Fitting Distributions with R

Debopriya Ghosh

2019-09-27

Introduction

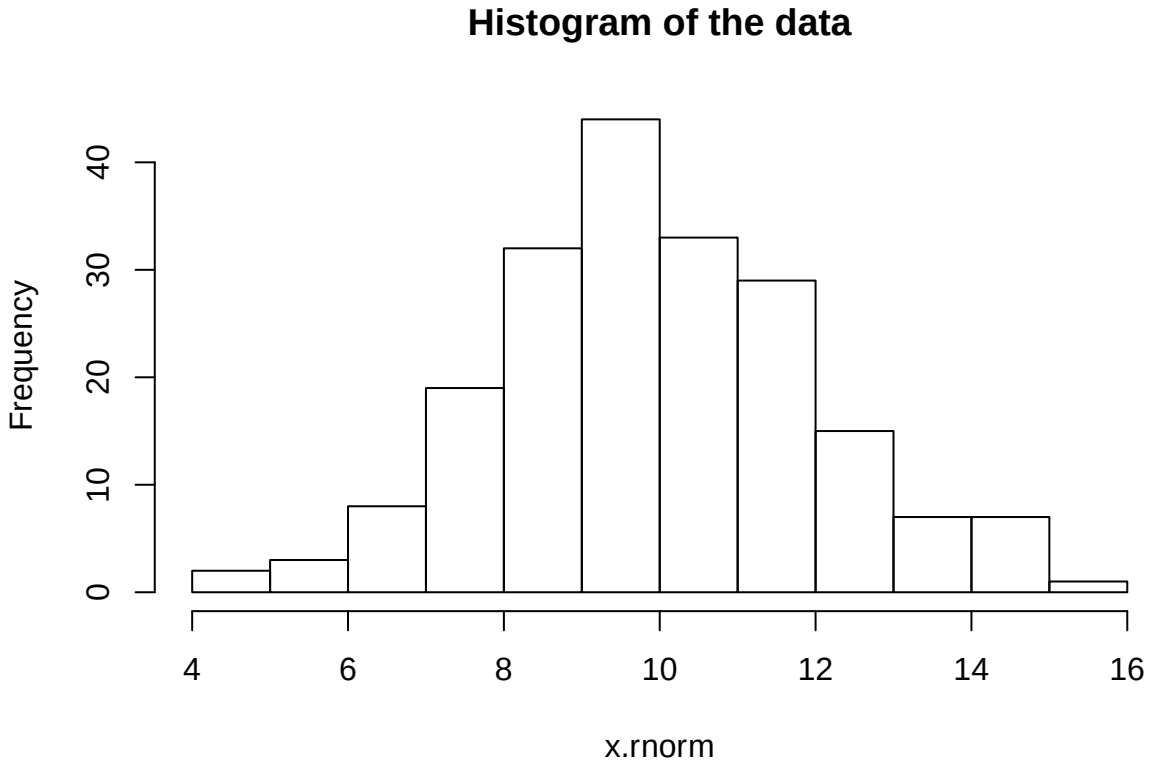
- Fitting distributions consists in finding a mathematical function which represents in a good way a statistical variable
- Suppose, you have some observations of quantitative character $x_1, x_2, ..., x_n$ and you wish to test if those observations, being a sample of an unknown population, belong from a population with probability density function (pdf) $f(x, \theta)$, where θ is a vector of parameters to estimate with available data
- The four steps in fitting distributions:
  - Model/function choice
  - Estimate parameters
  - Evaluate quality of fit
  - Goodness of statistical fits

Exploratory Data Analysis

EDA is the first step. Getting descriptive statistics (mean, standard deviation, skewness, kurtosis, etc.) and using graphical techniques (histograms, density plots, etcc.) which can suggest the kind of pdf to use to fit the model

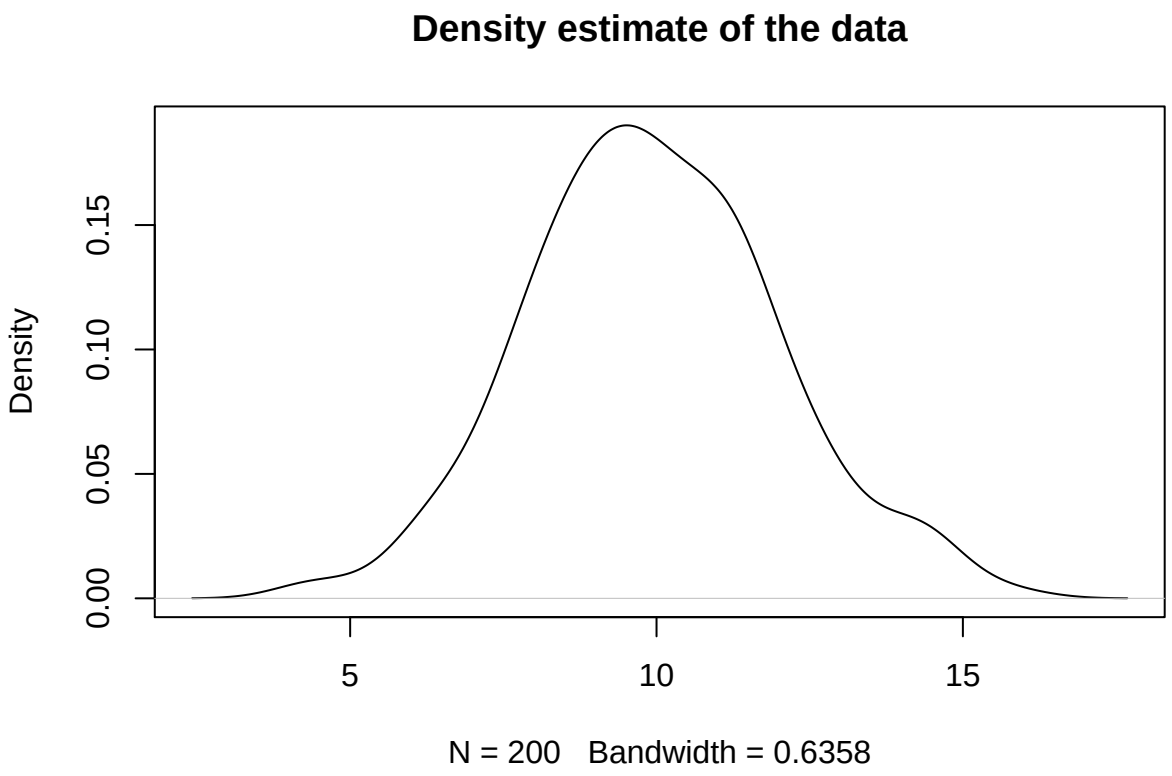
Let us obtain a sample of size 200 belonging from a normal distribution with mean 10 and standard deviation 2. We will draw the histogram of the data

```
x.rnorm = rnorm(n = 200, m = 10, sd = 2)
hist(x.rnorm, breaks = 12, main = "Histogram of the data")
```



