K-means function. The first 2 arguments are the data to be clustered, which must be all numeric(K-means does not work with categorical data), and the number of centers(clusters).
- Printing the K-means objects displays the size of the clusters, the cluster mean for each column, the cluster membership for each row and similarity measures.
- Plotting the results of K-means clustering can be difficult because of high dimensional nature of the data. To overcome this, the plot.kmeans function in useful performs multidimensional scaling to project the data into two dimensions, and then color codes the points according to cluster membership.

Code:

```
>library(ggplot2)

>ggplot(iris,aes(Petal.Length,Petal.Width,color=Species))+geom_point()

>set.seed(20)

>irisCluster<-kmeans(iris[,3:4],3,nstart=20)

>irisCluster

Clustering vector:

  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 [25] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 [49] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

 [73] 2 2 2 2 2 3 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2

 [97] 2 2 2 2 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2

[121] 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3

[145] 3 3 3 3 3 3

Within cluster sum of squares by cluster:

[1]  2.02200 13.05769 16.29167

 (between_SS / total_SS =  94.3 %)

Available components:

[1] "cluster"      "centers"      "totss"
```

```
[4] "withinss"     "tot.withinss" "betweenss"

[7] "size"        "iter"        "ifault"

>table(irisCluster$cluster,iris$Species)

     setosa  versicolor        virginica

 1    50                 0                 0

 2     0         48                 4

 3     0         2                  46

> irisCluster$cluster<-as.factor(irisCluster$cluster)

>irisCluster$cluster

 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 [25] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 [49] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

 [73] 2 2 2 2 2 3 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2

 [97] 2 2 2 2 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2

[121] 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3

[145] 3 3 3 3 3 3

Levels: 1 2 3

>ggplot(iris,aes(Petal.Length,Petal.Width, color=irisCluster$cluster))+geom_point()
```



C. Write a short note on PAM/K-Mediods

ANSWER:

- Two problems with K-means clustering are that it does not work with categorical data and it is susceptible to outliers.
- An alternative is K-medoids. Instead of the center of a cluster being the mean of the cluster, the center is one of the actual observations in the cluster.
- The most common K-mediods algorithm is Partitioning Around Mediods(PAM). The cluster package contains the pam().
- If the data has a few missing values, pam handles missing values well. Before we run the clustering algorithm we clean up the data some more.
- Now we fit the clustering using pam from the cluster package. Each line represents an observation, and each grouping of lines is a cluster.
- Observations that fit the cluster well have large positive lines and observations that do not fit well have small or negative lines.

A bigger average width for a cluster means a better clustering

Code:

```
>x<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5)),

     cbind(rnorm(15,5,0.5),rnorm(15,5,0.5)))

>library(pamr)

>pamx<-pam(x,2)

>pamx

Medoids:

   ID

[1,] 10 0.2117409 -0.1906546

[2,] 22 4.9652817  4.9246166

Clustering vector:

 [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Objective function:
    build      swap
0.8236515 0.6037261
Available components:
 [1] "medoids"    "id.med"     "clustering" "objective"
 [5] "isolation"  "clusinfo"   "silinfo"    "diss"
 [9] "call"       "data"
>summary(pamx)
Medoids:
     ID
[1,] 10 0.2117409 -0.1906546
[2,] 22 4.9652817  4.9246166
Clustering vector:
 [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
Objective function:
    build      swap
0.8236515 0.6037261
Numerical information per cluster:
     size  max_diss   av_diss diameter separation
[1,]   10 1.1593800 0.6674374 1.893649   5.528942
[2,]   15 0.9549435 0.5612518 1.583682   5.528942
Isolated clusters:
 L-clusters: character(0)
 L*-clusters: [1] 1 2
>(p2m<-pam(x,2,medoids=c(1,16)))
```

```
>p2.s<-pam(x,2,medoids=c(1,16),do.swap=FALSE)

>p2.s

>p3m<-pam(x,3,trace=2)
```

