SAKARYA
ÜNİVERSİTESİ

BIG DATA

TOO BIG TO IGNORE

SÜMEYYE KAYNAK

OUTLINE

SAKARYA ÜNİVERSİTESİ

Big Data Analysis Techniques with R programming

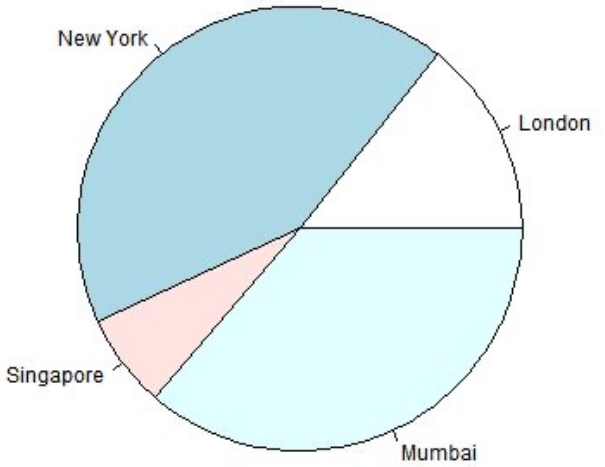
R CHARTS & GRAPHS-PIE

```
pie(x, labels, radius, main, col, clockwise)
```

- **x** is a vector containing the numeric values used in the pie chart.
- **labels** is used to give description to the slices.
- **radius** indicates the radius of the circle of the pie chart.(value between −1 and +1).
- **main** indicates the title of the chart.
- **col** indicates the color palette.
- **clockwise** is a logical value indicating if the slices are drawn clockwise or anti clockwise.

EXAMPLE-PIE

```
> x <- c(21, 62, 10, 53)
> labels <- c("London", "New York", "Singapore", "Mumbai")
> png(file = "city.png")
> pie(x,labels)
> dev.off()
null device
          1
```

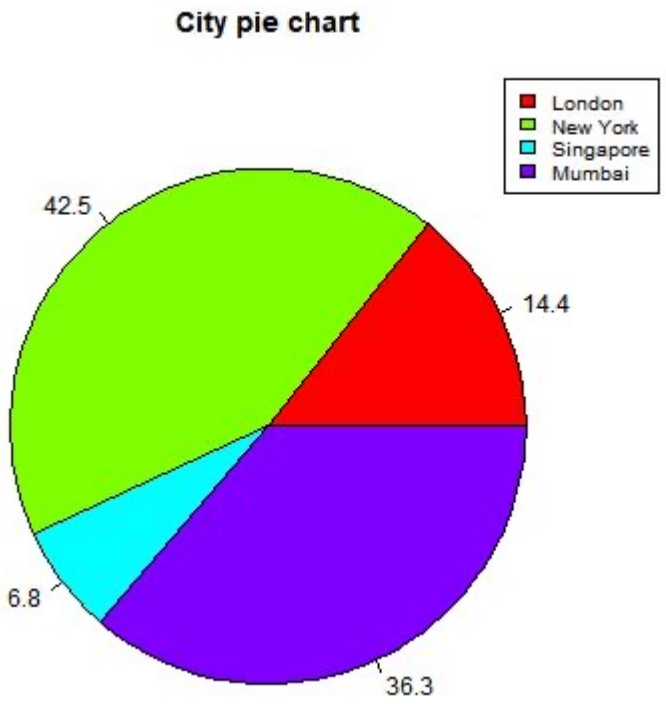


PIE CHART TITLE AND COLORS

```
> # Create data for the graph.
> x <- c(21, 62, 10, 53)
> labels <- c("London", "New York", "Singapore", "Mumbai")
>
> # Give the chart file a name.
> png(file = "city_title_colours.jpg")
>
> # Plot the chart with title and rainbow color pallet.
> pie(x, labels, main = "City pie chart", col = rainbow(length(x)))
>
> # Save the file.
> dev.off()
null device
          1
```

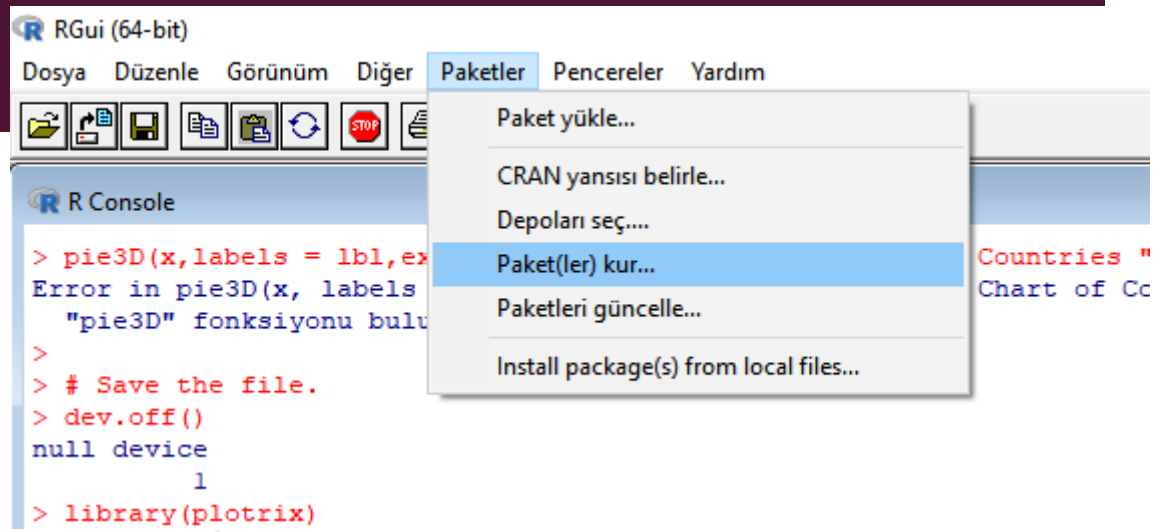
SLICE PERCENTAGES AND CHART LEGEND

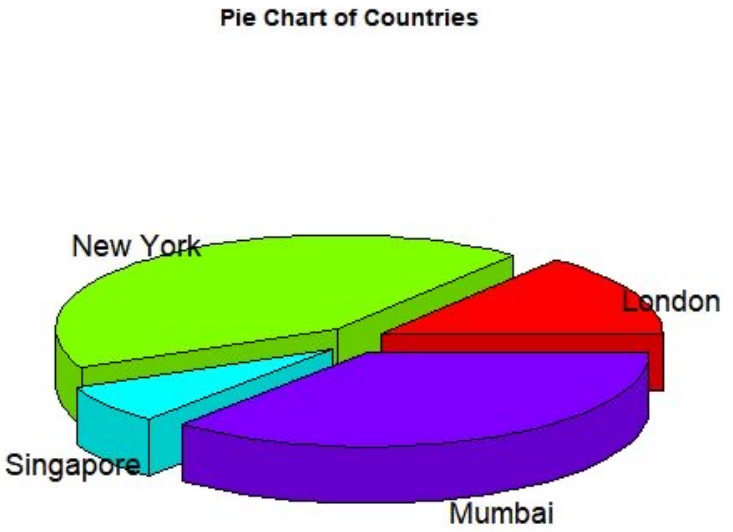
```
> # Create data for the graph.
> x <-  c(21, 62, 10,53)
> labels <-  c("London","New York","Singapore","Mumbai")
>
> piepercent<- round(100*x/sum(x), 1)
>
> # Give the chart file a name.
> png(file = "city_percentage_legends.jpg")
>
> # Plot the chart.
> pie(x, labels = piepercent, main = "City pie chart",col = rainbow(length(x)))
> legend("topright", c("London","New York","Singapore","Mumbai"), cex = 0.8,
+    fill = rainbow(length(x)))
>
> # Save the file.
> dev.off()
null device
          1
```



