Unit III(STB555)—Simulation and Fitting of Distributions

Athar Ali Khan (Email: atharkhan1962@gmail.com)
Department of Statistics and O R
Aligarh Muslim University, Aligarh.

November 26, 2020

Contents

List of Figures

Chapter 1

Random number generation

In this chapter we shall study about the **R** implementations for random number generation and sampling procedures. Fitting of polynomials and exponential curves. Application problems based on fitting of suitable distributions. Normal probability plot.

1.1 Rndom number generation from uniform $U(a, b)$

Random numbers can be generated by the function `runif()`. A general syntax is:

```
runif(n,min=a,max=b)
```

Execution of this command produces n pseudo random uniform numbers on the interval $[a, b]$. The default values are $a = 0$, and $b = 1$. The seed is selected internally. However, one can set the seed by using `set.seed()` function.

1.1.1 Example unifrom deviates

Generate 5 uniform pseudo random numbers on the interval $[0, 1]$, and 10 uniform such numbers on the interval $[-3, -1]$.

```
runif(5) # same as runif(5,0,1)

## [1] 0.9930331 0.1698851 0.2137684 0.9921755 0.9442345

runif(10,min=-3,max=-1)
```

```
##  [1] -2.585325 -1.411351 -1.493244 -2.332277 -2.493447
##  [6] -1.988594 -1.694848 -2.312639 -2.556556 -1.921420
```

One can use `set.seed()` to generate the same number every time.

```
set.seed(1)
runif(5)

## [1] 0.2655087 0.3721239 0.5728534 0.9082078 0.2016819

set.seed(1)
runif(10,-3,-1)

##  [1] -2.468983 -2.255752 -1.854293 -1.183584 -2.596636
##  [6] -1.203221 -1.110649 -1.678404 -1.741772 -2.876427
```

1.1.2 Exercise: random deviates from $U(a, b)$

Use `runif()` with `set.seed(32078)` to generate 10 pseudo random numbers from:

(a) the uniform $U(0, 1)$ distribution

(b) the uniform $U(3, 7)$ distribution

(c) the uniform $U(-2, 2)$ distribution.

1.1.3 Example on application of simulation

Generate 1000 uniform random variate using `runif()` function, assigning them to a vector called `U`. Do the following:

(a) Compute the average, variance,and standard deviation of the numbers in `U`.

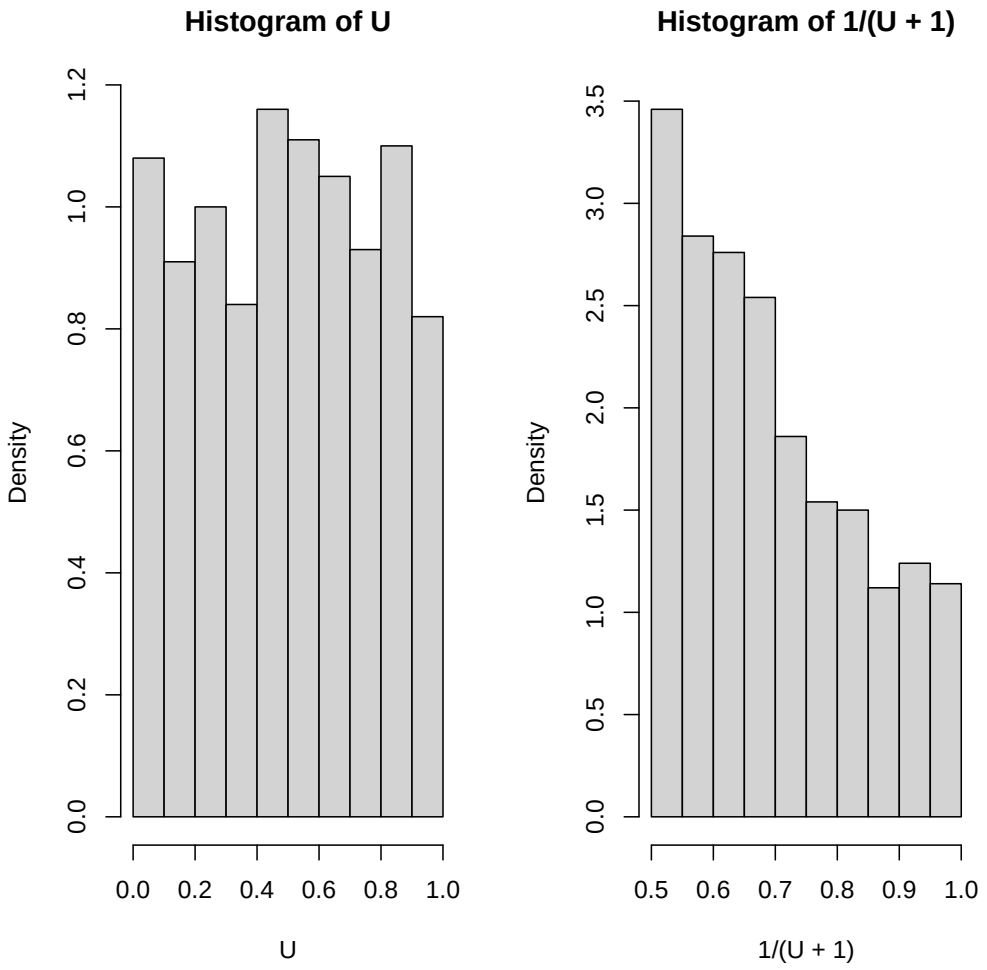
(b) Compare your results with true mean, variance, and standard deviation.

(c) Compute the proportion of values of `U` that are less than 0.6,and compare with the probability that uniform random variable `U` is less than 0.6.

(d) Estimate the expected value of $\frac{1}{U+1}$.

Solution using R commands

```
set.seed(19908)
U<-runif(1000)
# Part (a)
mean(U) # it should be 0.5

## [1] 0.496057

var(U) # it should be 1/12.

## [1] 0.08148008

sd(U)  # it should be sqrt(1/12)

## [1] 0.2854472

# Part (c)
# Note that U<0.6 returns a logical vector, Thus
mean(U<0.6) # same as Prob(U<0.6)=punif(0.6), F(x)=x for uniform.

## [1] 0.61

# Part (d) E(1/(U+1)) is
mean(1/(U+1))

## [1] 0.6946063

# histogram of U and x=1/(U+1)
opar<-par(mfrow=c(1,2))
hist(U,prob=TRUE)
hist(1/(U+1),prob=TRUE)
```




```
par(opar)
```

Note that the distribution of U is uniform whereas distribution of $\frac{1}{U+1}$ is more close to exponential distribution. These facts will be discussed in more detail in the later part of the course.

1.1.4 Exercise on uniform $U(3.7, 5.8)$

Simulate 10000 independent observations on a uniformly distributed random variable on the interval $[3.7, 5.8]$.

(a) Estimate the mean, variance, and standard deviation of such a uniform random variable and compare your estimates with the true values.

(b) Estimate the probability that such a random variable is greater than 4. Compare it with true value.

1.2 The `sample()` function

The `sample()` function allows you to take a simple random sample from a vector of values. For example,

`sample(c(3,5,7),size=2,replace=FALSE)`

will yield a vector of two values taken (without replacement) from the set {3,5,3}. Use the function `sample()` to generate 50 pseudo random integers from 1 through 100:

(a) sample without replacement

(b) sample with replacement.

Solution

```
# Part (a)
x<-sample(1:100,size=50) #without replacement
x

##  [1] 70 24 51 67  2 90 35 66 30 82 57 22 34 21 17 58
## [17]  1 20  7  5 75 77 79 94 29 45  9 85 54 63 41 25
## [33] 37 49 19 14 89 80 47 83 32 16 71 93  4 55 18 95
## [49] 73 72

# part (b)
y<-sample(1:100,size = 50,replace = TRUE) #with replacement
y

##  [1] 34 64 23 96 13 58 63 91 44 30 55 97 41 35 19 36
## [17] 49  9 94 47 39 99 35 86 46 67 68 95 68 42 56 10
## [33] 41 85 47 89 21 73  9 14 96 90 43 88 20 19 54 96
## [49] 84 11

# Note that in y repetitions are allowed, whereas it is not in x.

## The function sample is also used for randomization
sample(c("A","B","C")) #returns randomization.

## [1] "B" "A" "C"
```

1.3 Distributions in R

Most of the distributions are implemented in **base R**. Remaining are implemented in other packages.

1.3.1 Normal distribution $N(\mu, \sigma)$

There four main aspects of a distribution which are implemented in R. They are:

- `dnorm(x,mean,sd)`, this is the value of the distribution with `mean` and `sd`.
- `pnorm(q,mean,sd)`, the value of the CDF at q. You know that CDF is defined as $P(X < q) = F(q)$.
- `qnorm(p,mean,sd)`, this is the value of quantile at probability p. Note that quantile is the inverse image of CDF.
- `rnorm(n,mean,sd)`, stands for random simulations from $N(\mu, \sigma)$.