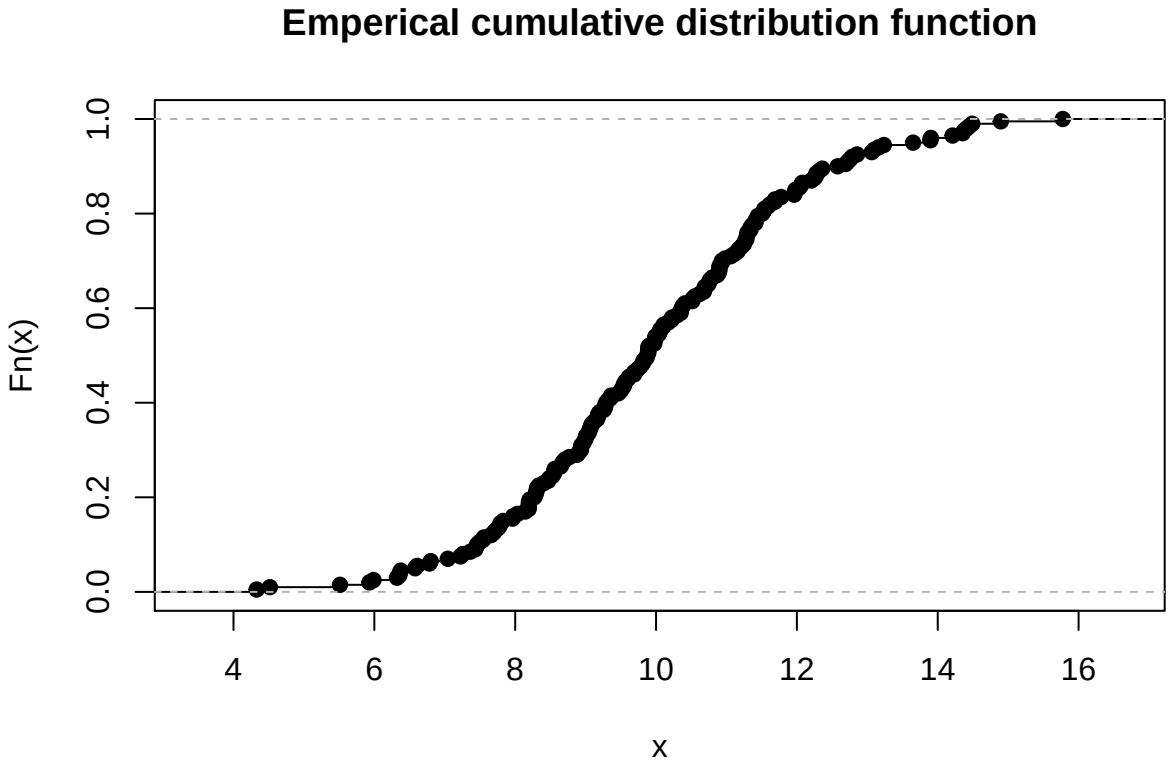
- Histograms provide insight on skewness, behavior of tails, presence of multi-modal behavior, and data outliers.
- Histograms can be compared to the shape associated with standard analytic distributions
- Next plot the density

```
plot(density(x.rnorm), main = "Density estimate of the data")
```



- Plot the cumulative density function

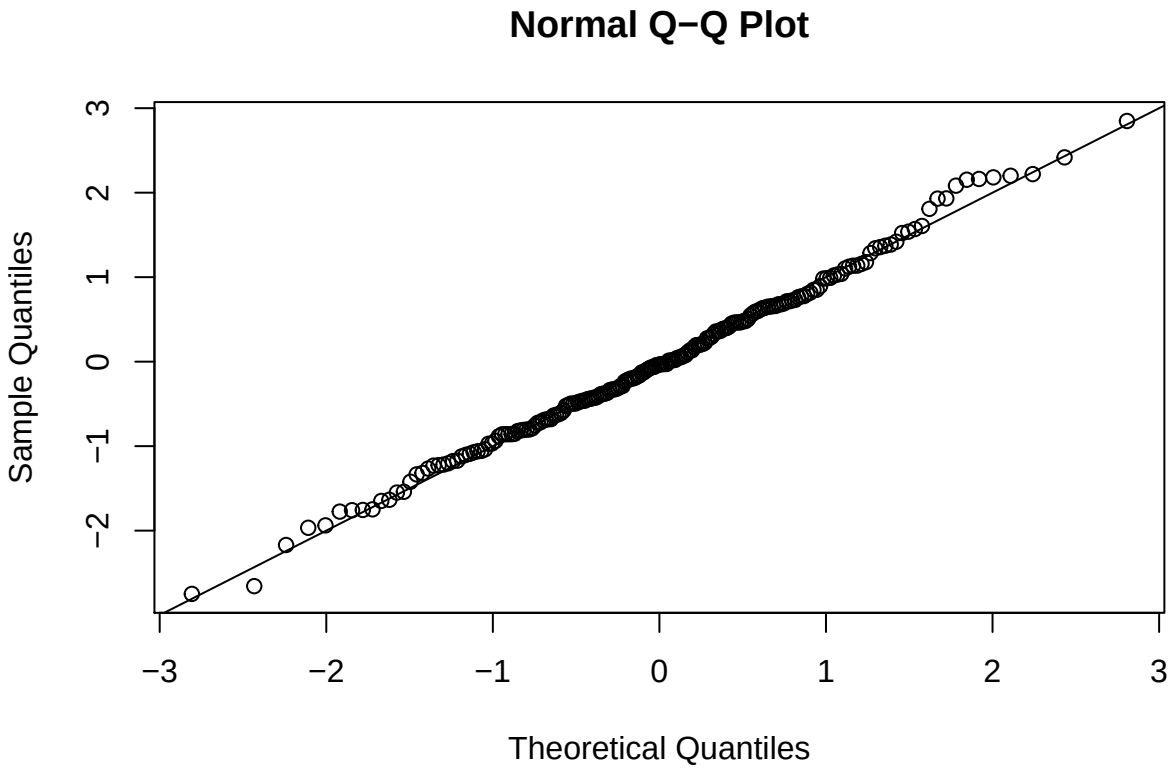
```
plot(ecdf(x.rnorm), main = "Emperical cumulative distribution function")
```