3D PIE CHART

```
> # Get the library.
> library(plotrix)
Error in library(plotrix) : there is no package called 'plotrix'
>
> # Create data for the graph.
> x <-  c(21, 62, 10,53)
> lbl <-  c("London","New York","Singapore","Mumbai")
>
> # Give the chart file a name.
> png(file = "3d_pie_chart.jpg")
>
> # Plot the chart.
> pie3D(x,labels = lbl,explode = 0.1, main = "Pie Chart of Countries ")
Error in pie3D(x, labels = lbl, explode = 0.1, main = "Pie Chart of Countries "$
  "pie3D" fonksiyonu bulunamadı
>
> # Save the file.
> dev.off()
null device
          1
```





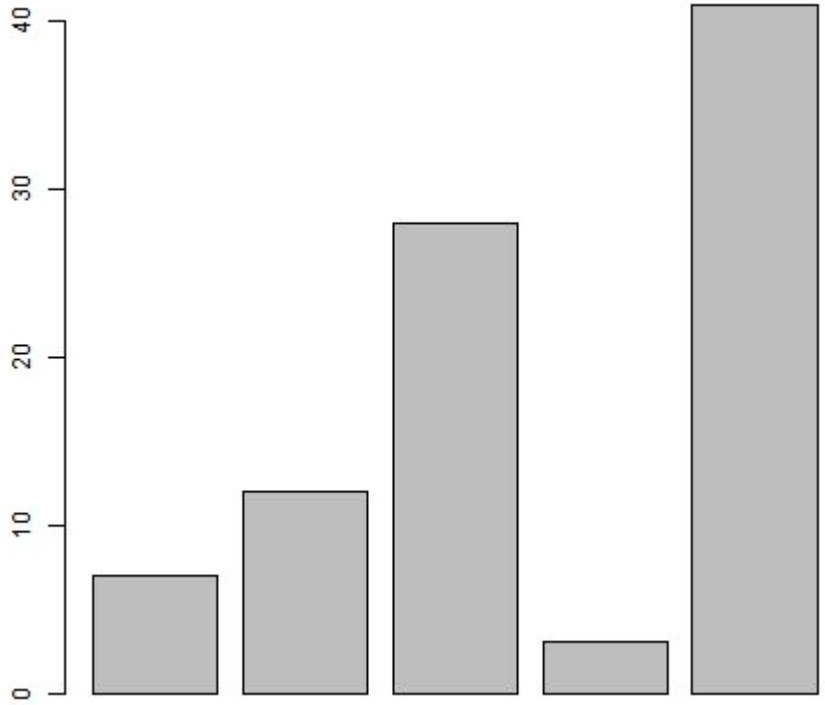
R - BAR CHARTS

```
barplot(H,xlab,ylab,main, names.arg,col)
```

- **H** is a vector or matrix containing numeric values used in bar chart.
- **xlab** is the label for x axis.
- **ylab** is the label for y axis.
- **main** is the title of the bar chart.
- **names.arg** is a vector of names appearing under each bar.
- **col** is used to give colors to the bars in the graph.

EXAMPLE - BAR CHARTS

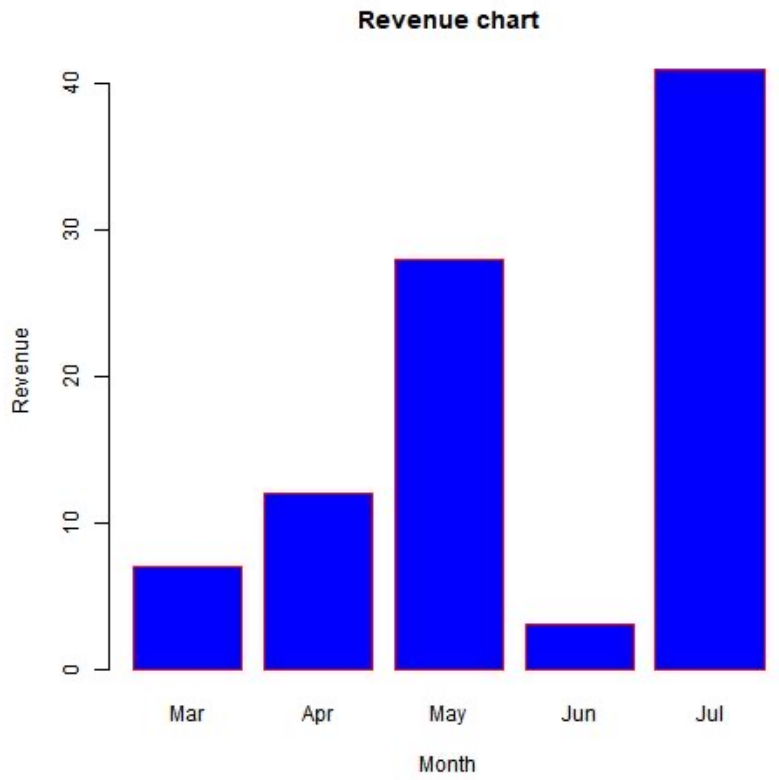
```
> # Create the data for the chart
> H <- c(7,12,28,3,41)
>
> # Give the chart file a name
> png(file = "barchart.png")
>
> # Plot the bar chart
> barplot(H)
>
> # Save the file
> dev.off()
null device
          1
```



BAR CHART LABELS, TITLE AND COLORS

R Console

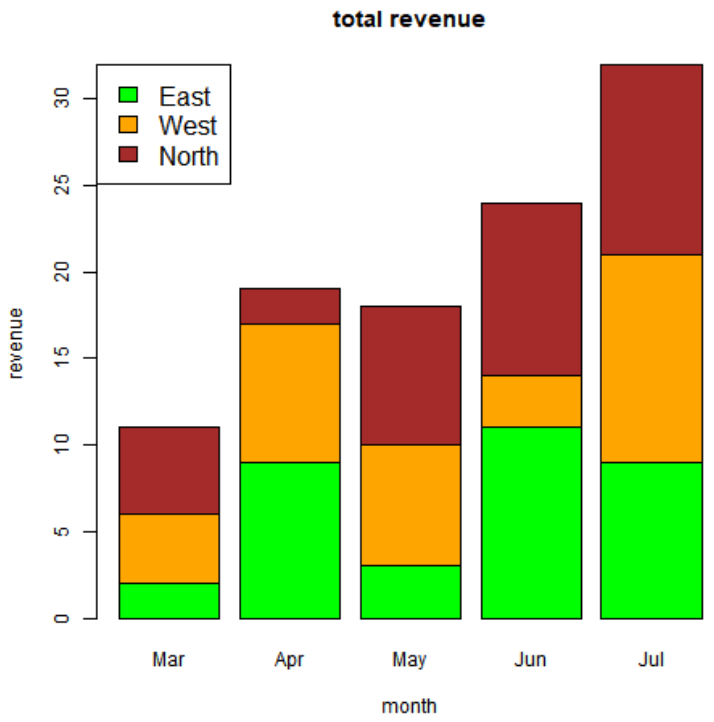
```
> # Create the data for the chart
> H <- c(7,12,28,3,41)
> M <- c("Mar","Apr","May","Jun","Jul")
>
> # Give the chart file a name
> png(file = "barchart_months_revenue.png")
>
> # Plot the bar chart
> barplot(H,names.arg=M,xlab="Month",ylab="Revenue",col="blue",
+ main="Revenue chart",border="red")
>
> # Save the file
> dev.off()
null device
          1
```