We are illustrating them using **R** commandss:

```
# Suppose we want to plot N(0,1).
curve(dnorm(x,mean=0,sd=1),from=-3,to=3)
# Value of density at 0
dnorm(0) # same as sqrt(2*pi)
```

```
## [1] 0.3989423
```

```
1/sqrt(2*pi)
```

```
## [1] 0.3989423
```

```
abline(v=0,lwd=2)
#value of cdf at 1.96
pnorm(1.96)
```

```
## [1] 0.9750021
```

```
abline(v=1.96)
pnorm(-1.96)
```

```
## [1] 0.0249979
```

```
abline(v=-1.96)
```

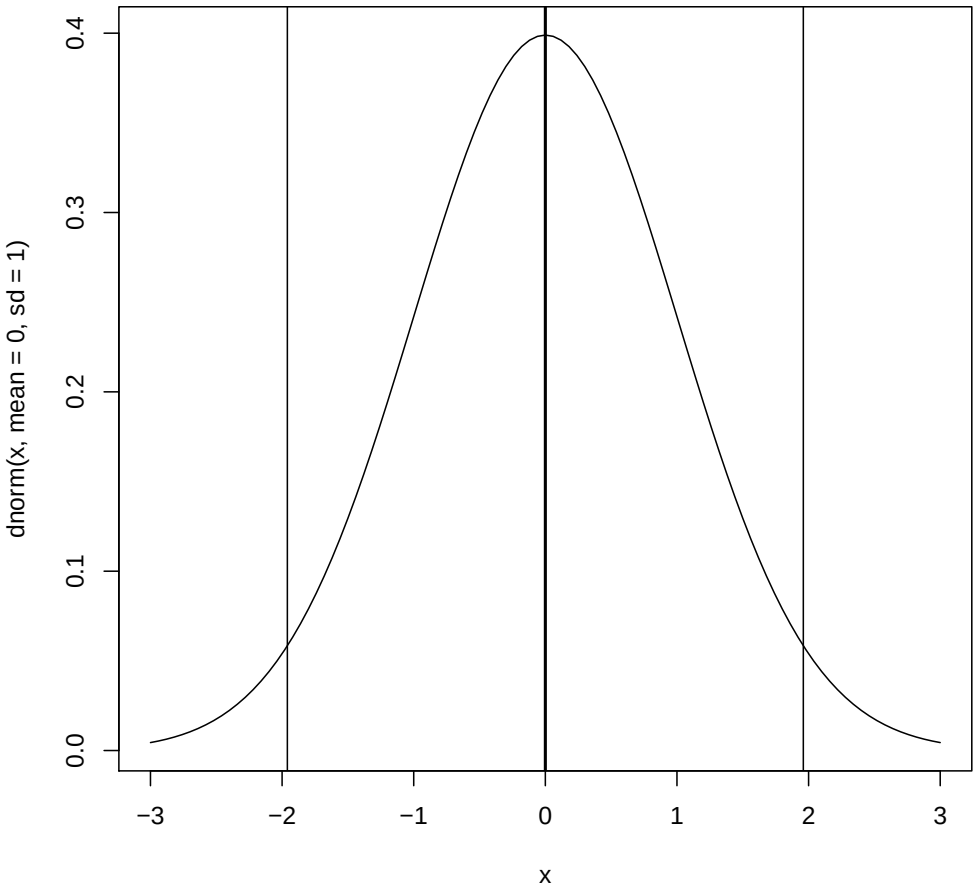


Figure 1.1: Standard Normal Distribution

```r
# Note that area between -1.96 and 1.96 is 0.95
pnorm(1.96)-pnorm(-1.96)

## [1] 0.9500042

# What should be pnorm(0), it should be 0.5
pnorm(0)

## [1] 0.5

## An example of qnorm(.5)
qnorm(0.5)

## [1] 0

qnorm(0.975)

## [1] 1.959964

qnorm(0.025)

## [1] -1.959964

## An example of 10 random simulation from N(0,1)
rnorm(n=10,mean=0,sd=1)

##  [1]  0.6902348  0.6467930 -0.8413699 -0.5348958
##  [5] -1.2831117  0.8464102  0.4518511 -0.7419146
##  [9] -0.1756651  0.2121791
```

1.3.2 Exponential distribution

Like normal we can also define exponential distribution:

```
dexp(x,rate) #value of density
pexp(q,rate) #value of cdf
qexp(p,rate) #value of quantile
rexp(n,rate) # random simulation from exponential
```

```r
re<-rexp(n=10000,rate=1) #standard exponential
mean(re)

## [1] 0.9833871

sd(re)

## [1] 0.983216

#compute Pr(re>3)
mean(re>3) # probabilty re>3, through si,ulation

## [1] 0.0458

# Check it using pexp
pexp(3,lower.tail = FALSE) #exact

## [1] 0.04978707

hist(re,prob=TRUE)
curve(dexp(x,rate=1),add=TRUE,lwd=2)
```

Since histogram is made from simulated data and curve is exact, but both are matching. This implies that our simulation is correct.

1.4 Inverse CDF method of simulation

It is a well known result of probability that

$$If \quad X \sim F, cdf \quad of \quad X$$

then

$$F(X) \sim U(0,1)$$

This implies that one can simulate quantiles from F by using the following algorithm:

(i) Equal the cdf $F(x) = u$ and solve it for x, where u is $U(0,1)$ deviate.

(ii) The equation $x = F^{-1}(u)$ will be a quantile from F.

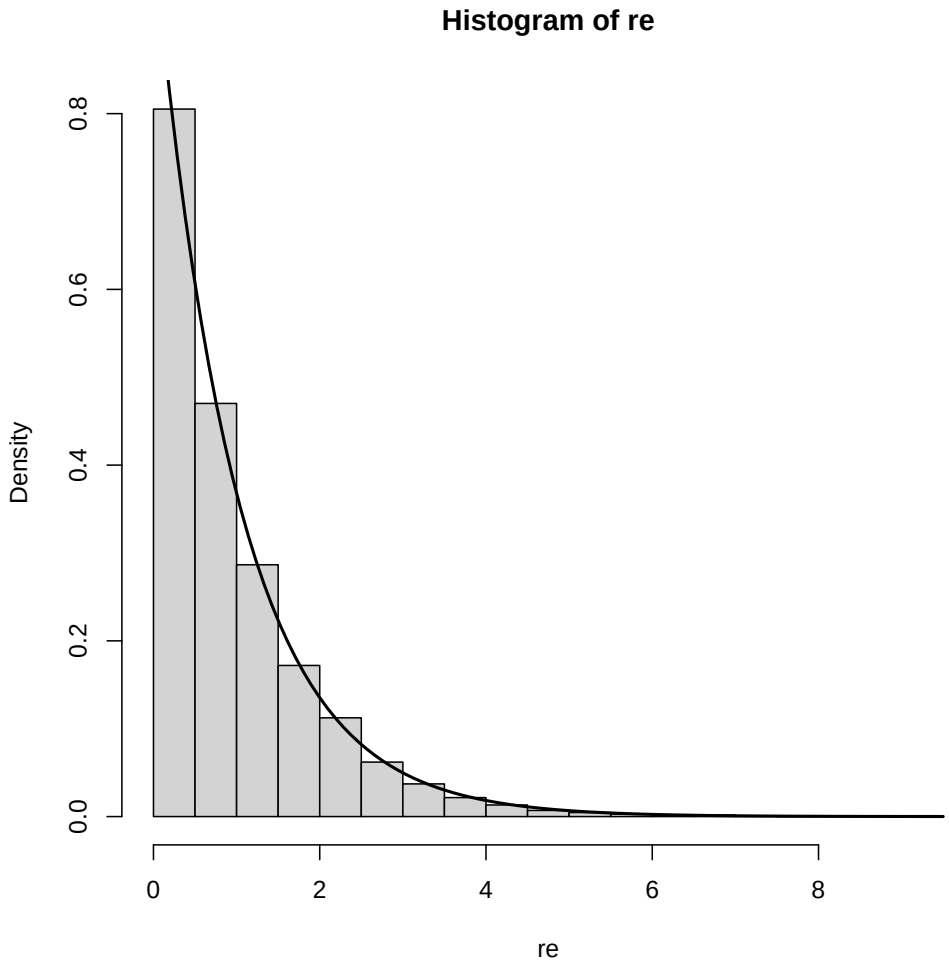


Figure 1.2: Standard Exponential distribution

1.4.1 Simulation from exponential distribution

We can make use of *Inverse CDF* method for simulation from exponential distribution with $rate = \lambda$. Note that

$$f(x) = \lambda e^{-\lambda x} \quad x > 0 \Rightarrow F(x) = 1 - e^{-\lambda x} \tag{1.1}$$

Thus,

$$F(x) = u \Rightarrow 1-e^{-\lambda x} = u \Rightarrow 1-u = e^{-\lambda x} \Rightarrow log(1-u) = -\lambda x \Rightarrow x = -\frac{1}{\lambda} log(1-u) \tag{1.2}$$

Thus **R** commands are:

```
u<-runif(1000)
lambda<-.02 # value of rate
x<--(1/lambda)*log(1-u) # simulated quantiles
#Check for mean, 1/0.02=50
mean(x) #1/lambda

## [1] 49.3553

var(x) #1/lambda^2

## [1] 2328.932

sd(x) #1/lambda

## [1] 48.25901

mean(x>50) #Prob(X>50)

## [1] 0.377

hist(x,prob=TRUE)
curve(dexp(x,rate=.02),add=TRUE,lwd=2)
abline(v=50,col="red",lwd=2) # vertical line at mean
```

Note that simulated results obtained by using *Inverse CDF* method are matching with the exact results, both in terms of numeric as well as graphics.

1.4.2 Exercise on Simulation from Possion

Estimate the mean and variance of a Poisson random variable whose mean is 7.2 by simulating 10000 Poisson pseudo random numbers. Compare your