- Quatile-Quantile plot is a scatter plot comparing the fitted and emperical distributions in terms of dimensional values of the variable (emperical quantiles)

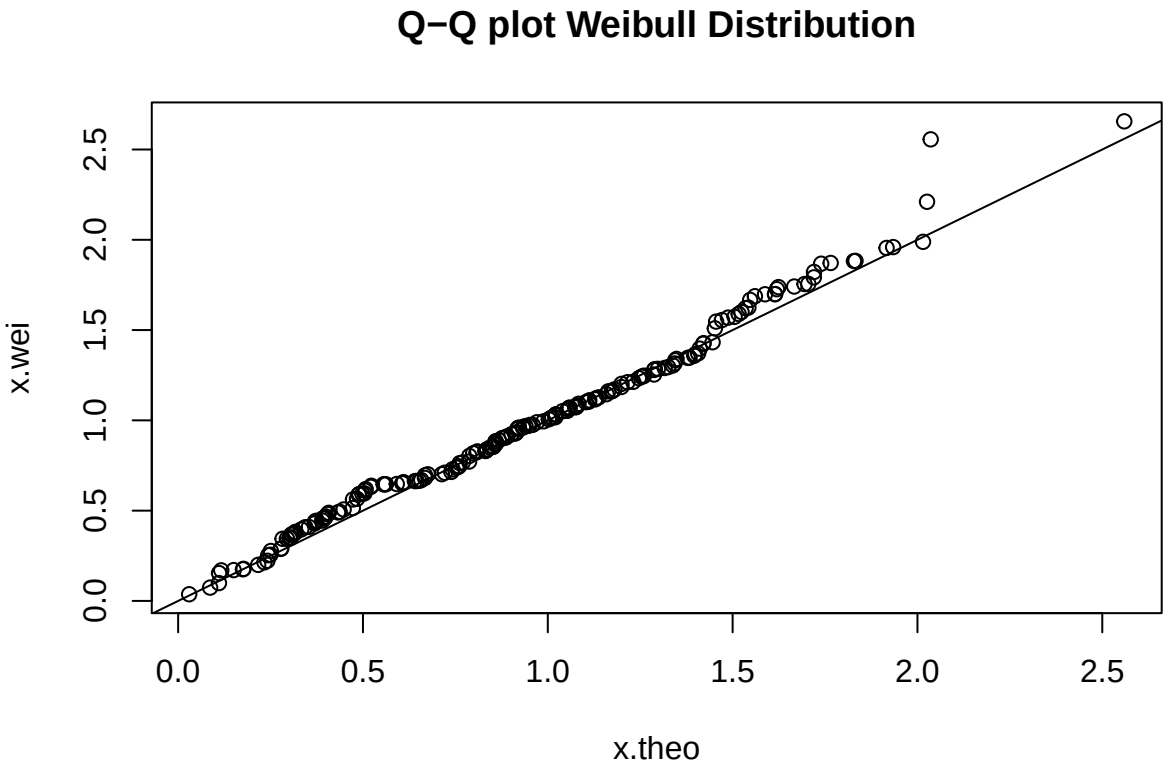
- Graphical technique for determining if a data come from a known population
- In this plot, on y-axis we have emperical quantiles and on the x-axis we have the ones from the theoretical model

```
z.norm = (x.rnorm - mean(x.rnorm))/sd(x.rnorm) # standardize data
qqnorm(z.norm) #  drawing q-q plot
abline(0,1)
```



- 45-degree reference line is plotted. If the emperical data come from the population with the chosen distribution, the points should fall approximately along the reference line
- Greater the departure from the reference line, greater is the evidence for conclusion that the data set have come from a population with a different distribution
- Lets see the q-q plot for data differing from a normal distribution (i.e., data belong to weibull distribution)

```
# weibull population with known parameters scale = 1.1 and shape = 2.1
x.wei = rweibull(n = 200, shape = 2.1, scale = 1.1)
# theoretical quantile from weibull population with known parameters scale = 1 and shape = 2
x.theo = rweibull(n = 200, shape = 2, scale = 1)
qqplot(x.theo,x.wei, main = "Q-Q plot Weibull Distribution")
abline (0,1)
```