GROUP BAR CHART AND STACKED BAR CHART

R Console

```
> # Create the input vectors.
> colors = c("green","orange","brown")
> months <- c("Mar","Apr","May","Jun","Jul")
> regions <- c("East","West","North")
>
> # Create the matrix of the values.
> Values <- matrix(c(2,9,3,11,9,4,8,7,3,12,5,2,8,10,11), nrow = 3, ncol = 5, by$
>
> # Give the chart file a name
> png(file = "barchart_stacked.png")
>
> # Create the bar chart
> barplot(Values, main = "total revenue", names.arg = months, xlab = "month", y$
>
> # Add the legend to the chart
> legend("topleft", regions, cex = 1.3, fill = colors)
>
> # Save the file
> dev.off()
null device
          1
```



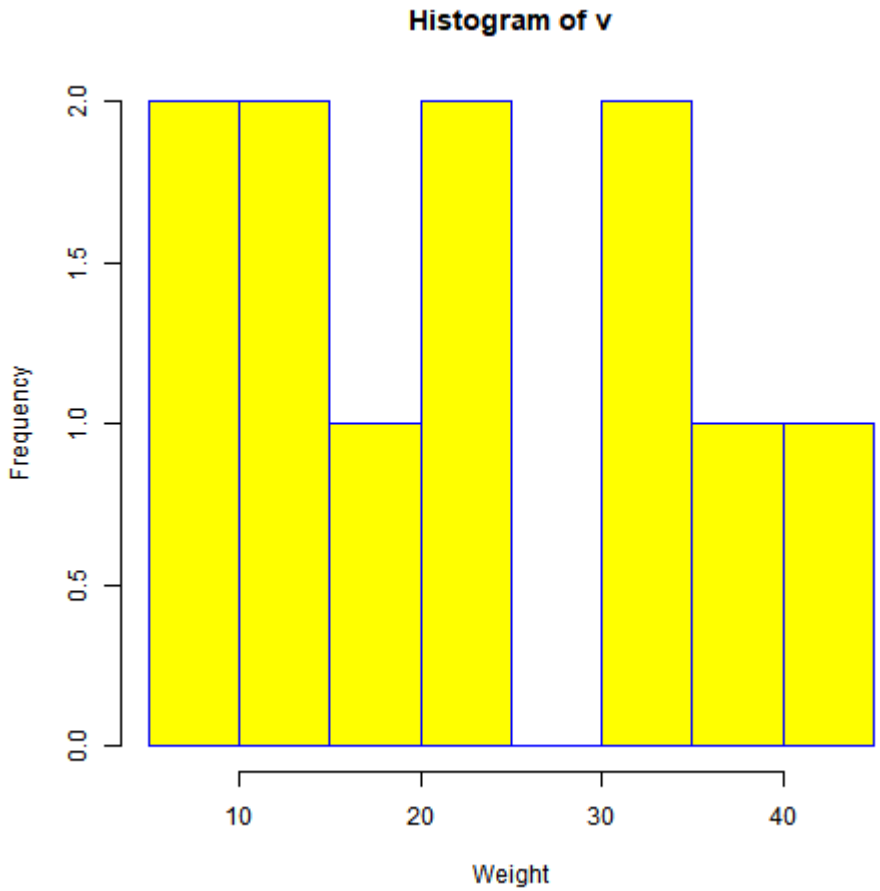
R - HISTOGRAMS

```
hist(v,main,xlab,xlim,ylim,breaks,col,border)
```

- **v** is a vector containing numeric values used in histogram.
- **main** indicates title of the chart.
- **col** is used to set color of the bars.
- **border** is used to set border color of each bar.
- **xlab** is used to give description of x-axis.
- **xlim** is used to specify the range of values on the x-axis.
- **ylim** is used to specify the range of values on the y-axis.
- **breaks** is used to mention the width of each bar.

EXAMPLE - HISTOGRAMS

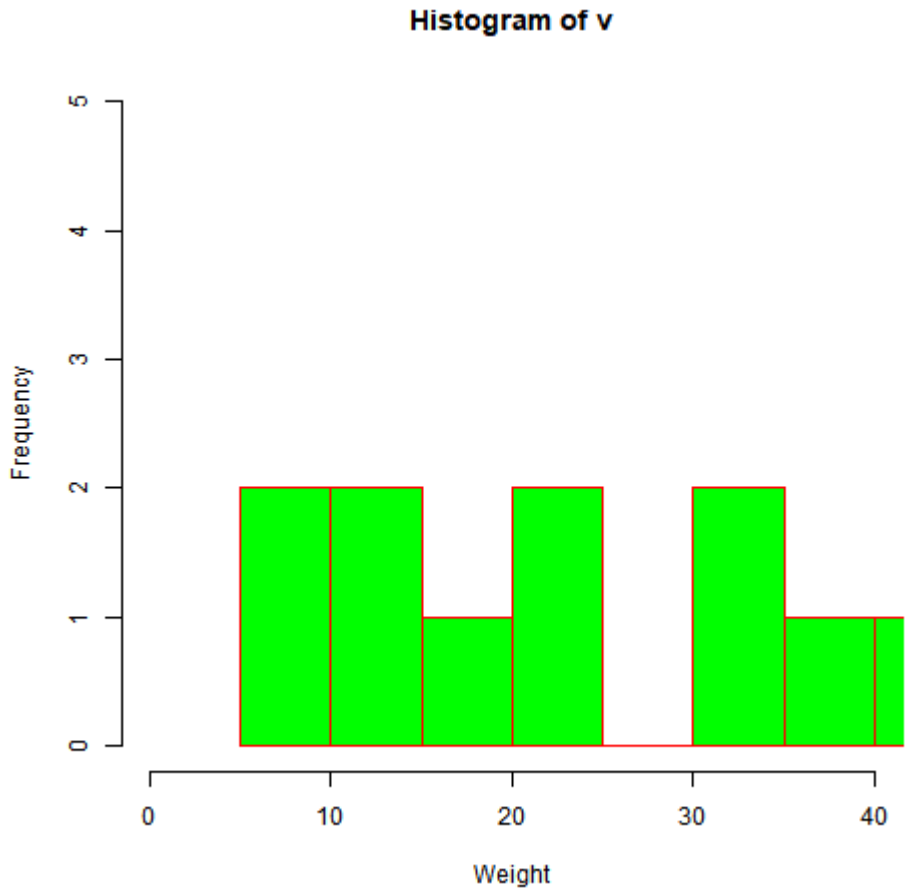
```
> # Create data for the graph.
> v <-  c(9,13,21,8,36,22,12,41,31,33,19)
>
> # Give the chart file a name.
> png(file = "histogram.png")
>
> # Create the histogram.
> hist(v,xlab = "Weight",col = "yellow",border = "blue")
>
> # Save the file.
> dev.off()
null device
          1
```



RANGE OF X AND Y VALUES

R Console

```
> # Create data for the graph.
> v <- c(9,13,21,8,36,22,12,41,31,33,19)
>
> # Give the chart file a name.
> png(file = "histogram_lim_breaks.png")
>
> # Create the histogram.
> hist(v,xlab = "Weight",col = "green",border = "red", xlim = c(0,40), ylim = c$
+    breaks = 5)
>
> # Save the file.
> dev.off()
null device
          1
```