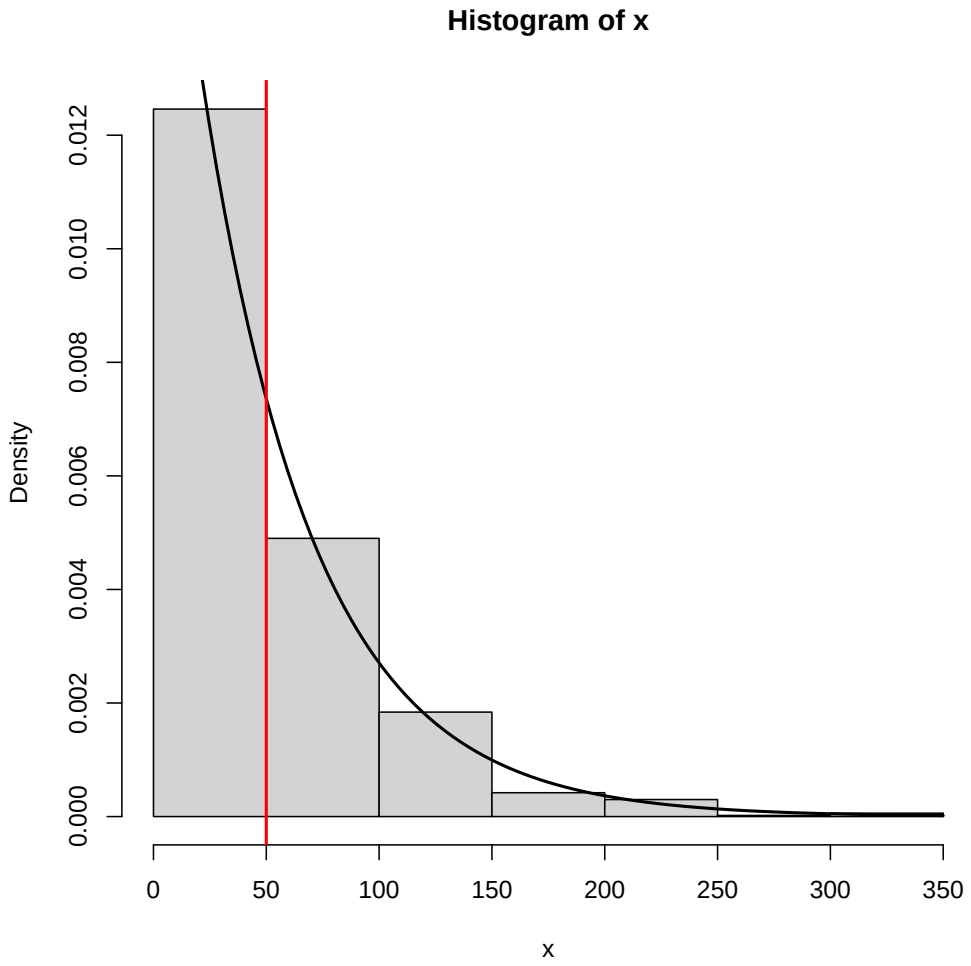


Figure 1.3: Exponential density with mean 50 or rate=0.02

results with theoretical results. (Hint `rpois(n,lambda)`).

1.4.3 Exercise on simulation from $N(0,1)$

Simulate 10000 realizations of standard normal random variable Z, and use your simulated sample to estimate:

(a) $P(Z > 2.5)$

(b) $P(0 < Z < 1.645)$

(c) $P(1.2 < Z < 1.45)$

(d) $P(-1.2 < Z < 1.3)$

Compare your results with exact ones using `pnorm()`.
Hint:

```
Z<-rnorm(10000)
(a) mean(Z>2.5)
(b) mean(Z>0 & Z<1.645)
```

1.4.4 Exercise on Simulation from χ^2 distribution

Simulate 1000 realizations of a $\chi^2_{(8)}$, and estimate its mean and variance. Compare it with theoretical mean (8) and variance (16). Note that for $\chi^2_{(k)}$ the mean is k and variance is $2k$. Make a histogram of simulated data and overlay exact density curve on it.

Hint:

```
x<-rchisq(n=1000,df=8)
mean(x)
var(x)
```

1.4.5 Exercise on simulation from binomial

Simulate 1000 observations from $binomial(10, 0.5)$. Compute its mean and variance,and compare it with true mean(10×0.5) and variance ($10\times0.5\times0.5$).

(a) Make a histogram of simulated observations, and add a Normal curve $N(5, 2.5)$ on it.

(b) Comment on the closeness of histogram and Normal curve. Do you feel that binomial probability can be approximated by Normal probability?

Hint:

```
x<-rbinom(n=1000,size=10,prob=0.5)
mean(x)
var(x)
```

1.5 Law of large numbers

It states that as

$$n \to \infty \Rightarrow \bar{x} \xrightarrow{p} \mu \tag{1.3}$$

Thus for large sample size, sample mean ($\bar{x}$) converges to population mean (μ).

1.5.1 Height of American women

In this example we simulate heights of American women from $N(64.5, 2.5)$ try to see the behavior of sample mean which converges to population mean as sample size becomes large. This phenomenon has been illustrated in the following **R** commands.

Note that we have used a new function `sapply()` whose first argument is a vector or listed data and second argument is the expression which calls the first argument repeatedly. For example, in the above situation `sapply()` repeatedly calculates mean of random sample of sizes $1 : 10000$. The graphic output is reported in Figure 1.4.

1.6 Sampling distribution—`replicate()`

Note that *statistic* is a function of sampled values, and distribution of a statistic is called *sampling distribution*. In **R**, the function `replicate()` is an implementation of sampling distribution. Its syntax is:

```
replicate(n,expr)
```

1.6.1 Sampling from normal dsitribution

If $x_i \sim N(0, 1)$, for $i = 1 \cdots n$,then $\bar{x} \sim N(0, \frac{1}{n}$. Note that n is not an argument of `replicate()`. We can express this fact using the function `replicate()` for $n = 10$, in the following **R** command. This command will simulate $n = 10$ random observations from $N(0, 1)$ and then computes its mean. This process is repeated 1000 times, resulting 1000 sample means

```r
#generate sample means for different sample sizes.
xbar<-sapply(1:10000, function(n) mean(rnorm(n,64.5,2.5)))
plot(xbar,xlim=c(0,10000))
abline(h=64.5,lwd=2,col="blue")
```

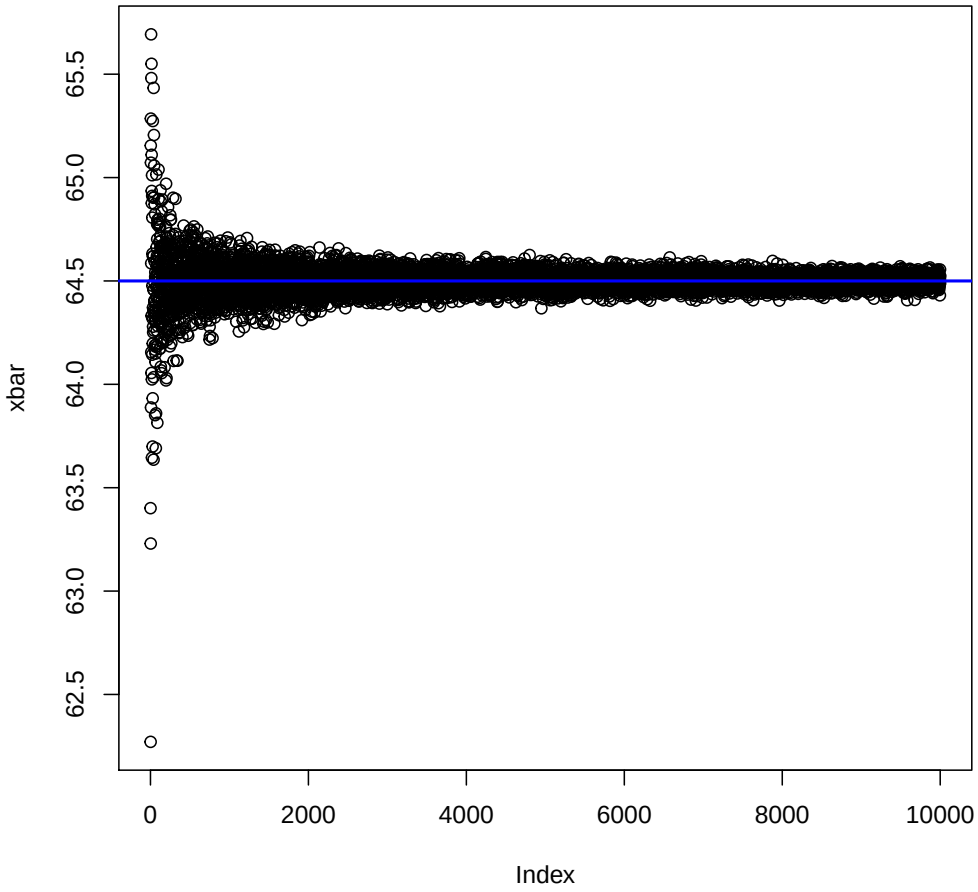


Figure 1.4: Note that as sample size increases,that is $n \to \infty$, sample mean converges to population mean (64.5),that is, $\bar{x} \xrightarrow{p} \mu$, which is shown by the blue horizontal line. Note that after $n = 4000$ sample mean is converging around the population mean 64.5.

from $N(0,1)$. We can verify the generated sample numerically as well as graphically.

```
xbar<-replicate(1000,mean(rnorm(10)))
mean(xbar) # zero

## [1] 0.0005253195

sd(xbar) #1/sqrt(10)

## [1] 0.3149039
```

Graphical illustration of sampling from $N(0,1)$

```
xbar<-replicate(1000,mean(rnorm(10)))
hist(xbar,prob=TRUE)
curve(dnorm(x,mean = 0,sd=1/sqrt(10)),add=TRUE,lwd=1)
```

1.7 Central limit theorem (CLT)

If $x_i \sim (\mu, \sigma^2)$ for $i = 1, \cdots, n \Rightarrow \bar{x} \xrightarrow{L} N(\mu, \frac{\sigma^2}{n})$. This is known as *central limit theorem*. Note that in **CLT** even if parent population is not known the sampling distribution of mean can be approximated by a normal distribution $N(\mu, \sigma^2/n)$.

