Normal Distribution

In a random collection of data from independent sources, it is generally observed that the distribution of data is normal. Which means, on plotting a graph with the value of the variable in the horizontal axis and the count of the values in the vertical axis we get a bell shape curve. The center of the curve represents the mean of the data set. In the graph, fifty percent of values lie to the left of the mean and the other fifty percent lie to the right of the graph. This is referred as normal distribution in statistics.

R has four in built functions to generate normal distribution. They are described below.

dnorm(x, mean, sd)

pnorm(x, mean, sd)

qnorm(p, mean, sd)

rnorm(n, mean, sd)

Following is the description of the parameters used in above functions −

* **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.

dnorm()

This function gives height of the probability distribution at each point for a given mean and standard deviation.

# 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.

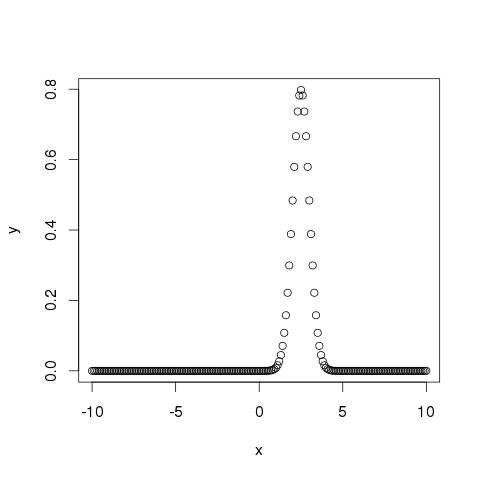
png(file = "dnorm.png")

plot(x,y)

# Save the file.

dev.off()

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



pnorm()

This function gives the probability of a normally distributed random number to be less that the value of a given number. It is also called "Cumulative Distribution Function".

# 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")

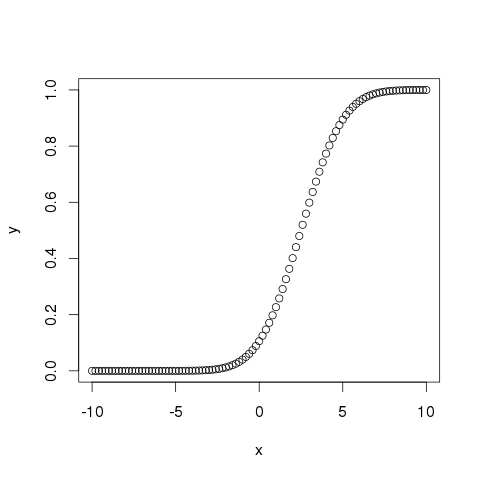
# Plot the graph.

plot(x,y)

# Save the file.

dev.off()

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



qnorm()

This function takes the probability value and gives a number whose cumulative value matches the probability value.

# 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")

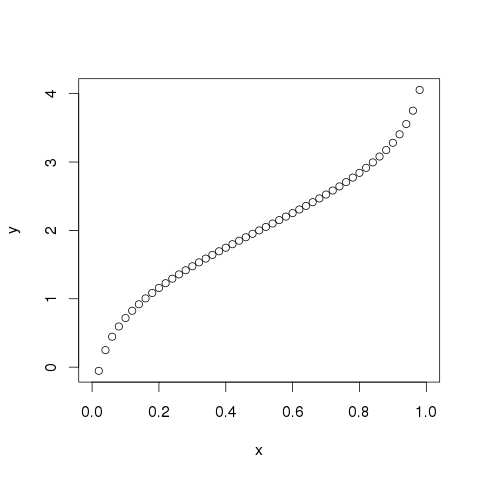
# Plot the graph.

plot(x,y)

# Save the file.

dev.off()

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



rnorm()

This function is used to generate random numbers whose distribution is normal. It takes the sample size as input and generates that many random numbers. We draw a histogram to show the distribution of the generated numbers.

# Create a sample of 50 numbers which are normally distributed.

y <- rnorm(50)

# Give the chart file a name.

png(file = "rnorm.png")

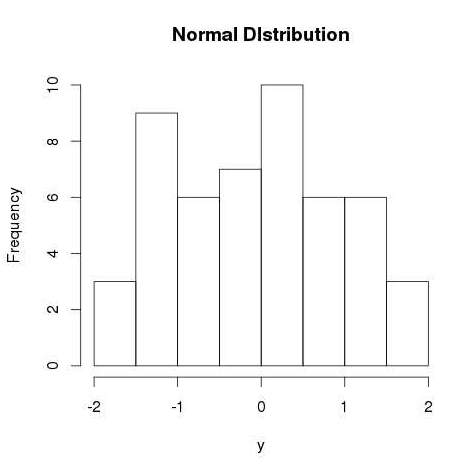
# Plot the histogram for this sample.

hist(y, main = "Normal DIstribution")

# Save the file.

dev.off()

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



Binomial Distribution

The binomial distribution model deals with finding the probability of success of an event which has only two possible outcomes in a series of experiments. For example, tossing of a coin always gives a head or a tail. The probability of finding exactly 3 heads in tossing a coin repeatedly for 10 times is estimated during the binomial distribution.

R has four in-built functions to generate binomial distribution. They are described below.

dbinom(x, size, prob)

pbinom(x, size, prob)

qbinom(p, size, prob)

rbinom(n, size, prob)

Following is the description of the parameters used −

* **x** is a vector of numbers.
* **p** is a vector of probabilities.
* **n** is number of observations.
* **size** is the number of trials.
* **prob** is the probability of success of each trial.

dbinom()

This function gives the probability density distribution at each point.