R - LINE GRAPHS

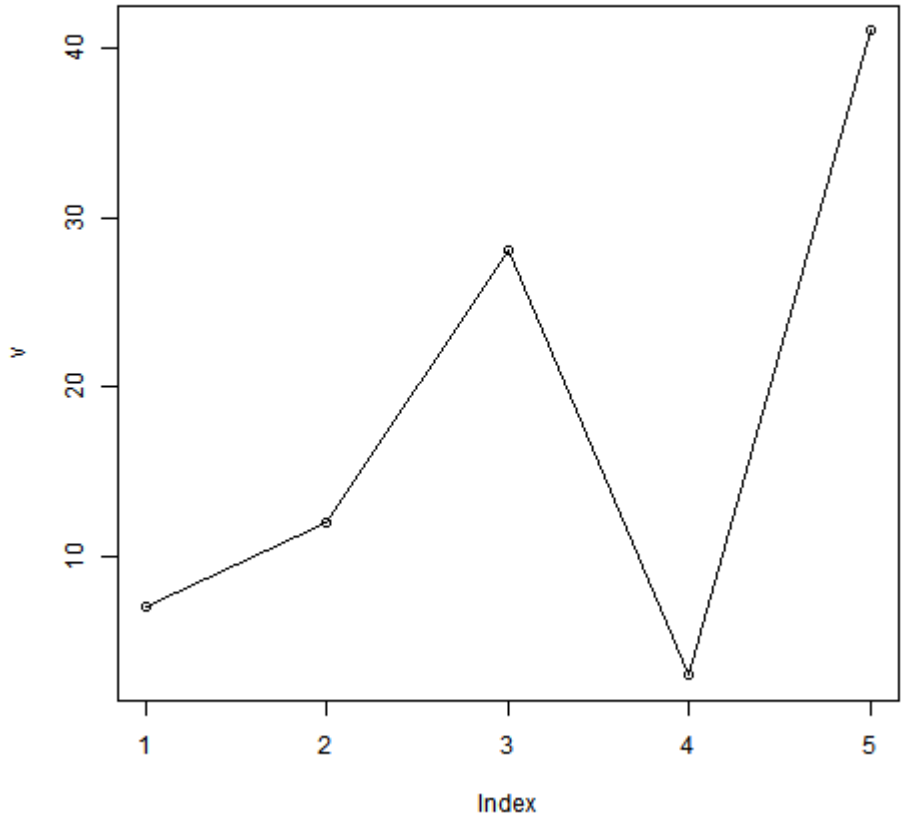
```
plot(v,type,col,xlab,ylab)
```

- **v** is a vector containing the numeric values.
- **type** takes the value "p" to draw only the points, "l" to draw only the lines and "o" to draw both points and lines.
- **xlab** is the label for x axis.
- **ylab** is the label for y axis.
- **main** is the Title of the chart.
- **col** is used to give colors to both the points and lines.

EXAMPLE - LINE GRAPHS

R Console

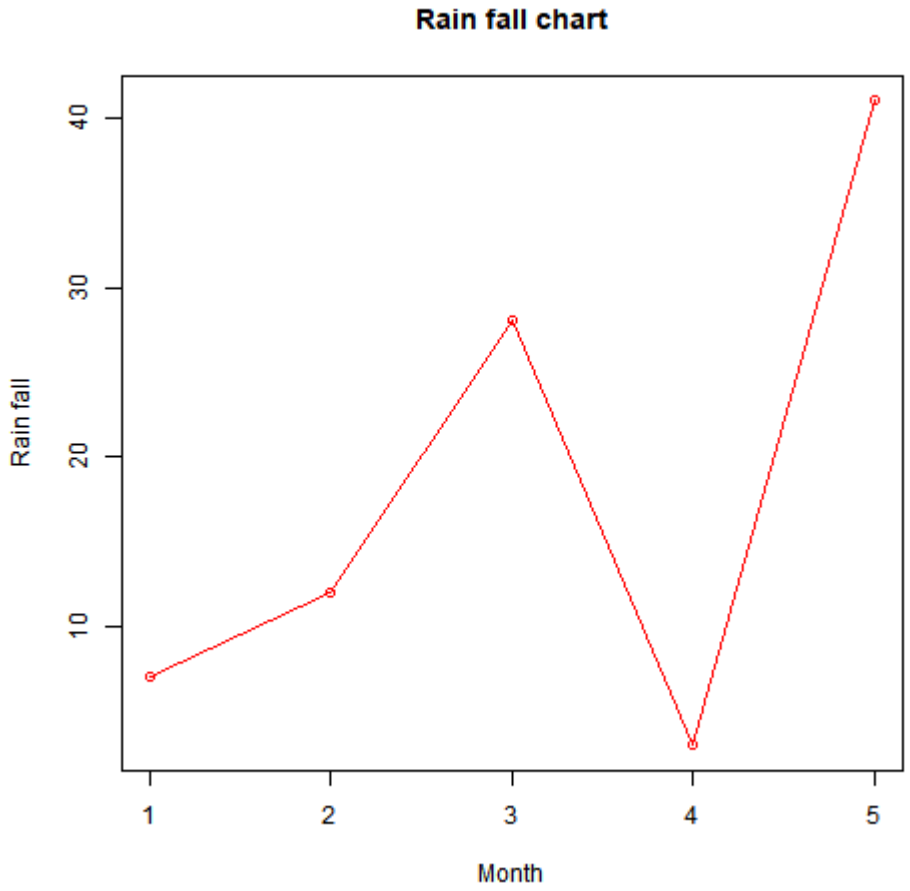
```
> # Create the data for the chart.
> v <- c(7,12,28,3,41)
>
> # Give the chart file a name.
> png(file = "line_chart.jpg")
>
> # Plot the bar chart.
> plot(v,type = "o")
>
> # Save the file.
> dev.off()
null device
          1
```



LINE CHART TITLE, COLOR AND LABELS

R Console

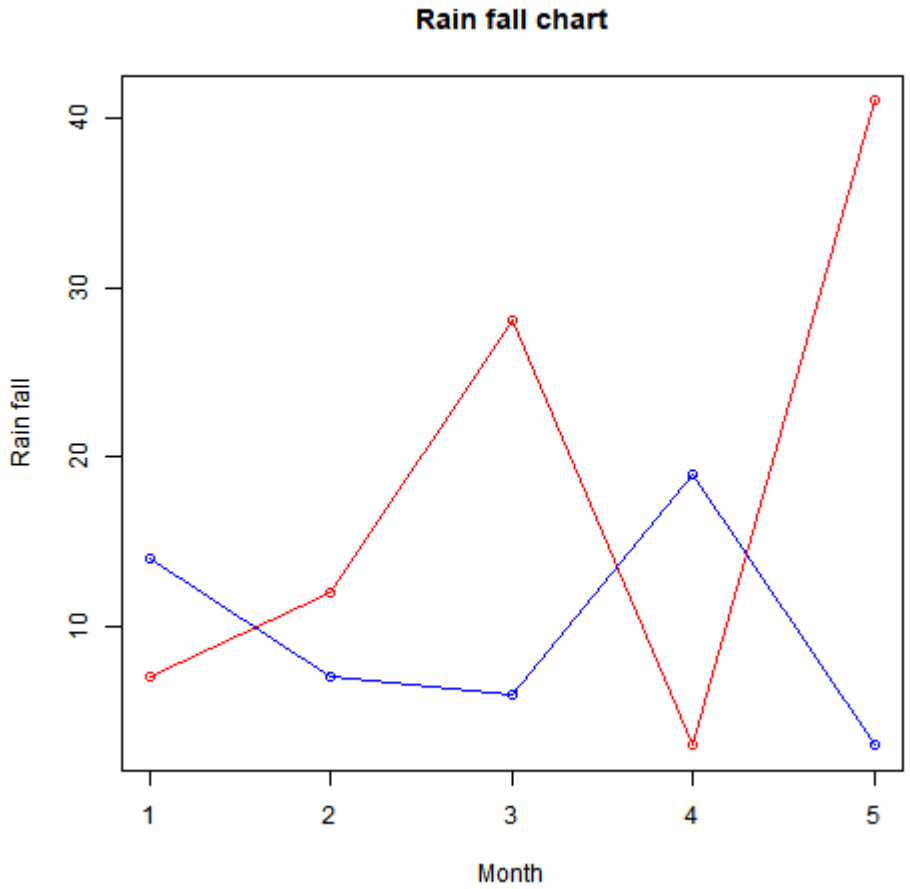
```
> # Create the data for the chart.
> v <- c(7,12,28,3,41)
>
> # Give the chart file a name.
> png(file = "line_chart_label_colored.jpg")
>
> # Plot the bar chart.
> plot(v,type = "o", col = "red", xlab = "Month", ylab = "Rain fall",
+    main = "Rain fall chart")
>
> # Save the file.
> dev.off()
null device
          1
```