D. Write a short note on Hierarchical Clustering

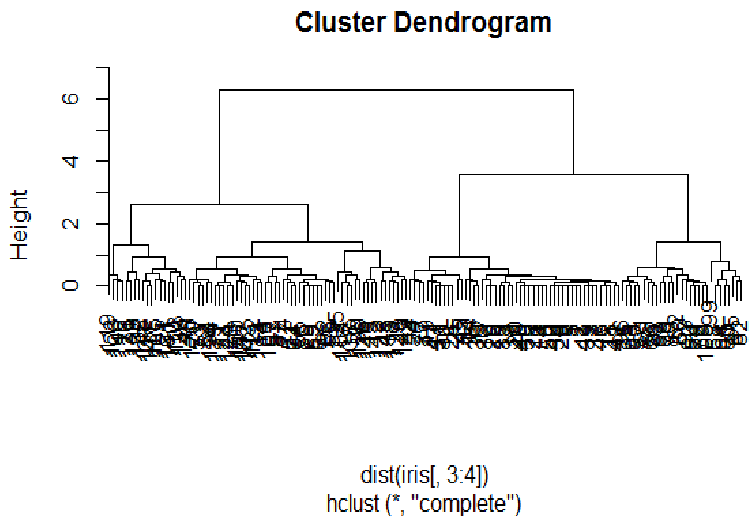
ANSWER:

Hierarchical Clustering:

- Hierarchical Clustering builds clusters within clusters, and does not require a prespecified number of cluster like K-means and K-medoids do.
- A Hierarchical Clustering can be thought of as a tree and displayed as a dendogram, at the top there is just one cluster consisting of all the observations, and at the bottom each observation is an entire cluster.
- In between are varying levels of clustering.
- Hierarchical clustering also works on categorical data like the country information data. However, its dissimilarity matrix must be calculated differently.
- There are a number of ways to compute the distance between clusters and they can have a significant impact on the results of a hierarchical clustering.

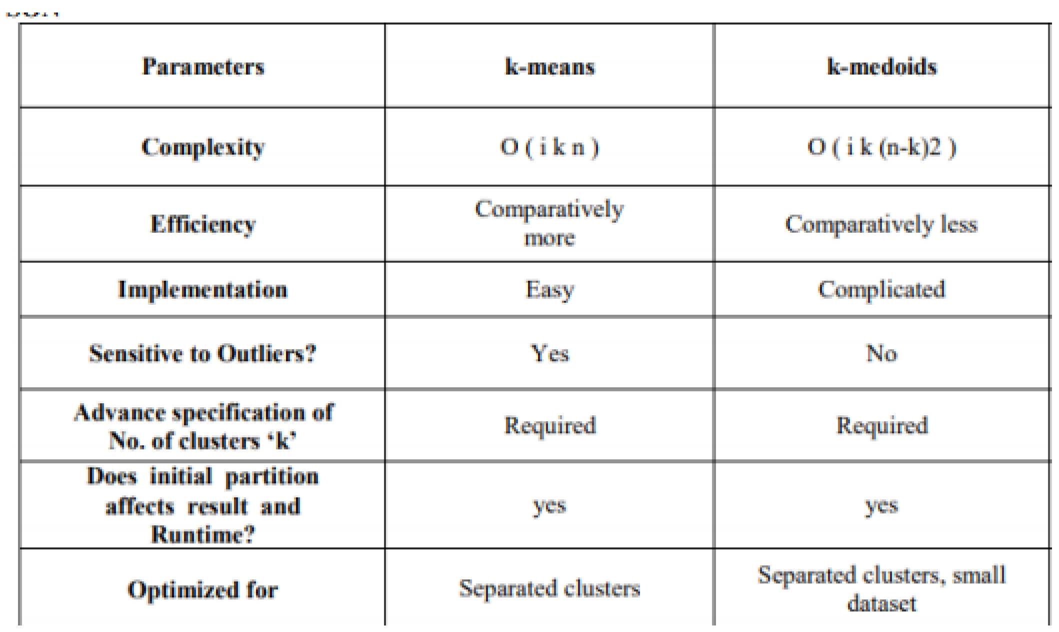
Code:

```
>clusters<-hclust(dist(iris[,3:4]))
>clusters
Call:
hclust(d = dist(iris[, 3:4]))
Cluster method   : complete
Distance         : euclidean
Number of objects: 150
>plot(clusters)
```