1.7.1 CLT when parent is *standard exponential*

Suppose that parent population is standard exponential, and a sample of size 10 is taken to see the sampling behavior of the sample mean.This means that:

$$f(x) = \lambda e^{-\lambda x},\ E(x) = \frac{1}{\lambda},\ Var(x) = \frac{1}{\lambda^2},\ SD(x) = \frac{1}{\lambda} \tag{1.4}$$

Thus for standard exponential

$$f(x) = e^{-x},\ mean = 1,\ sd = 1,\ E(\bar{x}) = n,\ SD(\bar{x}) = \frac{1}{\sqrt{n}} \tag{1.5}$$

Using CLT one can approximate the sampling distribution of sample mean by a normal distribution $N(mean = 1, sd = 1/\sqrt{10})$. Note that exact distribution of sample mean is gamma distribution with $shape = 10$ and $rate = 10$. Keeping in view these facts we sue the following set of (R) commands:

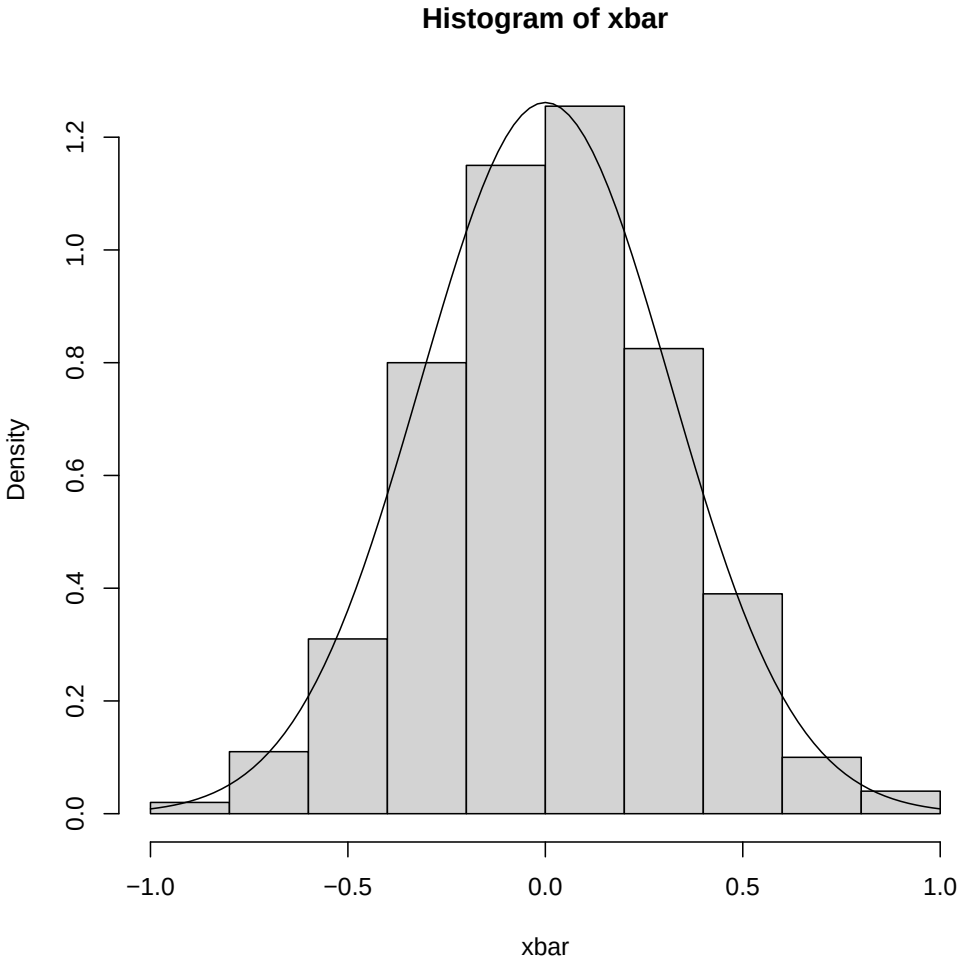


Figure 1.5: Sampling distribution from N(0,1) for a sample of size 10.

```r
#simulate 1000 sample means of size 10 from dexp(x,rate=1)
xbare10<-replicate(1000,expr=mean(rexp(10)))
mean(xbare10) #mean=1

## [1] 0.9994845

sd(xbare10) #1/sqrt(10)

## [1] 0.3340785

hist(xbare10,prob=TRUE,ylim=c(0,1.4))
curve(dnorm(x,1,1/sqrt(10)),add=TRUE)
curve(dgamma(x,shape=10,rate=10),add=TRUE,lty=2)
legend("topright",legend = c("Normal approximation","Exact"),lty=c(1,2))
```

Note that approximation is not excellent as $n = 10$ only. See what happens when $n = 100$. We expect that it will improve the approximation. Whole story can bee seen in the following set of commands:

```r
#simulate 1000 sample means of size 100 from dexp(x,rate=1)
xbare100<-replicate(1000,expr=mean(rexp(100)))
mean(xbare100) #mean=1

## [1] 1.001335

sd(xbare100) #1/sqrt(100)

## [1] 0.1011547

hist(xbare100,prob=TRUE,ylim=c(0,4))
curve(dnorm(x,1,1/sqrt(100)),add=TRUE)
curve(dgamma(x,shape=100,rate=100),add=TRUE,lty=2)
legend("topright",legend = c("Normal approximation","Exact"),lty=c(1,2))
```

1.8 Student's t distribution with 5 dgrees of freedom

If

$$t = \frac{z}{\sqrt{x/k}} \sim t_k$$

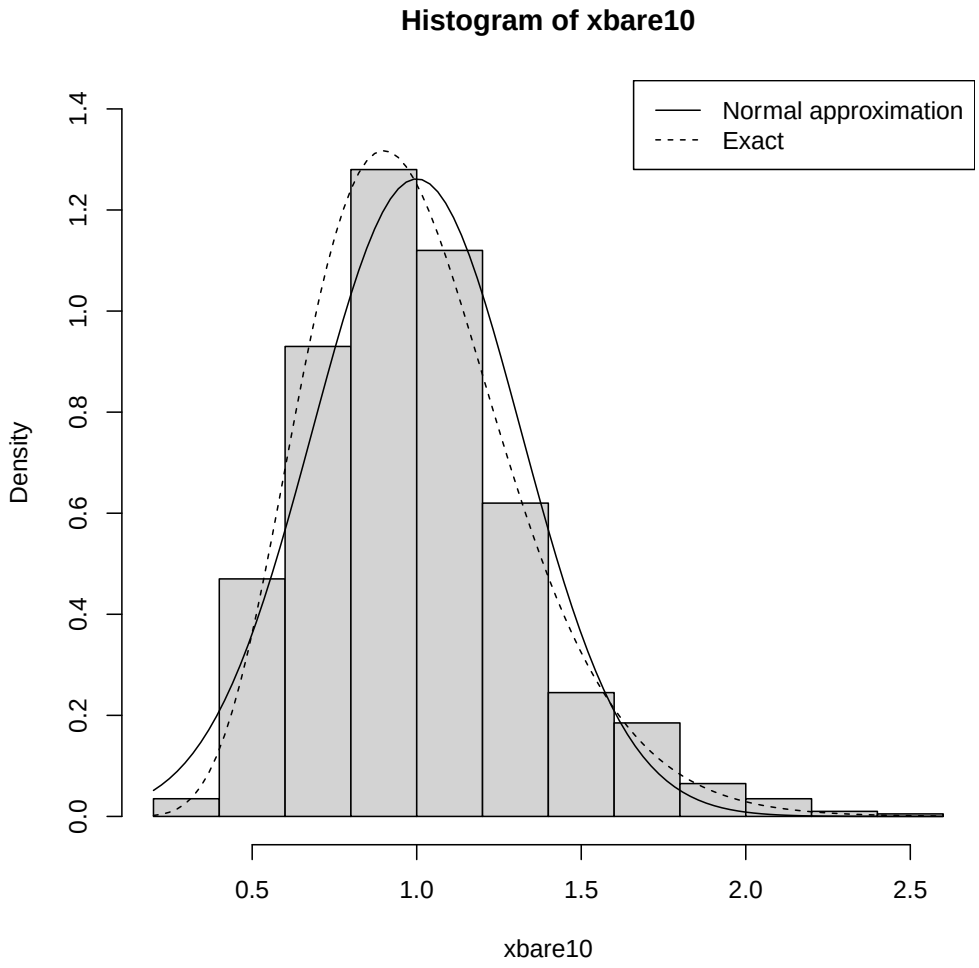


Figure 1.6: Comparison of exact and approximate sampling distributions of sampled mean (n=10) from standard exponential.

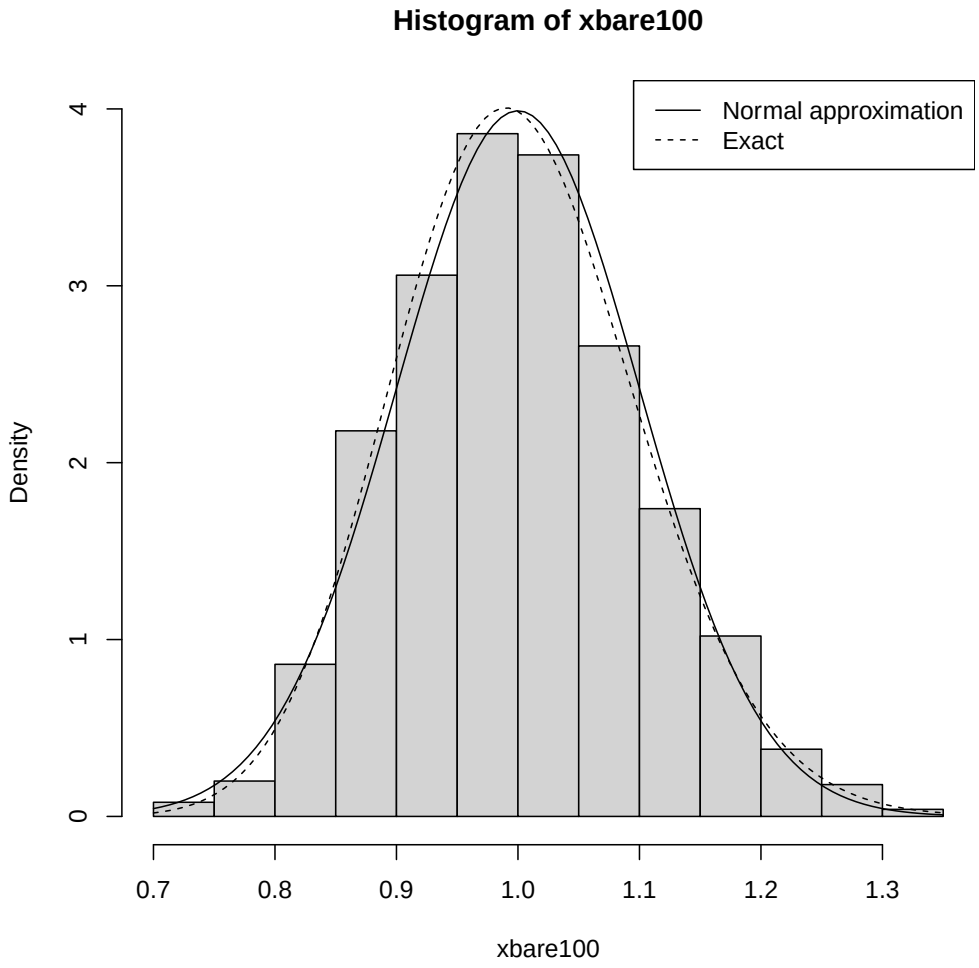


Figure 1.7: Comparison of exact and approximate sampling distributions of sampled mean (n=100) from standard exponential.

then it is termed as Students't distribution with k degrees of freedom. It may be noted that $z \sim N(0,1)$ and $x \sim \chi^2_k$, that is χ^2 with k degrees of freedom. It must be noted that both z and x are independent random variables. We implement this sampling distribution with $k = 5$ degrees of freedom as:

```
t5<-replicate(n=1000,expr=rnorm(1)/sqrt(rchisq(1,df=5)/5))
hist(t5,prob=TRUE,ylim=c(0,0.4),xlim=c(-5,5))
curve(dt(x,df=5),add=TRUE)
```

Note that histogram is constructed on simulated values using `replicate()` whereas density curve is exact t with 5 degrees of freedom. However, both are matching, which is an indication of the fact that both follow t distribution with 5 degrees of freedom.