MULTIPLE LINES IN A LINE CHART

R Console

```
> # Create the data for the chart.
> v <- c(7,12,28,3,41)
> t <- c(14,7,6,19,3)
>
> # Give the chart file a name.
> png(file = "line_chart_2_lines.jpg")
>
> # Plot the bar chart.
> plot(v,type = "o",col = "red", xlab = "Month", ylab = "Rain fall",
+    main = "Rain fall chart")
>
> lines(t, type = "o", col = "blue")
>
> # Save the file.
> dev.off()
null device
          1
```



R – MEAN, MEDIAN,

The basic syntax for calculating mean in R is −

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Following is the description of the parameters used −

- **x** is the input vector.
- **trim** is used to drop some observations from both end of the sorted vector.
- **na.rm** is used to remove the missing values from the input vector.

The basic syntax for calculating median in R is −

```
median(x, na.rm = FALSE)
```

Following is the description of the parameters used −

- **x** is the input vector.
- **na.rm** is used to remove the missing values from the input vector.

EXAMPLE

R Console

```
> # Create a vector.
> x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
>
> # Find Mean.
> result.mean <- mean(x)
> print(result.mean)
[1] 8.22
>
```

```
# Create the vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)

# Find the median.
median.result <- median(x)
print(median.result)
```

When we execute the above code, it produces the following result -

```
[1] 5.6
```

```
# Create a vector.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)

# Find mean.
result.mean <-  mean(x)
print(result.mean)

# Find mean dropping NA values.
result.mean <-  mean(x,na.rm = TRUE)
print(result.mean)
```

R – LINEAR REGRESSION

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.

R – LINEAR REGRESSION

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.

R – LINEAR REGRESSION

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

# Apply the lm() function.
relation <- lm(y~x)

print(relation)
```

```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   -38.4551          0.6746
```

R – LINEAR REGRESSION

Predict the weight of new persons

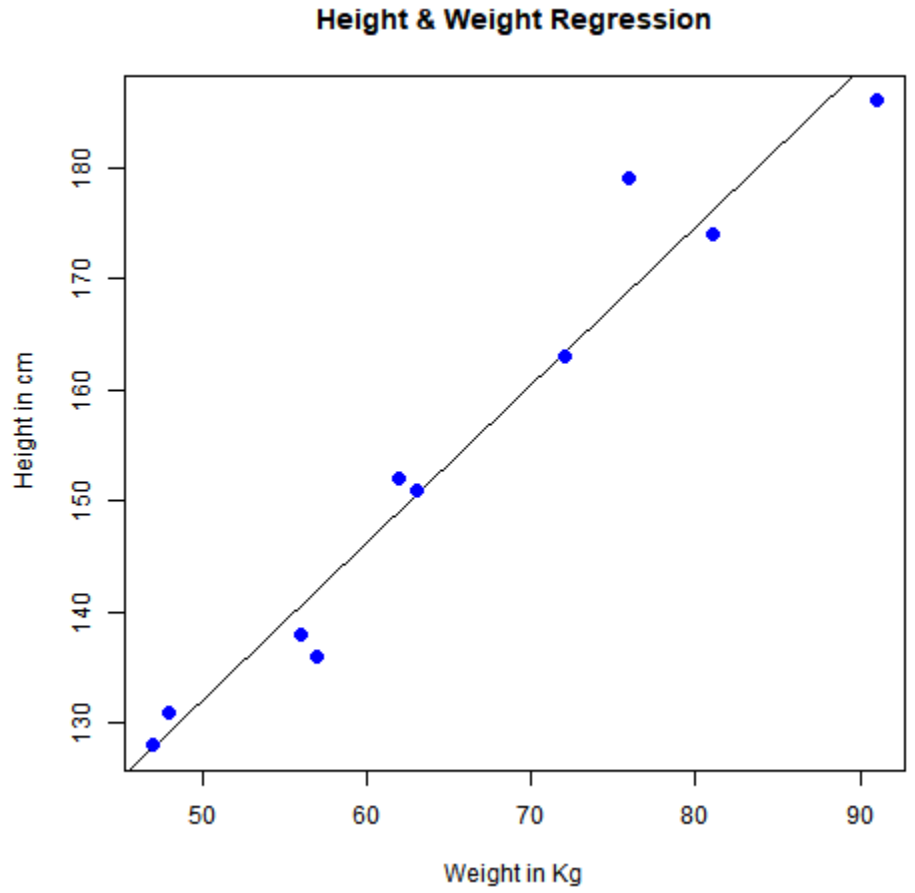
```
# The predictor vector.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

# 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

R Console

```
> # 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()
null device
          1
```



NORMAL DISTRIBUTION

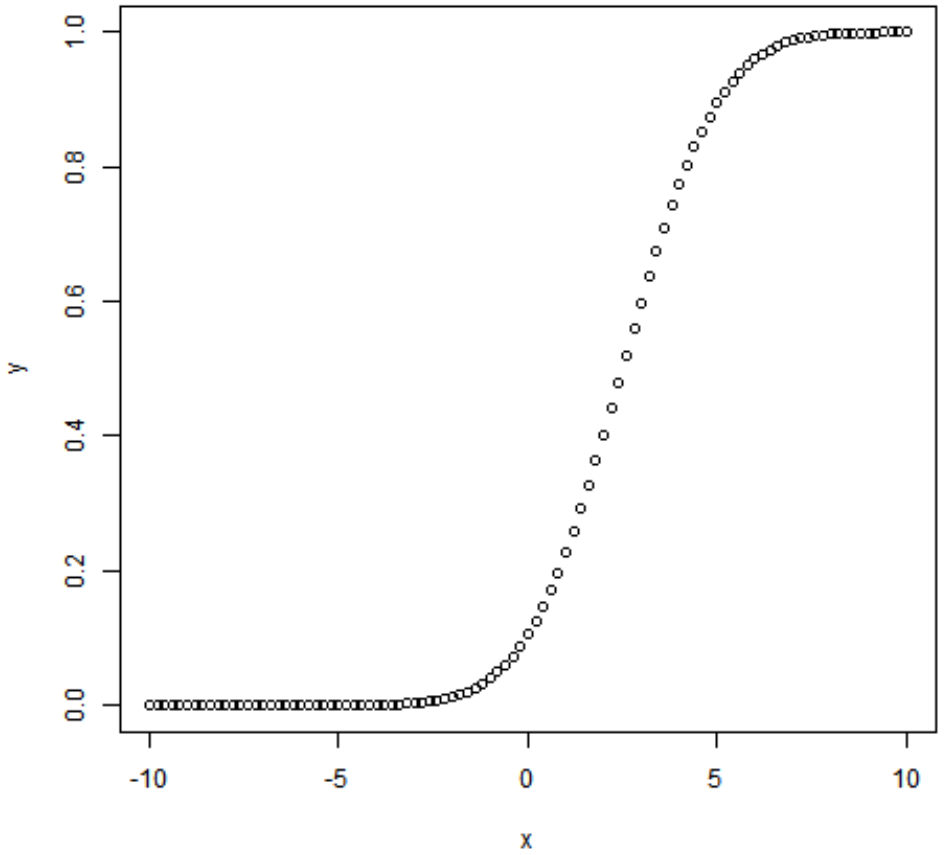
- R has four in built functions to generate normal distribution.
- dnorm(x, mean, sd)
- pnorm(x, mean, sd)
- qnorm(p, mean, sd)
- Rnorm(n, mean, sd)

- x is a vector of numbers.
- p is a vector of probabilities.
- n is number of observations(sample size).
- mean is the mean value of the sample data. It's default value is zero.
- sd is the standard deviation. It's default value is 1.