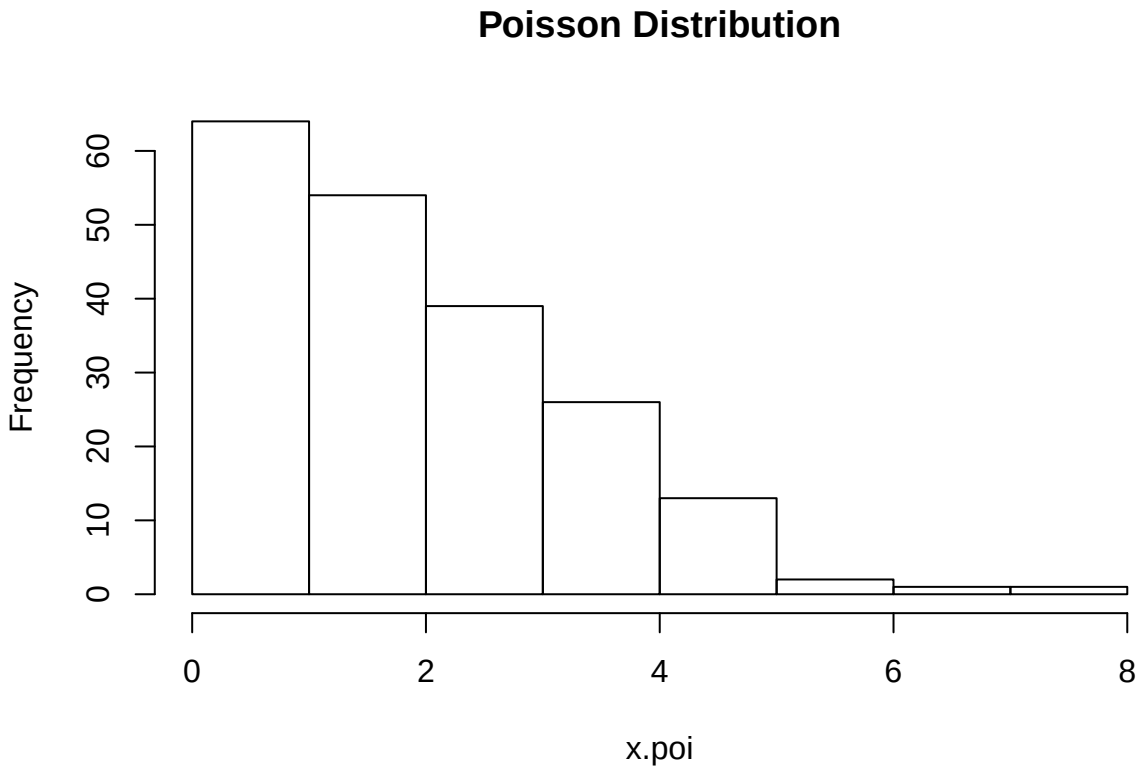
Model Choice

- First step in fitting distributions consists of choosing a mathematical model/function to represent the data
- Sometimes the type of model can be argued by some hypothesis concerning the nature of the data
- Histograms and other graphical techniques help in this step

Discrete Data Distributions

Poisson Distribution

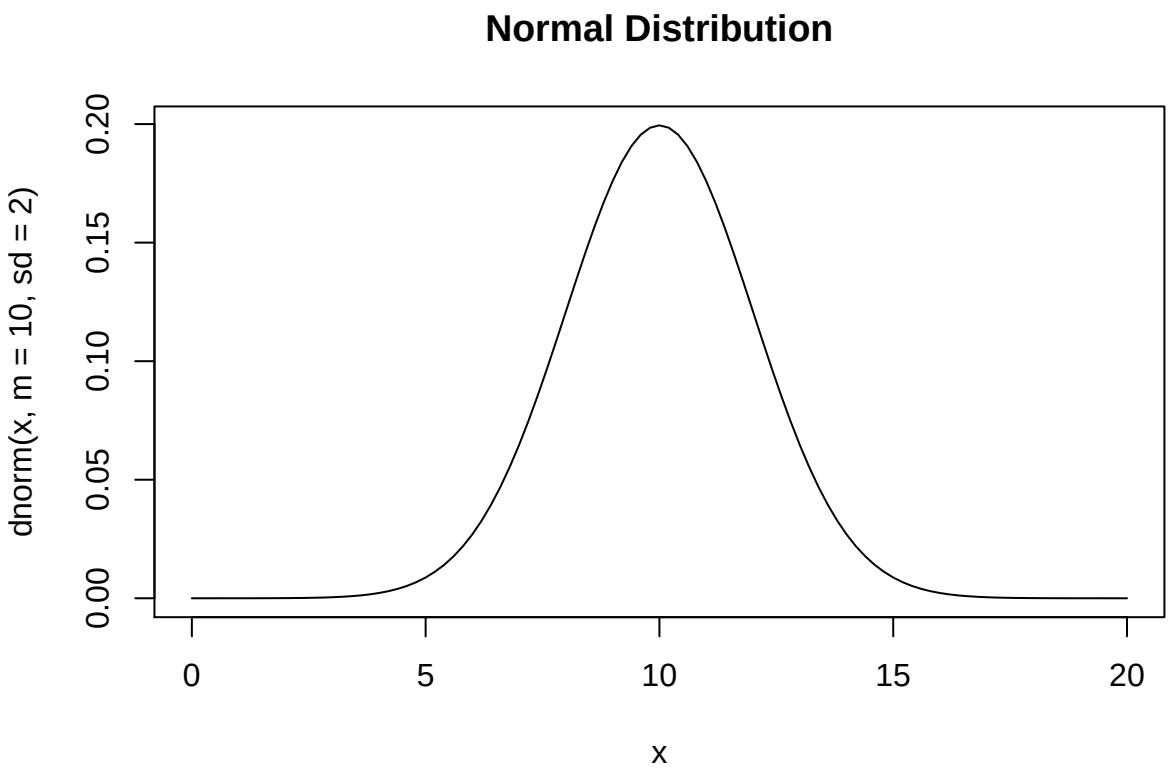
```
x.poi= rpois(n = 200, lambda = 2.5)
hist(x.poi, main = "Poisson Distribution")
```