1.8.1 χ^2_k distribution with k degrees of freedom

It is a well known fact that if $z \sim N(0,1)$ then $z^2 \sim \chi^2_1$, that is χ^2 distribution with single degree of freedom. Similarly, χ^2_5 distribution with 5 degrees of freedom is defined as if

$$z_i \sim N(0,1),\ i = 1, \cdots, 5$$

then

$$\Sigma_{i=1}^{i=5} z_i^2 \sim \chi^2_5$$

This fact can be implemented using **R** commands as:

```
chisq5<-replicate(1000,expr=sum(rnorm(5)^2))
hist(chisq5,prob=TRUE)
curve(dchisq(x,df=5),add=TRUE)
```

It is evident from above figure that simulated and exact curve are very close, depicting the fact that both represent the same distribution.

1.9 Exercise on $F(2, 12)$

Define F distribution with 2 and 12 degrees of freedoms. Implement that using the `replicate()` function, and support your findings graphically.

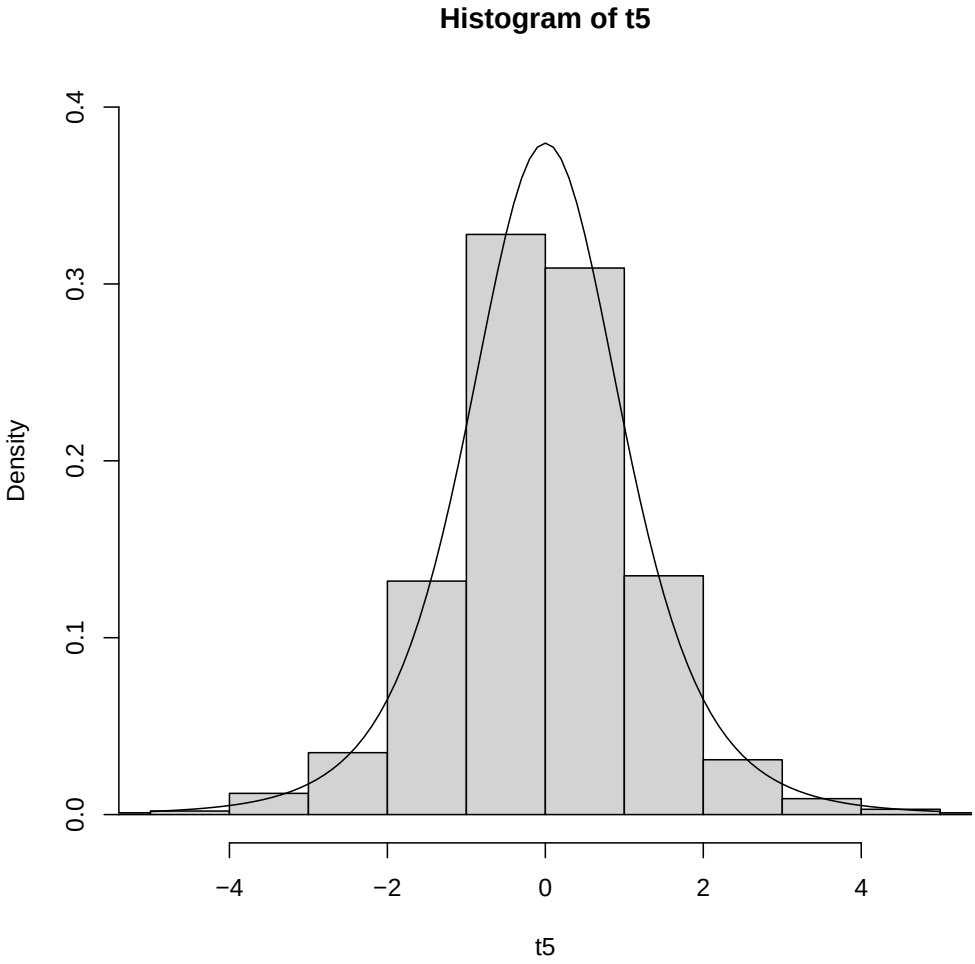


Figure 1.8: Students,t distribution with 5 degrees of freedom

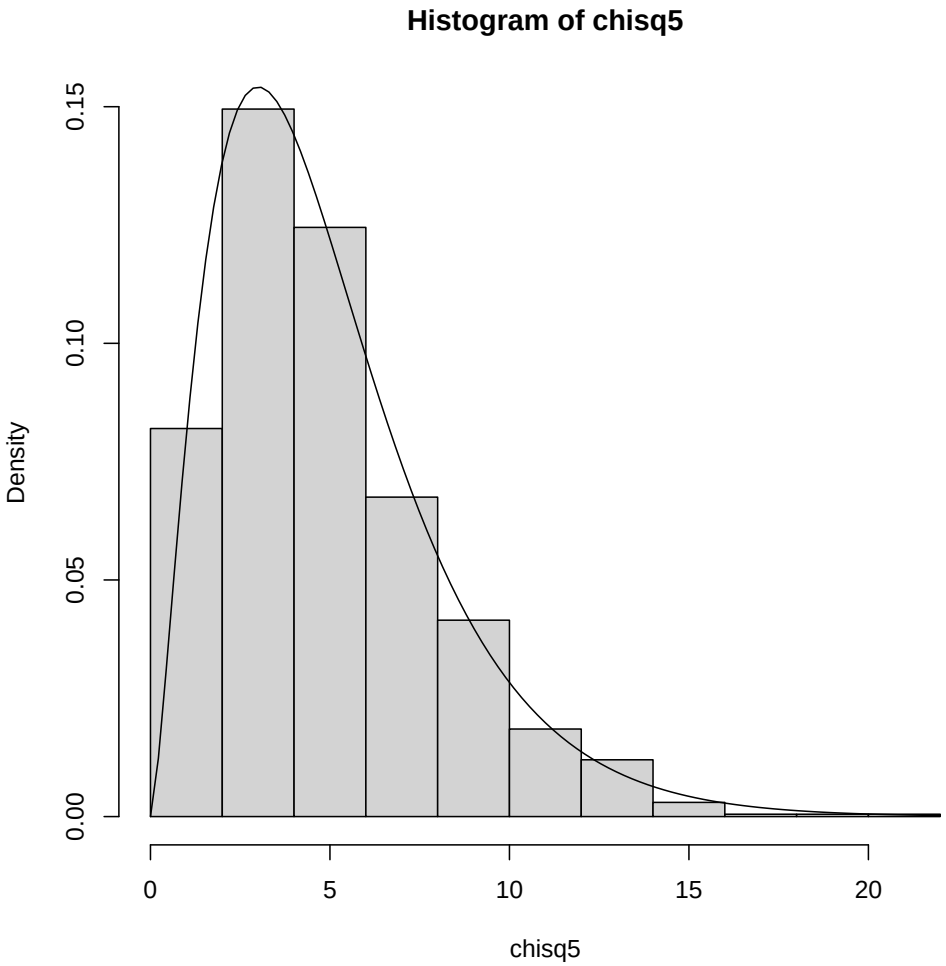


Figure 1.9: Chi-square sampling distribution with 5 degrees of freedom.

1.10 Integration using simulation for *beta* distribution

Compute $E(X)$ where $x \sim beta(shape1 = 3, shape2 = 4)$ using simulation and verify that by using `integrate()` function of `R`.

```
x<-rbeta(10000,shape1=3,shape2=4)
mean(x) #(3/(3+4))

## [1] 0.4282891

#Verification using integrate
# define a function for integrand of E(x)
f1<-function(x,shape1,shape2) x*dbeta(x,shape1,shape2)
integrate(f1,shape1=3,shape2=4,lower=0,upper=1)

## 0.4285714 with absolute error < 4.8e-15
```

Note that there is excellent agreement between the two results obtained from simulation as well as integration.

1.11 Fitting of linear, quadratic, exponential, log-log models

The `ironslag` data available with the **DAAG** package is a data frame which has 53 rows and 2 columns on measurements of iron contents obtained from iron slags, by using two methods, *chemical* and *magnetic* . A scatter plot of data suggests that chemical and magnetic variables are positively correlated, but the relation may not be *linear*. From the plot, it appears that a quadratic polynomial, or possibly an exponential or logarithmic model might fit the data better than a line. Thus, proposed models for y (magnetic measurement) vs x (chemical measurement) are:

$$M_1\ Linear: \quad y = \beta_0 + \beta_1 x + e \tag{1.6}$$

$$M_2\ Quadratic: \quad y = \beta_0 + \beta_1 x + \beta_2 x^2 + e \tag{1.7}$$

$$M_3\ Exponential: \quad log(y) = log(\beta_0) + \beta_1 x + e \Rightarrow y = \beta_0 e^{\beta_1 x} e^{error} \tag{1.8}$$

$$M_4\ Log - Log: \quad log(y) = \beta_0 + \beta_1 log(x) + e \Rightarrow y = e^{\beta_0} x^{\beta_1} e^{error} \tag{1.9}$$

Table 1.1: Iron slag data on two measurements

chemical	magnetic
24	25
16	22
24	17
18	21
18	20
10	13

1.11.1 The `ironslag` data with DAAG

The package **DAAG** contains the `ironslag` data frame which can obtained from the package as:

```
library(DAAG)
data(ironslag)
require(knitr)
kable(head(ironslag),caption="Iron slag data on two measurements")
```