PNORM()

R Console

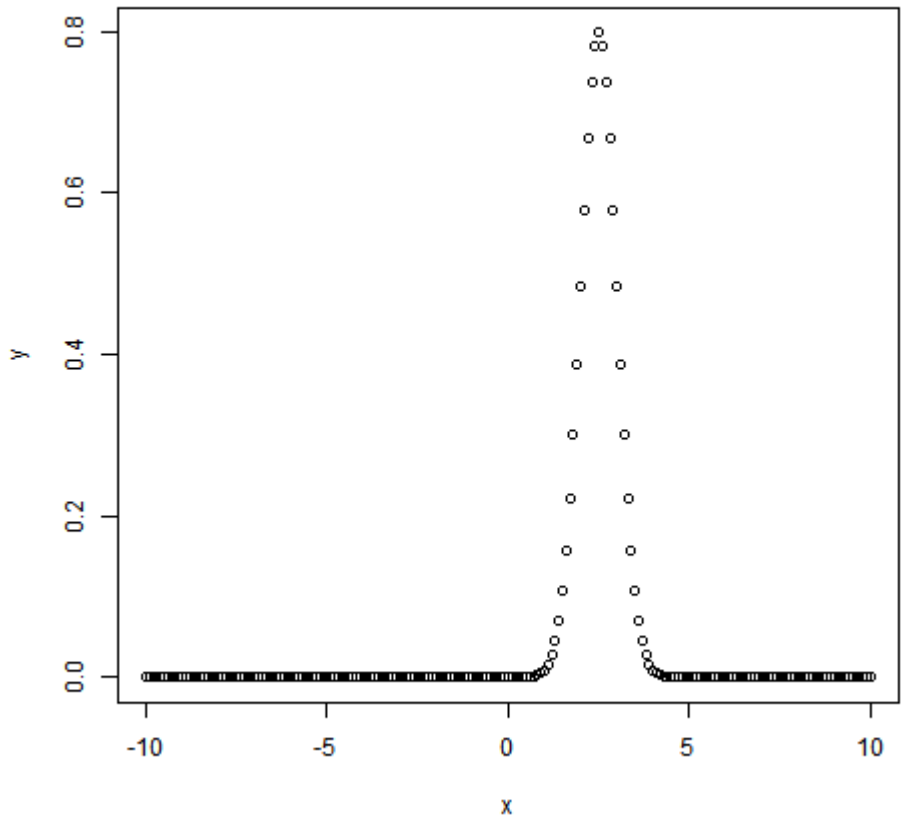
```
> # Create a sequence of numbers between -10 and 10 incrementing by 0.2.
> x <- seq(-10,10,by = .2)
>
> # Choose the mean as 2.5 and standard deviation as 2.
> y <- pnorm(x, mean = 2.5, sd = 2)
>
> # Give the chart file a name.
> png(file = "pnorm.png")
>
> # Plot the graph.
> plot(x,y)
>
> # Save the file.
> dev.off()
null device
          1
>
```



DNORM()

R Console

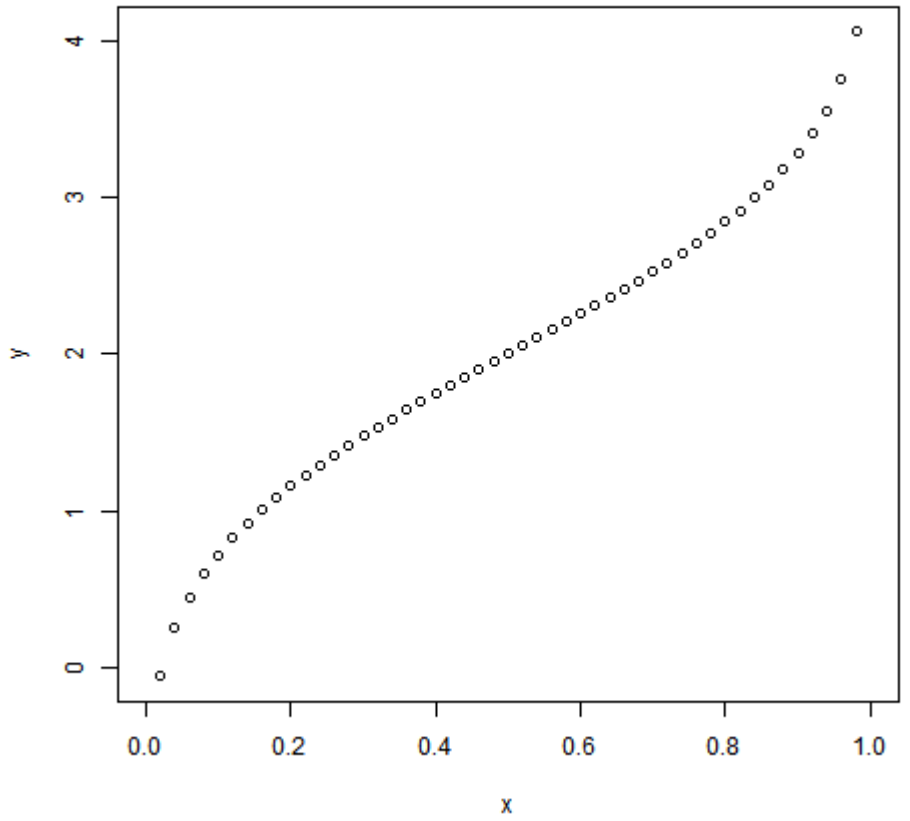
```
> # Create a sequence of numbers between -10 and 10 incrementing by 0.1.
> x <- seq(-10, 10, by = .1)
>
> # Choose the mean as 2.5 and standard deviation as 0.5.
> y <- dnorm(x, mean = 2.5, sd = 0.5)
>
> # Give the chart file a name.
> png(file = "dnorm.png")
>
> plot(x,y)
>
> # Save the file.
> dev.off()
null device
          1
```



QNORM()