Continuous Data Distributions

Normal (Gaussian) Distributions

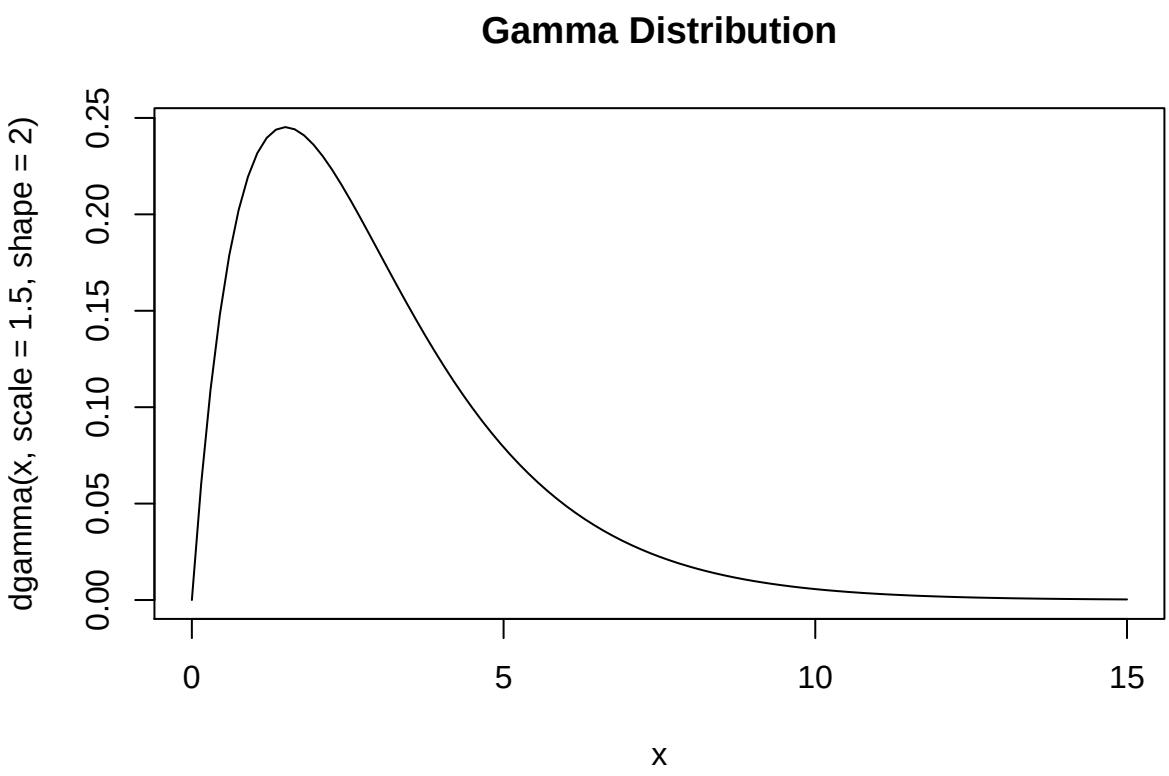
```
curve(dnorm(x,m=10,sd = 2), from = 0, to = 20, main = "Normal Distribution")
```



###

Gamma Distribution

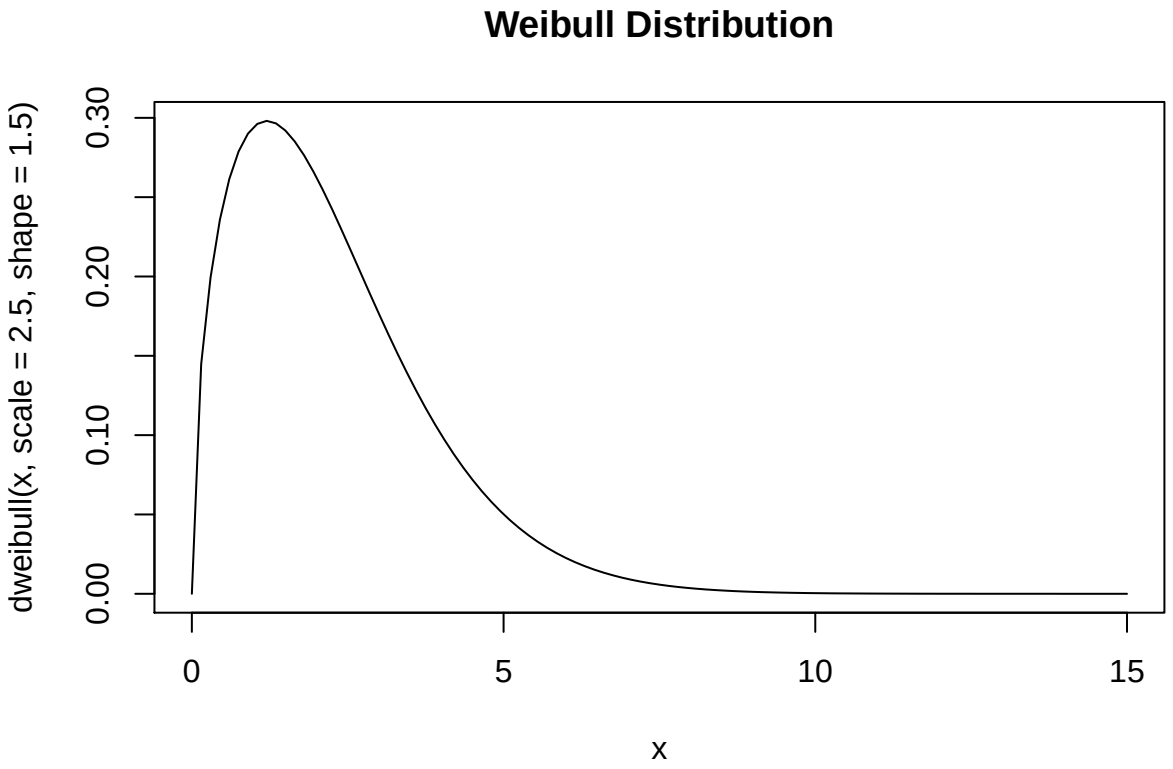
```
curve(dgamma(x, scale = 1.5, shape = 2), from = 0, to = 15, main = "Gamma Distribution")
```