1.11.2 Fitting of four models

```
opar<-par(mfrow=c(2,2))
x<-seq(10,40,.1) #sequence in the range of chemical
#Fit linear model M1
M1<-lm(magnetic~chemical,data=ironslag)
plot(magnetic~chemical,data=ironslag,main="Linear",pch=16)
yhat1<-M1$coef[1]+M1$coef[2]*x
lines(x,yhat1,lwd=2)
# Fit quadratic model M2
M2<-lm(magnetic~chemical+I(chemical^2),data=ironslag)
plot(magnetic~chemical,data=ironslag,main="Quadratic",pch=16)
yhat2<-M2$coef[1]+M2$coef[2]*x+M2$coef[3]*x^2
lines(x,yhat2,lwd=2)
# Fit Model M3
M3<-lm(log(magnetic)~chemical,data=ironslag)
plot(magnetic~chemical,main="Eponential",pch=16,data=ironslag)
logyhat3<-M3$coef[1]+M3$coef[2]*x
```

```
yhat3<-exp(logyhat3)
lines(x,yhat3,lwd=2)
# fit Model M4
M4<-lm(log(magnetic)~log(chemical),data=ironslag)
plot(log(magnetic)~log(chemical),main="Log-Log",data=ironslag,pch=16)
logyhat4<-M4$coef[1]+M4$coef[2]*log(x)
lines(log(x),logyhat4,lwd=2)
```

```
par(opar) #go to default settings
```

It is a different issue that which model fits best. Form graphics it seems that *Quadratic* model fits best. However, this is totally subjective on the basis of graphics. There are standard objective methods to compare different models. These are the issues which will be covered at higher level courses.

1.12 Fitting of distributions using ML methods

Fitting of distributions is a very simple task in **R**. We shall use the function `optim()` for maximum likelihood estimation. This function returns negative of the *Hessian* matrix provided negative of log-likelihood is the objective function to be minimized. It is a well known fact that if $l_i(\theta)$ is the log-likelihood for i^{th} observation and $l(\theta)$ is the loglikelihood for n observations, then

$$l(\theta) = \Sigma_{i=1}^{n} l_i(\theta) \; for \; i = 1, \cdots, n \tag{1.10}$$

Thus maximum likelihood estimate is obtained by

$$min(-l(\theta)) \tag{1.11}$$

and *Hessian* matrix is $-l''(\theta)|_{\theta=\hat{\theta}} = I(\hat{\theta})$, which is termed as observed Fisher's information matrix. It may be noted that that asymptotic variance of $\hat{\theta}$ is $I^{-1}((\hat{\theta}))$. This implies that standard error of $\hat{\theta}$ is

$$\sqrt{(diag(I^{-1}(\hat{\theta})))}$$

. Thus, $100(1-\alpha)\%$ confidence interval for θ is

$$\hat{\theta} \pm qnorm(\alpha/2, lower.tail = FALSE) * se(\hat{\theta})$$

where

$$se(\hat{\theta}) = \sqrt{(diag(solve(I(\hat{\theta}))))}$$

.

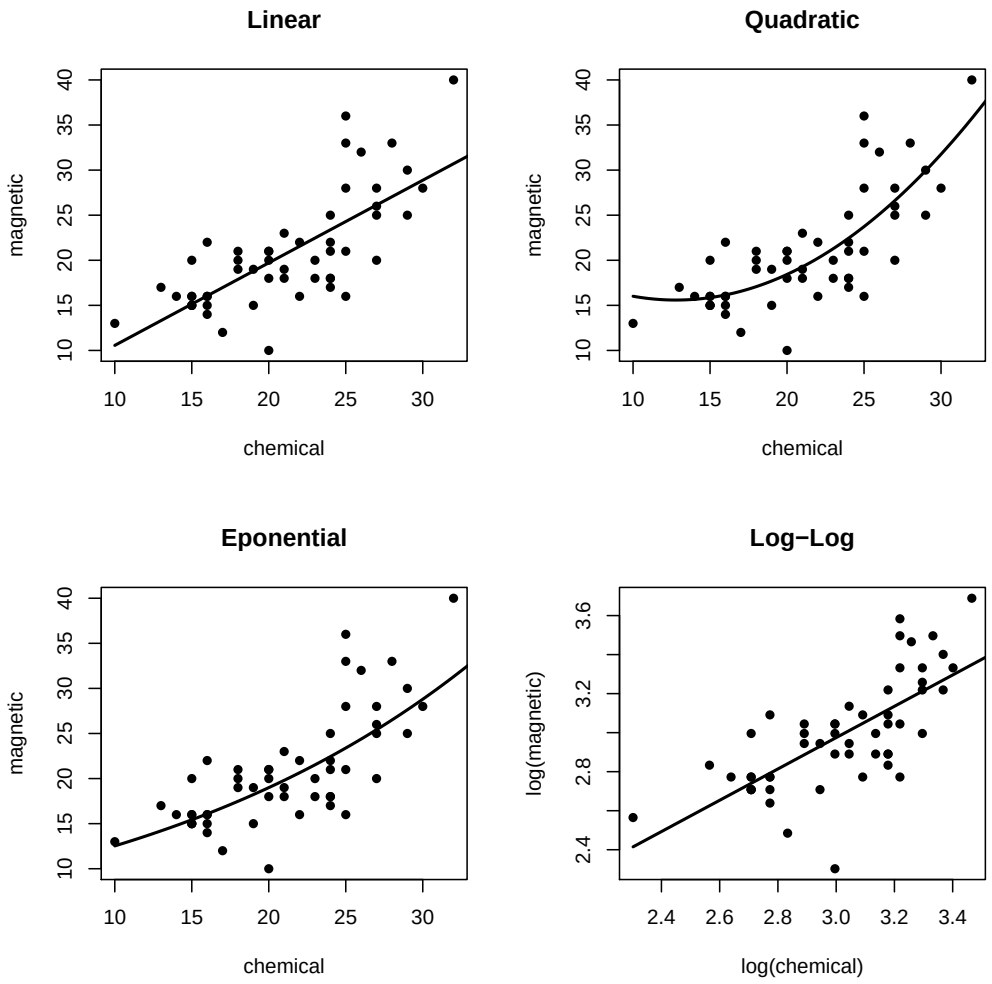


Figure 1.10: Fitting of four different models for iron slag data

1.12.1 Introduction to `optim()` of R

The function `optim(par,fn,hessian=TRUE,...)` is meant for nonlinear unconstrained optimization (preferarably minimization). It requires guess value of the parameter (argument `par`) and objective function $-l(\theta)$. The argument `hessian=TRUE` is required to return *Hessian* matrix at the optimum point. Three dots $(\cdots)$ at the end in arguments stand for arguments which are not specified.

1.12.2 Fitting of exponential distribution

Note that

$$f(y|\lambda) = \lambda e^{-\lambda y} = dexp(y, rate = \lambda)$$

This implies that loglikelihood for exponential with $rate = \lambda$ is

$$l_i(\lambda) = log(f(y|\lambda)) = log(\lambda) - \lambda y_i \quad for\ i = 1, \cdots, n,$$

This implies that

$$l(\lambda) = \Sigma l_i(\lambda) = \Sigma_{i=1}^{i=n}(log(\lambda) - \lambda y_i) = sum(dexp(y, \lambda, log = TRUE))$$

```
# fit exponential distribution
set.seed(1)
y<-rexp(20,rate=0.03) # simulate 20 random observations from exponential distri
y

##  [1]  25.172728  39.388093   4.856891   4.659842
##  [5]  14.535621  96.498951  40.985402  17.989428
##  [9]  31.885583   4.901533  46.357838  25.400995
## [13]  41.253452 147.464474  35.151439  34.508132
## [17]  62.534506  21.824888  11.231116  19.615991

# define negative of loglikelihood of exponential distribution
nlle<-function(theta,y){
  ll<-dexp(y,rate=theta,log=TRUE)
  ll<-sum(ll)
  return(-ll)
}
dump("nlle",file="nlle.txt") #save it
# Fitting with optim
M1<-optim(par=0.01,fn=nlle,hessian=TRUE,method="L-BFGS-B",lower=0.001,y=y)
M1
```

```
## $par
## [1] 0.02755197
##
## $value
## [1] 91.84233
##
## $counts
## function gradient
##       25       25
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
##
## $hessian
##          [,1]
## [1,] 26416.26
```

```
MLE<-M1$par #ML estimator
se<-sqrt(diag(solve(M1$hessian))) # SE of thetahat
MLE
```

```
## [1] 0.02755197
```

```
se
```

```
## [1] 0.00615268
```

```
out<-rbind(MLE,se)
colnames(out)<-c("rate")
out
```

```
##           rate
## MLE 0.02755197
## se  0.00615268
```

1.12.3 Fitting of Weibull model

Weibull distribution is a very popular distribution for fitting of reliability and survival data. It has a *shape* and a *scale* parameter. It is fitted as:

```
# define a function which returns negative of lolglikelihood of weibull.
set.seed(1)
y<-rweibull(30,shape=1.5,scale=50) # simulate 30 observations
##define -logliklihood of Weibull,-l(theta), theta=c(shape,scale)

nllw<-function(theta,y){
  ll<-dweibull(y,shape=theta[1],scale=theta[2],log=TRUE)
  ll<-sum(ll)
  return(-ll)
}
dump("nllw",file="nllw.txt") #save it
##Fit it
M2<-optim(par=c(1,30),fn=nllw,hessian=TRUE,method="L-BFGS-B",
          lower=c(0.1,2),y=y)
MLE=M2$par
se=sqrt(diag(solve(M2$hessian)))
out<-rbind(MLE,se)
colnames(out)<-c("shape","scale")
out

##         shape     scale
## MLE 1.4892233 48.163083
## se  0.2156091  6.206236
```

It may be noted that fitted values are in close agreement with the true values, which shows the strength of fitting procedure.

1.12.4 Assignment

Describe this fitting of Weibull, graphically.