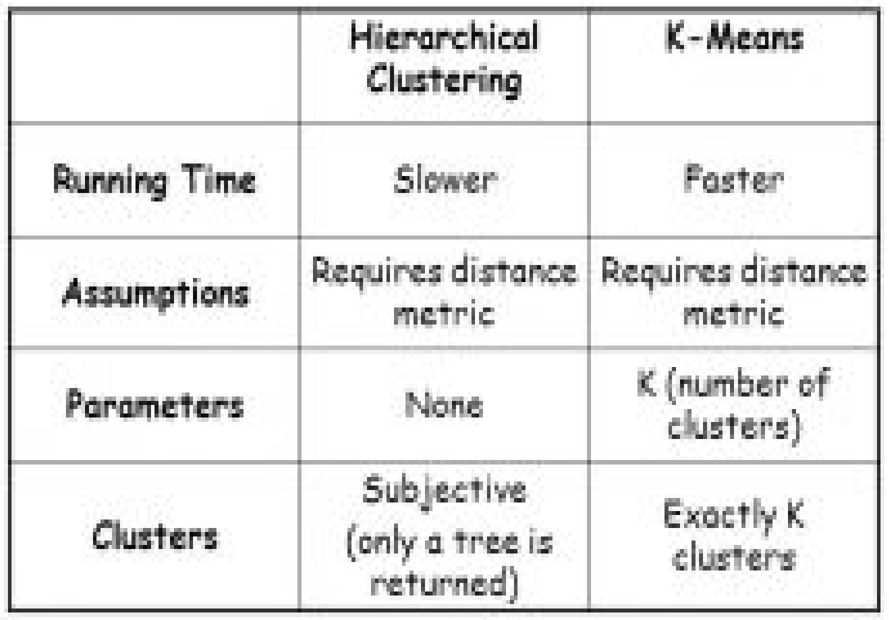
E. Difference between K-means & PAM

ANSWER:



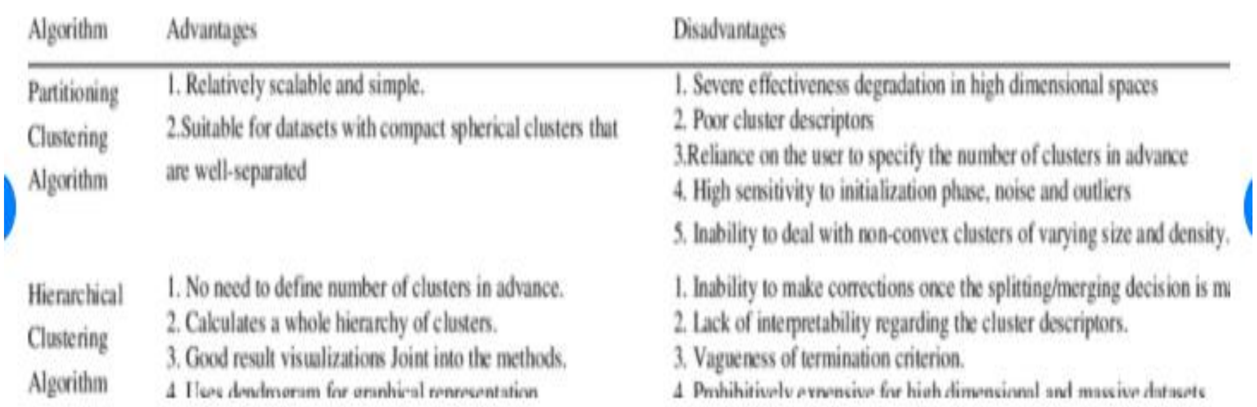
Parameters	k-means	k-medoids
Complexity	O (i k n)	O (i k (n-k)2)
Efficiency	Comparatively more	Comparatively less
Implementation	Easy	Complicated
Sensitive to Outliers?	Yes	No
Advance specification of No. of clusters 'k'	Required	Required
Does initial partition affects result and Runtime?	yes	yes
Optimized for	Separated clusters	Separated clusters, small dataset

F. Difference between K-means and hierarchical clustering



	Hierarchical Clustering	K-Means
Running Time	Slower	Faster
Assumptions	Requires distance metric	Requires distance metric
Parameters	None	K (number of clusters)
Clusters	Subjective (only a tree is returned)	Exactly K clusters

G. Difference between PAM and Hierarchical Clustering



Algorithm	Advantages	Disadvantages
Partitioning Clustering Algorithm	1. Relatively scalable and simple. 2.Suitable for datasets with compact spherical clusters that are well-separated	1. Severe effectiveness degradation in high dimensional spaces 2. Poor cluster descriptors 3.Reliance on the user to specify the number of clusters in advance 4. High sensitivity to initialization phase, noise and outliers 5. Inability to deal with non-convex clusters of varying size and density.
Hierarchical Clustering Algorithm	1. No need to define number of clusters in advance. 2. Calculates a whole hierarchy of clusters. 3. Good result visualizations Joint into the methods. 4. Uses dendrogram for graphical representation	1. Inability to make corrections once the splitting/merging decision is m 2. Lack of interpretability regarding the cluster descriptors. 3. Vagueness of termination criterion. 4. Prohibitively expensive for high dimensional and massive datasets