###

Weibull Distribution

```r
curve(dweibull(x, scale = 2.5, shape = 1.5), from = 0, to = 15, main = "Weibull Distribution")
```



Compute Skewness and Kurtosis

```r
library(fBasics)
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

```r
skewness(x.rnorm)# skewness of normal distribution
```

```
## [1] 0.1404685
## attr(,"method")
## [1] "moment"
```

```r
kurtosis(x.rnorm)# kurtosis of normal distribution
```

```
## [1] 0.04210433
## attr(,"method")
## [1] "excess"
```

```r
skewness(x.wei)# skewness of Weibull distribution
```

```
## [1] 0.5536081
## attr(,"method")
## [1] "moment"
```

```r
kurtosis(x.wei)# kurtosis of Weibull distribution
```

```
## [1] 0.1660218
```

```
## attr(,"method")
## [1] "excess"
```

Parameter Estimate

- After choosing a model that can mathematically represent the data, we have to estimate the parameters of such model
- There are several methods –
  - Analogic
  - Moments
  - Maximum Likelihood
- Analogic method estimate model parameters by applying the same function to emperical data

```
mean.hat = mean(x.rnorm)
mean.hat
```

```
## [1] 9.950639
```

- Method of moments is a technique for constructing estimators of the paramters that is based on matching the sample moments with corresponding distibution moments
- Equates sample moments to population ones

```
x.gam = rgamma(200, rate = 0.5, shape = 3.5) # sampling from gamma distributon with lamda = 0.5 and sha
med.gam = mean(x.gam) # sample mean
var.gam = var(x.gam) # sample variance
l.est = med.gam/var.gam # lambda estimate
a.est = ((med.gam)^2)/var.gam # shape estimate

l.est
```

```
## [1] 0.4791956
```

```
a.est
```

```
## [1] 3.296033
```

- Maximum likelihood Method can be computed using *mle()* included in *stats4* package and *fitdistr()* in package MASS

```
library(stats4)
ll = function(lambda, alfa){
  n = 200
  x = x.gam
  -n*alfa*log(lambda) +n*log(gamma(alfa)) - (alfa -1)*sum(log(x)) + lambda *sum(x)
}

est = mle(minuslogl = ll, start = list(lambda = 2, alfa = 1))
```

```
## Warning in log(lambda): NaNs produced
```

```
## Warning in minuslogl(lambda = -1273.6525816967, alfa = 612.605913307585):
## value out of range in 'gammafn'
```

```
## Warning in log(lambda): NaNs produced
```

```
## Warning in log(lambda): NaNs produced
```

```
## Warning in log(lambda): NaNs produced

## Warning in log(lambda): NaNs produced

## Warning in log(lambda): NaNs produced

## Warning in minuslogl(lambda = -1276.40905449053, alfa = -621.097005317676):
## underflow occurred in 'gammafn'

## Warning in log(lambda): NaNs produced

## Warning in log(lambda): NaNs produced

## Warning in log(lambda): NaNs produced

## Warning in log(lambda): NaNs produced

## Warning in log(lambda): NaNs produced

## Warning in log(gamma(alfa)): NaNs produced

## Warning in log(lambda): NaNs produced

## Warning in log(gamma(alfa)): NaNs produced

## Warning in log(lambda): NaNs produced

## Warning in log(gamma(alfa)): NaNs produced

## Warning in log(lambda): NaNs produced

## Warning in log(gamma(alfa)): NaNs produced

## Warning in log(gamma(alfa)): NaNs produced

## Warning in log(gamma(alfa)): NaNs produced
```

```
summary(est)
```

```
## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = ll, start = list(lambda = 2, alfa = 1))
##
## Coefficients:
##         Estimate Std. Error
## lambda 0.5396955 0.05538923
## alfa   3.7121678 0.35579176
##
## -2 log L: 1038.197
```

```
library(MASS)
fitdistr(x.gam, "gamma")
```

```
##      shape         rate
##   3.71210192   0.53968584
##  (0.35578734) (0.05538867)
```

```
fitdistr(x.wei, densfun = dweibull, start = list(scale = 1, shape = 2))
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
##      scale        shape
##   1.08066890   2.02049518
##  (0.03976689) (0.11214563)
```

```
fitdistr(x.rnorm,"normal")
```

```
##      mean         sd
##   9.9506386   2.0399272
##  (0.1442446) (0.1019964)
```

Measures of goodness of fit

- Useful for matching emperical frequencies with fitted ones by a theoretical model
- Example – Poisson distribution

```
lamda.est = mean(x.poi) # estimate of parameter lambda
tab.os = table(x.poi) # table with emperical frequencies
tab.os
```

```
## x.poi
##  0  1  2  3  4  5  6  7  8
## 22 42 54 39 26 13  2  1  1
```

```
freq.os = vector()
for( i in 1: length(tab.os))
  freq.os[i] = tab.os[[i]] # vector of emperical frequencies
freq.ex = (dpois(0:max(x.poi),lambda = lamda.est) * 200) # vector of fitted expected frequencies
freq.os
```

```
## [1] 22 42 54 39 26 13  2  1  1
```

```
freq.ex
```

```
## [1] 19.7532368 45.7287432 52.9310202 40.8451040 23.6391039 10.9449051
## [7]  4.2229092  1.3965764  0.4041343
```