R Console

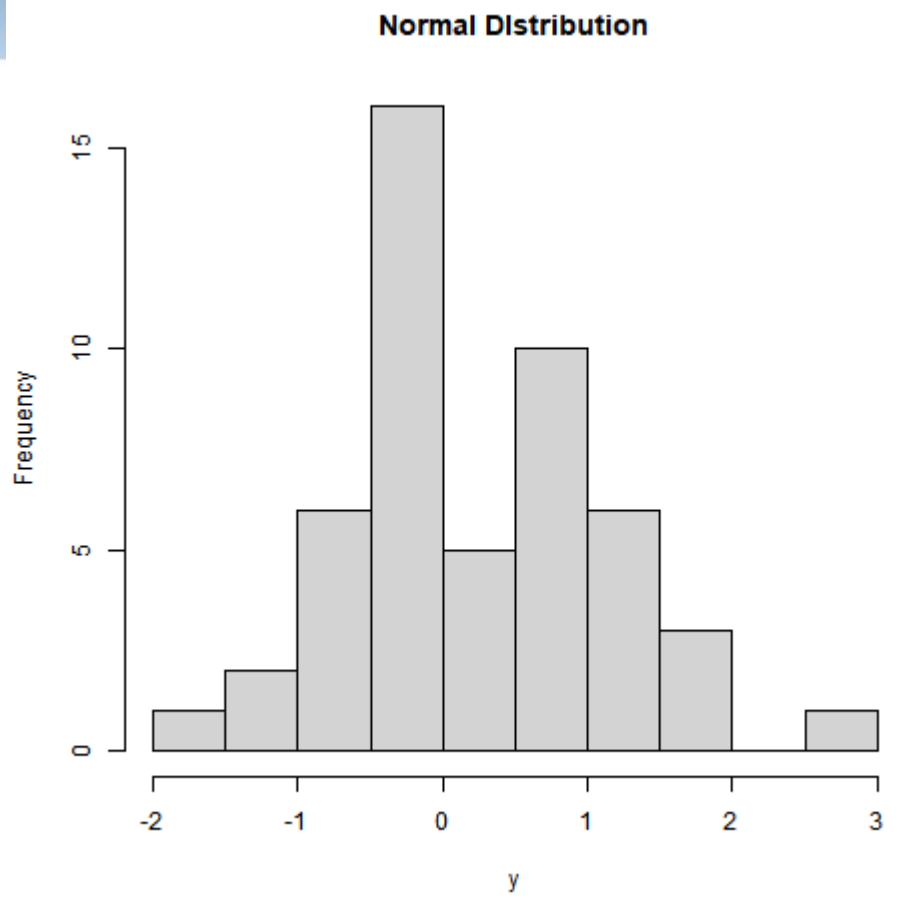
```
> # Create a sequence of probability values incrementing by 0.02.
> x <- seq(0, 1, by = 0.02)
>
> # Choose the mean as 2 and standard deviation as 3.
> y <- qnorm(x, mean = 2, sd = 1)
>
> # Give the chart file a name.
> png(file = "qnorm.png")
>
> # Plot the graph.
> plot(x,y)
>
> # Save the file.
> dev.off()
null device
          1
```



RNORM()

R Console

```
> # Create a sample of 50 numbers which are normally distributed.
> y <- rnorm(50)
>
> # Give the chart file a name.
> png(file = "rnorm.png")
>
> # Plot the histogram for this sample.
> hist(y, main = "Normal DIstribution")
>
> # Save the file.
> dev.off()
null device
          1
```



TIME SERIES ANALYSIS-SYNTAX

The basic syntax for **ts()** function in time series analysis is −

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

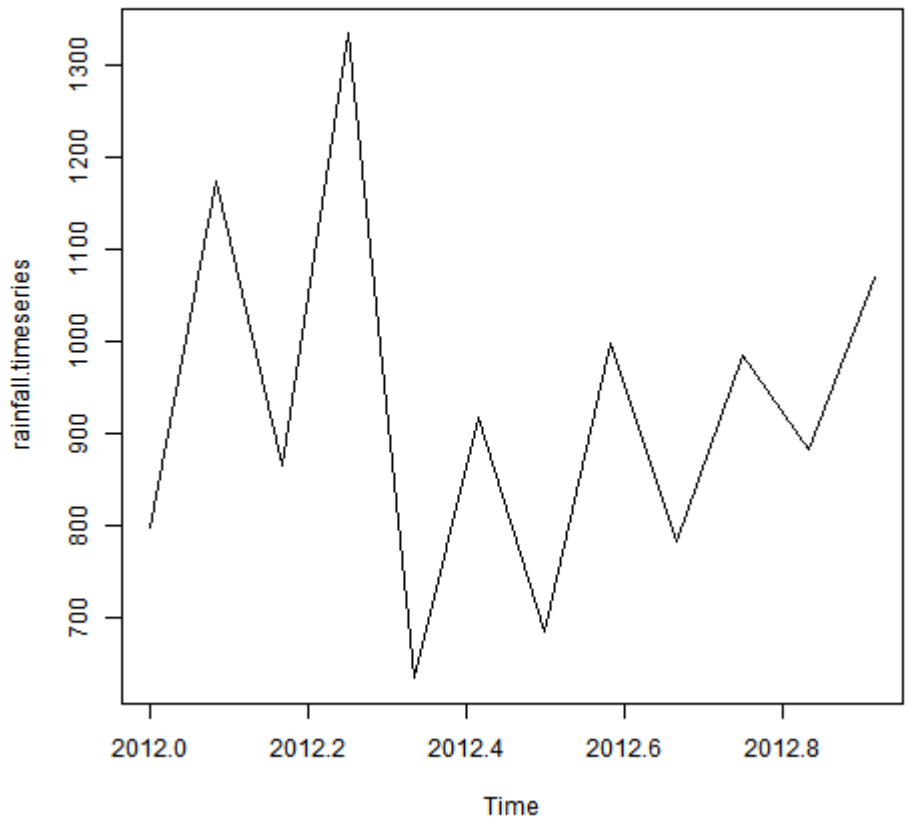
Following is the description of the parameters used −

- **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.

TIME SERIES ANALYSIS-EXAMPLE

R Console

```
> # 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$
>
> # Convert it to a time series object.
> rainfall.timeseries <- ts(rainfall,start = c(2012,1),frequency = 12)
>
> # Print the timeseries data.
> print(rainfall.timeseries)
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
2012  799.0 1174.8  865.1 1334.6  635.4  918.5  685.5  998.6  784.2  985.0
        Nov    Dec
2012  882.8 1071.0
>
> # 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()
null device
          1
```