# Create a sample of 50 numbers which are incremented by 1.

x <- seq(0,50,by = 1)

# Create the binomial distribution.

y <- dbinom(x,50,0.5)

# Give the chart file a name.

png(file = "dbinom.png")

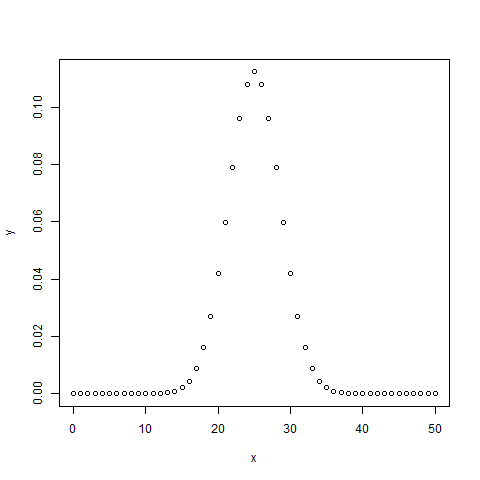
# Plot the graph for this sample.

plot(x,y)

# Save the file.

dev.off()

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



pbinom()

This function gives the cumulative probability of an event. It is a single value representing the probability.

# Probability of getting 26 or less heads from a 51 tosses of a coin.

x <- pbinom(26,51,0.5)

print(x)

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

[1] 0.610116

qbinom()

This function takes the probability value and gives a number whose cumulative value matches the probability value.

# How many heads will have a probability of 0.25 will come out when a coin is tossed 51 times.

x <- qbinom(0.25,51,1/2)

print(x)

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

[1] 23

rbinom()

This function generates required number of random values of given probability from a given sample.

# Find 8 random values from a sample of 150 with probability of 0.4.

x <- rbinom(8,150,.4)

print(x)

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

[1] 58 61 59 66 55 60 61 67

Survival Analysis

Survival analysis deals with predicting the time when a specific event is going to occur. It is also known as failure time analysis or analysis of time to death. For example predicting the number of days a person with cancer will survive or predicting the time when a mechanical system is going to fail.

The R package named **survival** is used to carry out survival analysis. This package contains the function **Surv()** which takes the input data as a R formula and creates a survival object among the chosen variables for analysis. Then we use the function **survfit()** to create a plot for the analysis.

Install Package

install.packages("survival")

Syntax

The basic syntax for creating survival analysis in R is −

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.

Example

We will consider the data set named "pbc" present in the survival packages installed above. It describes the survival data points about people affected with primary biliary cirrhosis (PBC) of the liver. Among the many columns present in the data set we are primarily concerned with the fields "time" and "status". Time represents the number of days between registration of the patient and earlier of the event between the patient receiving a liver transplant or death of the patient.

# Load the library.

library("survival")

# Print first few rows.

print(head(pbc))

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

id time status trt age sex ascites hepato spiders edema bili chol

1 1 400 2 1 58.76523 f 1 1 1 1.0 14.5 261

2 2 4500 0 1 56.44627 f 0 1 1 0.0 1.1 302

3 3 1012 2 1 70.07255 m 0 0 0 0.5 1.4 176

4 4 1925 2 1 54.74059 f 0 1 1 0.5 1.8 244

5 5 1504 1 2 38.10541 f 0 1 1 0.0 3.4 279

6 6 2503 2 2 66.25873 f 0 1 0 0.0 0.8 248

albumin copper alk.phos ast trig platelet protime stage

1 2.60 156 1718.0 137.95 172 190 12.2 4

2 4.14 54 7394.8 113.52 88 221 10.6 3

3 3.48 210 516.0 96.10 55 151 12.0 4

4 2.54 64 6121.8 60.63 92 183 10.3 4

5 3.53 143 671.0 113.15 72 136 10.9 3

6 3.98 50 944.0 93.00 63 NA 11.0 3

From the above data we are considering time and status for our analysis.

Applying Surv() and survfit() Function

Now we proceed to apply the **Surv()** function to the above data set and create a plot that will show the trend.

# Load the library.

library("survival")

# Create the survival object.

survfit(Surv(pbc$time,pbc$status == 2)~1)

# 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.

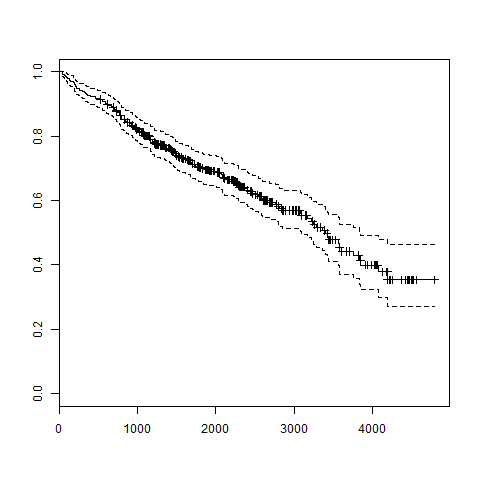
dev.off()

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

Call: survfit(formula = Surv(pbc$time, pbc$status == 2) ~ 1)

n events median 0.95LCL 0.95UCL

418 161 3395 3090 3853



The trend in the above graph helps us predicting the probability of survival at the end of a certain number of days.