```
acc = mean(abs(freq.os - trunc(freq.ex))) # absolute goodness of fit
acc
```

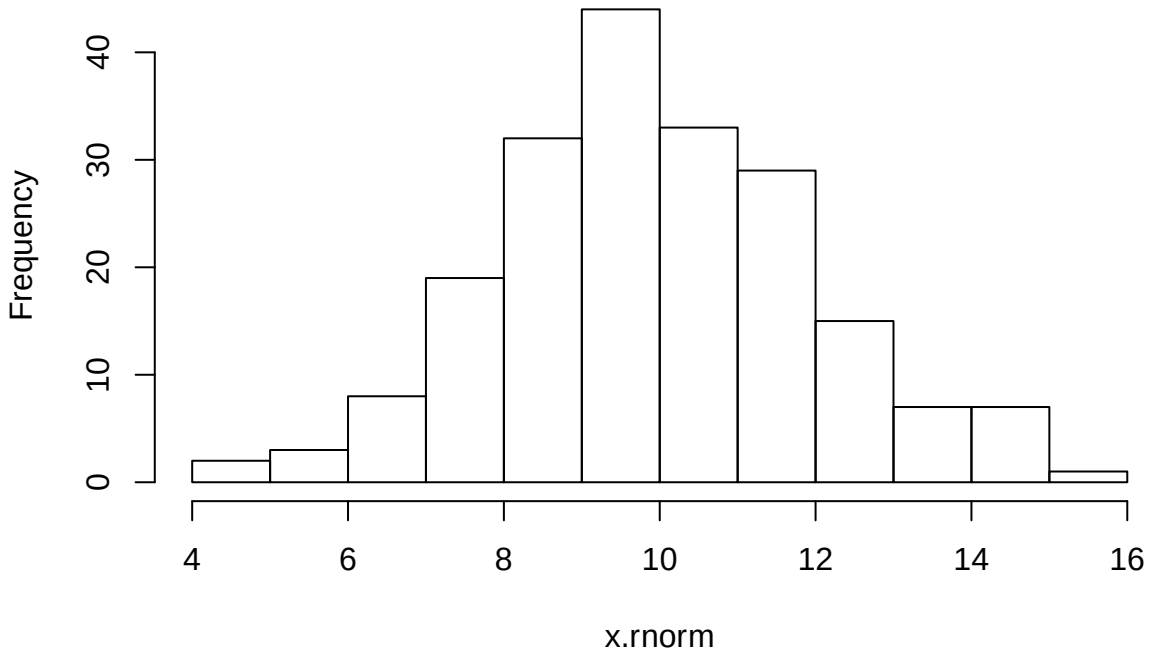
```
## [1] 2
```

```
acc/mean(freq.os)*100
```

```
## [1] 9
```

- Graphical techniques

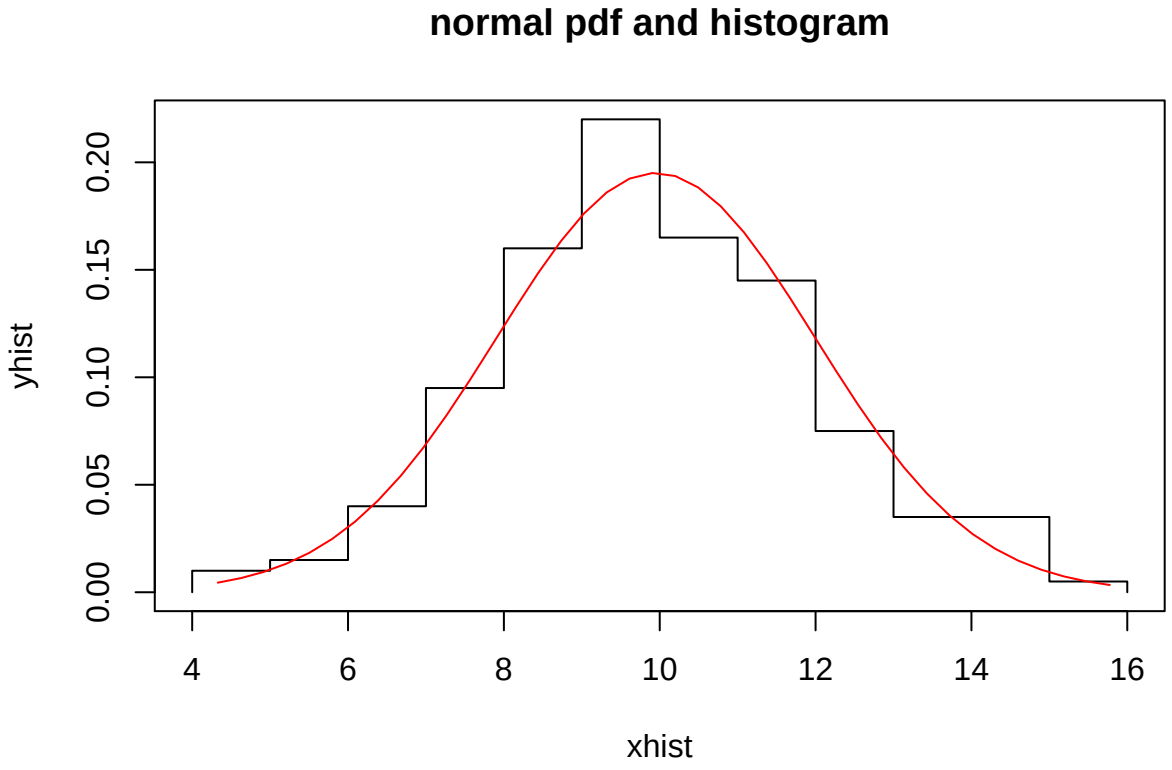
```
h = hist(x.rnorm, breaks = 15)
```




```
xhist = c(min(h$breaks),h$breaks)
yhist = c(0,h$density,0)

xfit = seq(min(x.rnorm),max(x.rnorm), length = 40)
yfit = dnorm(xfit, mean = mean(x.rnorm), sd = sd(x.rnorm))

plot(xhist,yhist, type = "s",ylim = c(0, max(yhist,yfit)), main = "normal pdf and histogram")
lines(xfit, yfit, col = "red")
```



Goodness of fit tests

- Goodness of fit tests indicate whether or not it is reasonable to assume that a random sample comes from a specific distribution.
- They are a form of hypothesis testing where the null and alternative hypothesis are:
  - H0: Sample data come from stated distribution
  - HA: Sample data do not come from the stated distribution
- These tests are called *omnibus test* and they are distribution free.