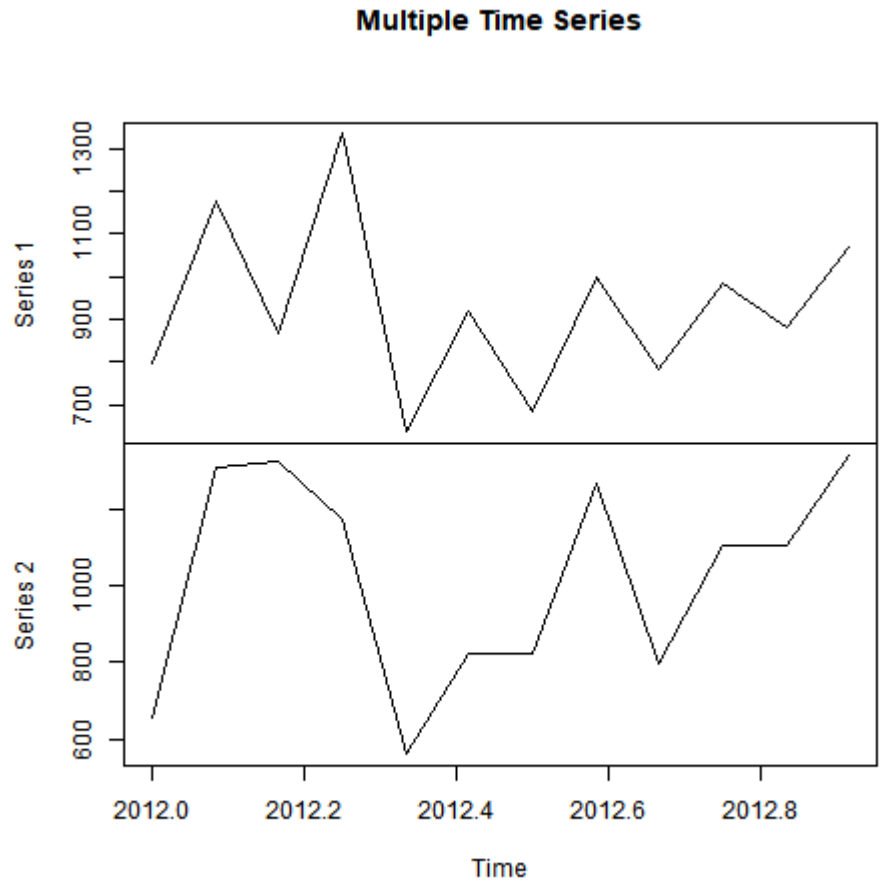


TIME SERIES ANALYSIS-DIFFERENT TIME INTERVALS

- The value of the **frequency** parameter in the ts() function decides the time intervals at which the data points are measured.
- frequency = 12 pegs the data points for every month of a year.
- frequency = 4 pegs the data points for every quarter of a year.
- frequency = 6 pegs the data points for every 10 minutes of an hour.
- frequency = 24*6 pegs the data points for every 10 minutes of a day.

MULTIPLE TIME SERIES

```
> # Get the data points in form of a R vector.
> rainfall1 <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)
> 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)
         Series 1 Series 2
Jan 2012    799.0    655.0
Feb 2012   1174.8   1306.9
Mar 2012    865.1   1323.4
Apr 2012   1334.6   1172.2
May 2012    635.4    562.2
Jun 2012    918.5    824.0
Jul 2012    685.5    822.4
Aug 2012    998.6   1265.5
Sep 2012    784.2    799.6
Oct 2012    985.0   1105.6
Nov 2012    882.8   1106.7
Dec 2012   1071.0   1337.8
>
> # 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()
```



DECISION TREE-SYNTAX

- The R package **"party"** is used to create decision trees. → install.packages("party")

Syntax

The basic syntax for creating a decision tree in R is −

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

Following is the description of the parameters used −

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

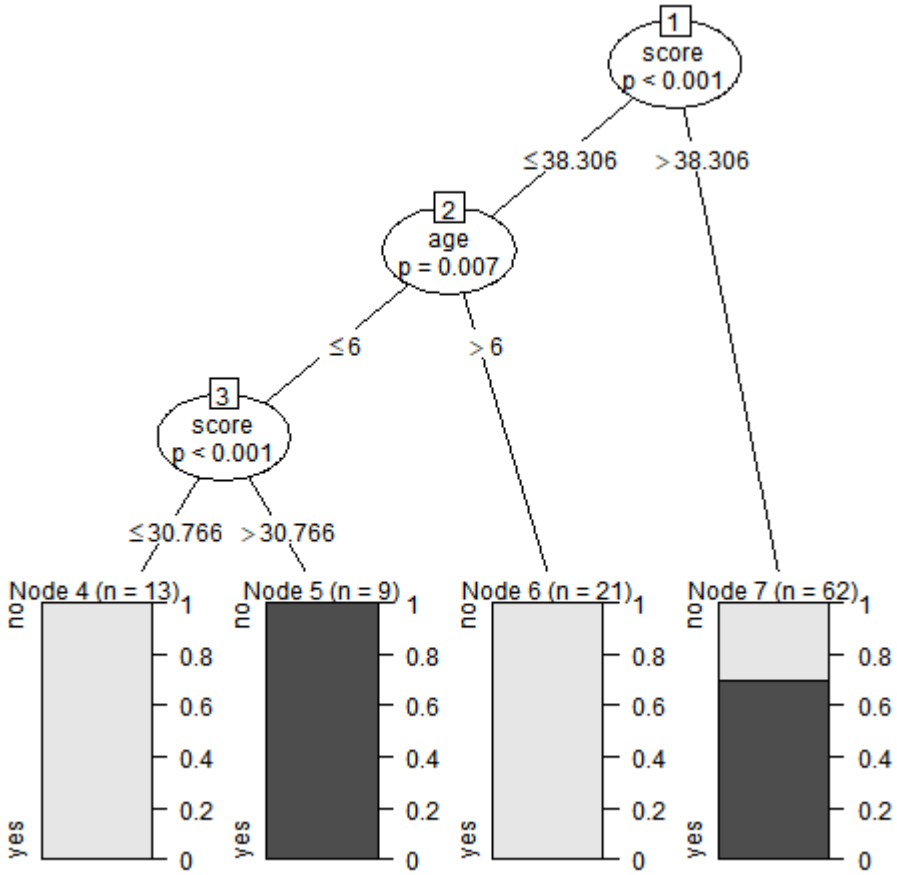
INPUT DATA

```
> library(party)
Zorunlu paket yükleniyor: grid
Zorunlu paket yükleniyor: mvtnorm
Zorunlu paket yükleniyor: modeltools
Zorunlu paket yükleniyor: stats4
Zorunlu paket yükleniyor: strucchange
Zorunlu paket yükleniyor: zoo

Attaching package: 'zoo'

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

    as.Date, as.Date.numeric

Zorunlu paket yükleniyor: sandwich
```

```
> print(head(readingSkills))
  nativeSpeaker age shoeSize    score
1           yes   5 24.83189 32.29385
2           yes   6 25.95238 36.63105
3            no  11 30.42170 49.60593
4           yes   7 28.66450 40.28456
5           yes  11 31.88207 55.46085
6           yes  10 30.07843 52.83124
>
```

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()
null device
          1
```



SURVIVAL ANALYSIS

- The R package named **survival** is used to carry out survival analysis.
- We use the function survfit() to create a plot for the analysis.

```
Surv(time,event)
survfit(formula)
```

Following is the description of the parameters used −

- **time** is the follow up time until the event occurs.
- **event** indicates the status of occurrence of the expected event.
- **formula** is the relationship between the predictor variables.

SURVIVAL ANALYSIS

```
> library("survival")
> print(head(pbc))
  id time status trt      age sex ascites hepato spiders edema bili chol albumin copper alk.phos    ast
1  1  400      2   1 58.76523   f       1      1       1   1.0 14.5  261    2.60    156   1718.0 137.95
2  2 4500      0   1 56.44627   f       0      1       1   0.0  1.1  302    4.14     54   7394.8 113.52
3  3 1012      2   1 70.07255   m       0      0       0   0.5  1.4  176    3.48    210    516.0  96.10
4  4 1925      2   1 54.74059   f       0      1       1   0.5  1.8  244    2.54     64   6121.8  60.63
5  5 1504      1   2 38.10541   f       0      1       1   0.0  3.4  279    3.53    143    671.0 113.15
6  6 2503      2   2 66.25873   f       0      1       0   0.0  0.8  248    3.98     50    944.0  93.00
  trig platelet protime stage
1  172      190    12.2     4
2   88      221    10.6     3
3   55      151    12.0     4
4   92      183    10.3     4
5   72      136    10.9     3
6   63       NA    11.0     3
```

SURVIVAL ANALYSIS

R Console

```
> # Create the survival object.
> survfit(Surv(pbc$time,pbc$status == 2)~1)
Call: survfit(formula = Surv(pbc$time, pbc$status == 2) ~ 1)

      n events median 0.95LCL 0.95UCL
[1,] 418    161   3395    3090    3853
>
> # Give the chart file a name.
> png(file = "survival.png")
>
> # Plot the graph.
> plot(survfit(Surv(pbc$time,pbc$status == 2)~1))
>
> # Save the file.
> dev.off()
null device
          1
```