```
library(vcd)
```

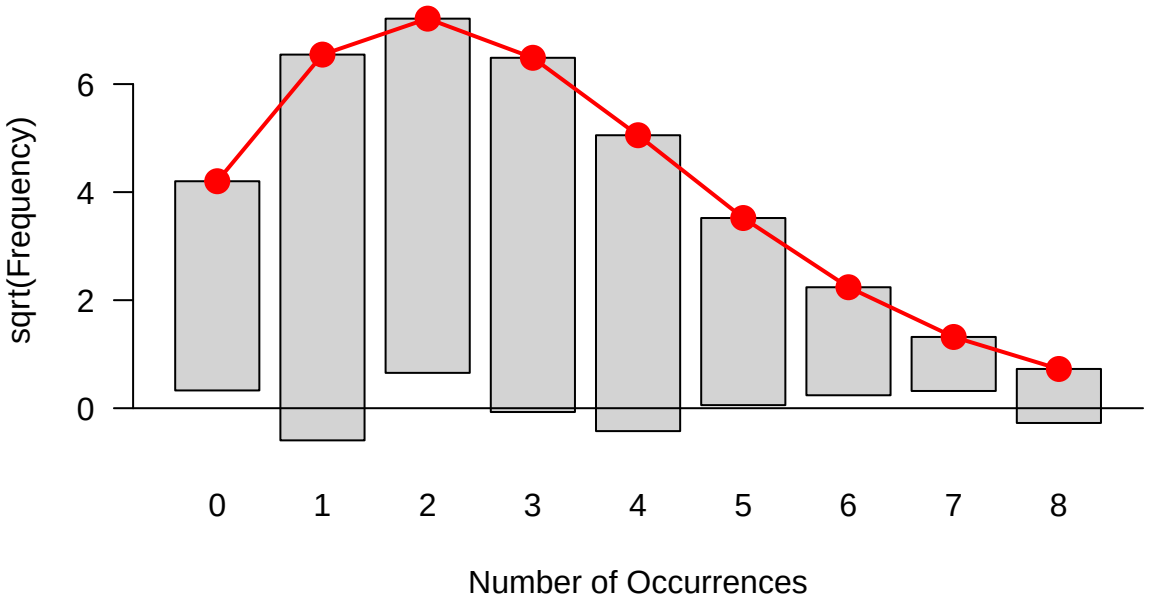
```
## Loading required package: grid
```

```
x.poi = rpois(n = 200, lambda = 2.5)
gf = goodfit(x.poi, type = "poisson", method = "MinChisq")
summary(gf)
```

```
## Warning in summary.goodfit(gf): Chi-squared approximation may be incorrect
```

```
##
##   Goodness-of-fit test for poisson distribution
##
##              X^2 df  P(> X^2)
## Pearson 4.955743  7 0.6653642
```

```
plot(gf, main = "Count data vs. Poisson distribution")
```



- Chi-square goodness of fit test can be applied to any univariate distribution for which you can calculate the cumalative distribution function.
- Chi-square goodness of fit test is applied to binned data (data put into classes).
- For non-binned data you can simple calculate the histogram or frequency table for generating chi-square test.
- Value of chi-square test statistic is dependent on how the data is binned.
- Requires sufficient sample size in order for chi-square approximations to be valid.
- Chi-square test can be applied either to discrete distributions or continuous ones.
- For chi-sqaure goodness of fit computation, data is divided into k bins and the test statistic is defined as follows

$$\chi^2 = \Sigma_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

- degree of freedom is k-p-1

```
x.gam = rgamma(200,rate = 0.5, shape = 3.5)
med.gam = mean(x.gam) # sample mean
var.gam = var(x.gam) #sample variance
l.est = med.gam/var.gam
a.est = ((med.gam)^2)/var.gam
x.gam.cut = cut(x.gam, breaks = c(0,3,6,9,12,18))
table(x.gam.cut)
```

```
## x.gam.cut
##   (0,3]   (3,6]   (6,9]  (9,12] (12,18]
```

```
##      24      77      49      25      20
```

```
# computing the expected frequencies
(pgamma(3,shape = a.est, rate = l.est) - pgamma(0,shape = a.est, rate = l.est))*200
```

```
## [1] 28.97455
```

```
(pgamma(6,shape = a.est, rate = l.est) - pgamma(3,shape = a.est, rate = l.est))*200
```

```
## [1] 67.23502
```

```
(pgamma(9,shape = a.est, rate = l.est) - pgamma(6,shape = a.est, rate = l.est))*200
```

```
## [1] 52.62651
```

```
(pgamma(12,shape = a.est, rate = l.est) - pgamma(9,shape = a.est, rate = l.est))*200
```

```
## [1] 28.87578
```

```
(pgamma(18,shape = a.est, rate = l.est) - pgamma(12,shape = a.est, rate = l.est))*200
```

```
## [1] 18.88232
```

```
# expected frequencies vector
f.ex = c(33,75,52,26,14)

f.os = vector()
for(i in 1:5)
  f.os[i] = table(x.gam.cut)[[i]] #emperical frequencies
x2 = sum(((f.os-f.ex)^2)/f.ex)
gdl = 5-2-1 #degrees of freedom
1-pchisq(x2,gdl)
```

```
## [1] 0.07097533
```

- p-value is greater than 0.05, therefore we fail to reject null hypothesis

Tests for normality

- checks if data colledted come from normal distribution or not.
- Shapiro-Wilk test is one of the most powerful normality tests, especially for small samples.

```
x.norm = rnorm(n =200, m = 10, sd = 2)
shapiro.test(x.norm)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x.norm
## W = 0.98686, p-value = 0.06069
```

- p-value is higher than signifcance level, therefore we fail to reject null hypothesis