1.12.5 Fitting with `fitdistr()` of MASS

A very powerful function `fitdistr()` is available with the **MASS** package. A large number of distributions can be fitted using this function. This includes, normal, Poisson, t, gamma etc. It can be reported as `fitdistr(x,densfun,start,...)`, where x stands for the data vector to be fitted, $densfun$ stands for the name

of the density to be fitted, *start* stands for guess vales to start the iterations. Let us begin with *gamma* distribution:

```
library(MASS)
set.seed(123)
x <- rgamma(100, shape = 5, rate = 0.1)
fitdistr(x, "gamma")

##      shape       rate
##   6.4869913   0.1365106
##  (0.8946010) (0.0195730)

## now do this directly with more control.
fitdistr(x, dgamma, list(shape = 1, rate = 0.1), lower = 0.001)

##      shape         rate
##   6.48686551   0.13651012
##  (0.89438948) (0.01956818)
```

1.12.6 Assignments

1. Fit gamma distribution using `optim()` as discussed for Weibull and compare your results with the results obtained using `fitdistr()`.

2. Simulate 30 observations from a Poisson distribution with mean 10 and fit that using `fitdistr()`.

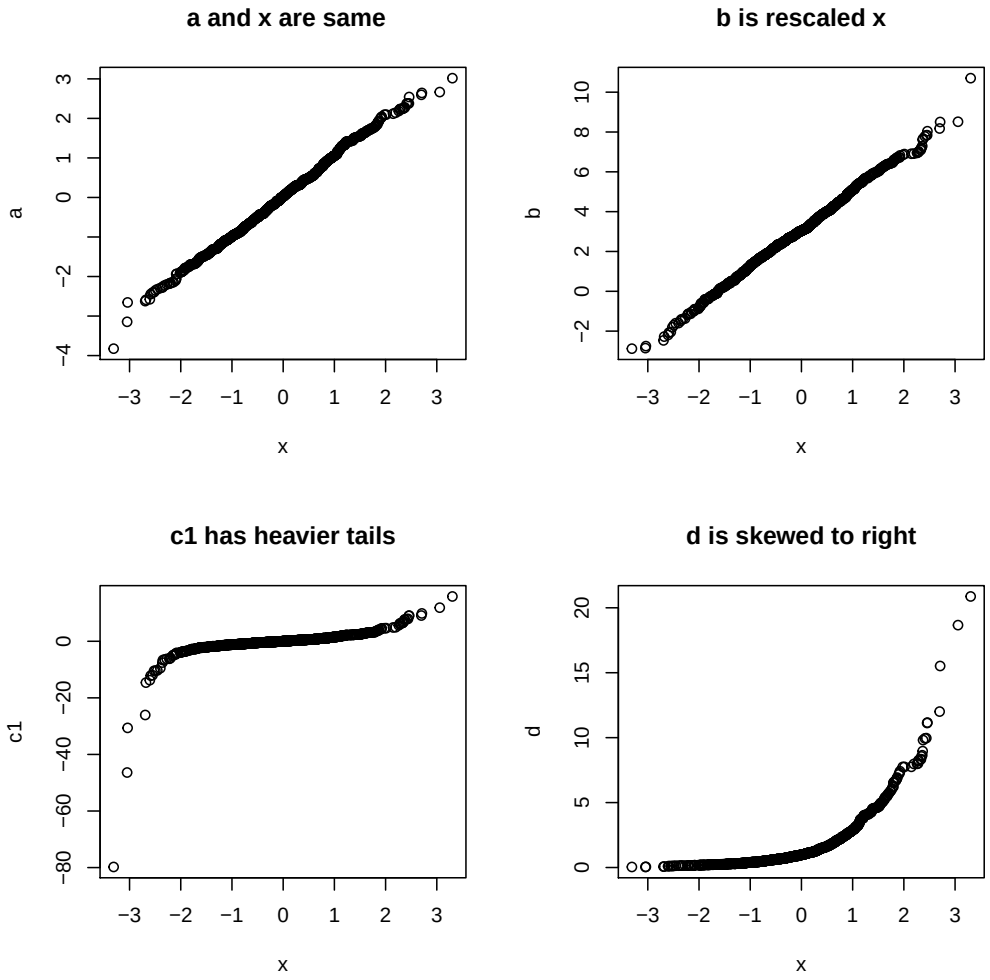
1.13 Quantile Quantile plots

Quantile-quantile plots (known as QQ plots) are a type of scatter plot used to compare the distributions of two groups or to compare a sample with a reference distribution. The line $y = x$ implies both are same.

1.13.1 The function `qqplot()`

The **R** function `qqplot()` is meant for graphic comparisons of the two data sets or one data set with a reference distribution.

```r
# compare two standard normal distributions
x<-rnorm(1000) #x~N(0,1)
a<-rnorm(1000)
opar<-par(mfrow=c(2,2))
qqplot(x,a,main=" a and x are same")
#comapre N(0,1) with N(3,2)
b<-rnorm(1000,mean=3,sd=2) #b~N(mean=3,sd=2)
qqplot(x,b,main="b is rescaled x")
#Compare Student's t with df=2 with N(0,1)
c1<-rt(1000,df=2)
qqplot(x,c1,main="c1 has heavier tails")
#Compare lognormal with normal
d<-exp(rnorm(1000)) #simulation from lognormal
qqplot(x,d,main="d is skewed to right")
```




```
par(opar) #return to original settings
```

Note that in the last two figures, distributions are not matching and line $y = x$ is not satisfied. Moreover, if you want to compare two real data vectors x and y, they can be compared in the same manner.

1.13.2 Exercise: qqplot

Compare the distribution height and weight of students of B Sc Final, using `qqplot()`.

1.13.3 Normal probability plot—qqnorm()

The function `qqnorm` compares Normal distribution with a data set. A reference line can also be added by `qqline()`.

```
opar<-par(mfrow=c(1,2))
x<-rnorm(1000)
qqnorm(x)
qqline(x)
xe<-rexp(1000,rate=800)
qqnorm(xe,main="Exponential with rate=800")
qqline(xe)
```

```
par(opar)
```

Note that data *xe* which follows exponential distribution is not matching with the normal distribution. This is evidenced from the fact that in case of exponential, reference line is not matching with plot.

1.14 Import Export

R has very powerful interface with other systems. One can import data from other packages by using the package **foreign**. Similarly, one can save output to the systems too by using the function `write.foreign()`. A set of commands follow:

```
library(foreign)
datafile<-tempfile()
codefile<-tempfile()
write.foreign(trees,datafile,codefile,package="SPSS")
file.show(datafile)
file.show(codefile)
```

1.14.1 Reading data from SPSS

We can read a data from **SPSS** into **R** very easily as:

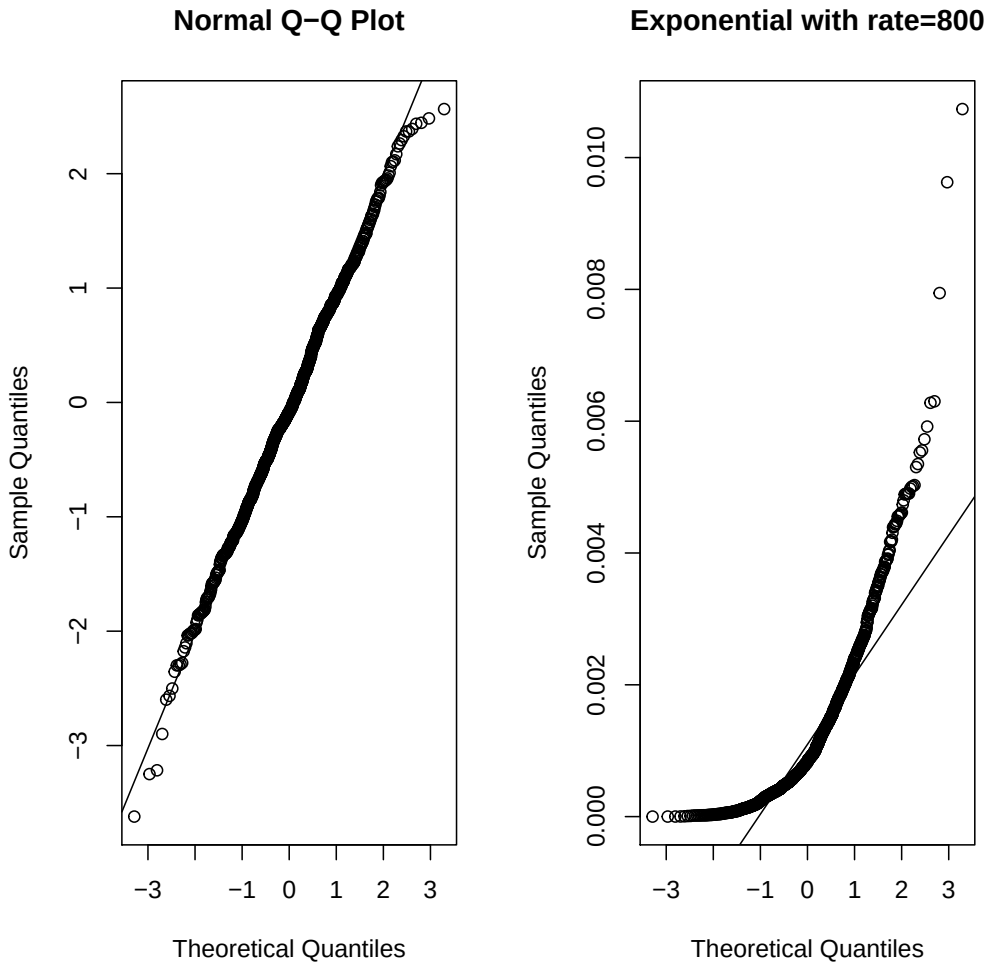


Figure 1.11: Normal probability plot.

```r
sav<-system.file("files","electric.sav",package="foreign")
dat<-read.spss(file=sav)
str(dat) #it is a listed structure
```

```
## List of 13
##  $ CASEID  : num [1:240] 13 30 53 84 89 102 117 132 151 153 ...
##  $ FIRSTCHD: Factor w/ 5 levels "NO CHD","SUDDEN  DEATH",..: 3 3 2 3 2 3 3 3
##  $ AGE     : num [1:240] 40 49 43 50 43 50 45 47 53 49 ...
##  $ DBP58   : num [1:240] 70 87 89 105 110 88 70 79 102 99 ...
##  $ EDUYR   : num [1:240] 16 11 12 8 NA 8 NA 9 12 14 ...
##  $ CHOL58  : num [1:240] 321 246 262 275 301 261 212 372 216 251 ...
##  $ CGT58   : num [1:240] 0 60 0 15 25 30 0 30 0 10 ...
##  $ HT58    : num [1:240] 68.8 72.2 69 62.5 68 68 66.5 67 67 64.3 ...
##  $ WT58    : num [1:240] 190 204 162 152 148 142 196 193 172 162 ...
##  $ DAYOFWK : Factor w/ 8 levels "SUNDAY","MONDAY",..: 8 5 7 4 2 1 8 1 3 5 ...
##  $ VITAL10 : Factor w/ 2 levels "ALIVE","DEAD": 1 1 2 1 2 2 1 1 2 2 ...
##  $ FAMHXCVR: Factor w/ 2 levels "NO","YES": 2 1 1 2 1 1 1 1 1 2 ...
##  $ CHD     : num [1:240] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "label.table")=List of 13
##   ..$ CASEID  : NULL
##   ..$ FIRSTCHD: Named num [1:5] 6 5 3 2 1
##   .. ..- attr(*, "names")= chr [1:5] "OTHER   CHD" "FATAL   MI" "NONFATALMI"
##   ..$ AGE     : NULL
##   ..$ DBP58   : NULL
##   ..$ EDUYR   : NULL
##   ..$ CHOL58  : NULL
##   ..$ CGT58   : NULL
##   ..$ HT58    : NULL
##   ..$ WT58    : NULL
##   ..$ DAYOFWK : Named num [1:8] 9 7 6 5 4 3 2 1
##   .. ..- attr(*, "names")= chr [1:8] "MISSING" "SATURDAY" "FRIDAY" "THURSDAY"
##   ..$ VITAL10 : Named num [1:2] 1 0
##   .. ..- attr(*, "names")= chr [1:2] "DEAD" "ALIVE"
##   ..$ FAMHXCVR: Named chr [1:2] "Y       " "N       "
##   .. ..- attr(*, "names")= chr [1:2] "YES" "NO"
##   ..$ CHD     : NULL
##  - attr(*, "variable.labels")= Named chr [1:13] "CASE IDENTIFICATION NUMBER"
##   ..- attr(*, "names")= chr [1:13] "CASEID" "FIRSTCHD" "AGE" "DBP58" ...
##  - attr(*, "missings")=List of 13
##   ..$ CASEID  :List of 1
##   .. ..$ type: chr "none"
```

```
##   ..$ FIRSTCHD:List of 1
##   .. ..$ type: chr "none"
##   ..$ AGE     :List of 1
##   .. ..$ type: chr "none"
##   ..$ DBP58   :List of 1
##   .. ..$ type: chr "none"
##   ..$ EDUYR   :List of 1
##   .. ..$ type: chr "none"
##   ..$ CHOL58  :List of 1
##   .. ..$ type: chr "none"
##   ..$ CGT58   :List of 1
##   .. ..$ type: chr "none"
##   ..$ HT58    :List of 1
##   .. ..$ type: chr "none"
##   ..$ WT58    :List of 1
##   .. ..$ type: chr "none"
##   ..$ DAYOFWK :List of 2
##   .. ..$ type : chr "one"
##   .. ..$ value: num 9
##   ..$ VITAL10 :List of 1
##   .. ..$ type: chr "none"
##   ..$ FAMHXCVR:List of 1
##   .. ..$ type: chr "none"
##   ..$ CHD     :List of 1
##   .. ..$ type: chr "none"
```

```
datF<-read.spss(file=sav,to.data.frame = TRUE)
str(datF) # now it is a data frame
```

```
## 'data.frame': 240 obs. of  13 variables:
##  $ CASEID  : num  13 30 53 84 89 102 117 132 151 153 ...
##  $ FIRSTCHD: Factor w/ 5 levels "NO CHD","SUDDEN  DEATH",..: 3 3 2 3 2 3 3 3
##  $ AGE     : num  40 49 43 50 43 50 45 47 53 49 ...
##  $ DBP58   : num  70 87 89 105 110 88 70 79 102 99 ...
##  $ EDUYR   : num  16 11 12 8 NA 8 NA 9 12 14 ...
##  $ CHOL58  : num  321 246 262 275 301 261 212 372 216 251 ...
##  $ CGT58   : num  0 60 0 15 25 30 0 30 0 10 ...
##  $ HT58    : num  68.8 72.2 69 62.5 68 68 66.5 67 67 64.3 ...
##  $ WT58    : num  190 204 162 152 148 142 196 193 172 162 ...
##  $ DAYOFWK : Factor w/ 7 levels "SUNDAY","MONDAY",..: NA 5 7 4 2 1 NA 1 3 5 .
##  $ VITAL10 : Factor w/ 2 levels "ALIVE","DEAD": 1 1 2 1 2 2 1 1 2 2 ...
```

```
##  $ FAMHXCVR: Factor w/ 2 levels "NO","YES": 2 1 1 2 1 1 1 1 1 2 ...
##  $ CHD     : num  1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "variable.labels")= Named chr [1:13] "CASE IDENTIFICATION NUMBER"
##   ..- attr(*, "names")= chr [1:13] "CASEID" "FIRSTCHD" "AGE" "DBP58" ...
```

```
head(datF)
```

```
##   CASEID      FIRSTCHD AGE DBP58 EDUYR CHOL58 CGT58
## 1     13    NONFATALMI  40    70    16    321     0
## 2     30    NONFATALMI  49    87    11    246    60
## 3     53 SUDDEN  DEATH  43    89    12    262     0
## 4     84    NONFATALMI  50   105     8    275    15
## 5     89 SUDDEN  DEATH  43   110    NA    301    25
## 6    102    NONFATALMI  50    88     8    261    30
##   HT58 WT58  DAYOFWK VITAL10 FAMHXCVR CHD
## 1 68.8  190     <NA>   ALIVE      YES   1
## 2 72.2  204 THURSDAY   ALIVE       NO   1
## 3 69.0  162 SATURDAY    DEAD       NO   1
## 4 62.5  152 WEDNSDAY   ALIVE      YES   1
## 5 68.0  148   MONDAY    DEAD       NO   1
## 6 68.0  142   SUNDAY    DEAD       NO   1
```

```
tail(datF)
```

```
##     CASEID FIRSTCHD AGE DBP58 EDUYR CHOL58 CGT58 HT58
## 235    147   NO CHD  45   105    14    200    30 66.5
## 236    148   NO CHD  49    87    15    169    20 69.0
## 237    149   NO CHD  54    69     9    207    20 64.5
## 238    150   NO CHD  42    71    16    178     0 69.8
## 239    154   NO CHD  53    77    18    225    10 71.3
## 240    155   NO CHD  47    83    NA    206     0 66.0
##     WT58 DAYOFWK VITAL10 FAMHXCVR CHD
## 235  171  FRIDAY   ALIVE       NO   0
## 236  193    <NA>   ALIVE       NO   0
## 237  136    <NA>   ALIVE       NO   0
## 238  172    <NA>   ALIVE       NO   0
## 239  175    <NA>   ALIVE       NO   0
## 240  185    <NA>   ALIVE       NO   0
```

1.14.2 Reading from stata

One can read and a `stata` file into **R**. An illustration is made with 'swiss' data available with the package **datasets**. The functions 'read.dta()' and 'write.dta()'.

```
head(swiss) #see the header part
```

```
##              Fertility Agriculture Examination
## Courtelary        80.2        17.0          15
## Delemont          83.1        45.1           6
## Franches-Mnt      92.5        39.7           5
## Moutier           85.8        36.5          12
## Neuveville        76.9        43.5          17
## Porrentruy        76.1        35.3           9
##              Education Catholic Infant.Mortality
## Courtelary          12     9.96             22.2
## Delemont             9    84.84             22.2
## Franches-Mnt         5    93.40             20.2
## Moutier              7    33.77             20.3
## Neuveville          15     5.16             20.6
## Porrentruy           7    90.57             26.6
```

```
tail(swiss) #see the tail part
```

```
##              Fertility Agriculture Examination
## Neuchatel         64.4        17.6          35
## Val de Ruz        77.6        37.6          15
## ValdeTravers      67.6        18.7          25
## V. De Geneve      35.0         1.2          37
## Rive Droite       44.7        46.6          16
## Rive Gauche       42.8        27.7          22
##              Education Catholic Infant.Mortality
## Neuchatel           32    16.92             23.0
## Val de Ruz           7     4.97             20.0
## ValdeTravers         7     8.65             19.5
## V. De Geneve        53    42.34             18.0
## Rive Droite         29    50.43             18.2
## Rive Gauche         29    58.33             19.3
```

```
str(swiss) #see the structure of the data
```

```
## 'data.frame': 47 obs. of  6 variables:
##  $ Fertility       : num  80.2 83.1 92.5 85.8 76.9 76.1 83.8 92.4 82.4 82.9 .
##  $ Agriculture     : num  17 45.1 39.7 36.5 43.5 35.3 70.2 67.8 53.3 45.2 ...
##  $ Examination     : int  15 6 5 12 17 9 16 14 12 16 ...
##  $ Education       : int  12 9 5 7 15 7 7 8 7 13 ...
##  $ Catholic        : num  9.96 84.84 93.4 33.77 5.16 ...
##  $ Infant.Mortality: num  22.2 22.2 20.2 20.3 20.6 26.6 23.6 24.9 21 24.4 ...

#write the data frame `swiss` to a stata file `swiss2.dta`
require(foreign)
write.dta(swiss,file="swiss2.dta")
# Read this data into R as data frame
swiss2<-read.dta("swiss2.dta")
head(swiss2)

##   Fertility Agriculture Examination Education
## 1      80.2        17.0          15        12
## 2      83.1        45.1           6         9
## 3      92.5        39.7           5         5
## 4      85.8        36.5          12         7
## 5      76.9        43.5          17        15
## 6      76.1        35.3           9         7
##   Catholic Infant_Mortality
## 1     9.96             22.2
## 2    84.84             22.2
## 3    93.40             20.2
## 4    33.77             20.3
## 5     5.16             20.6
## 6    90.57             26.6
```

The best strategy is that you download the '.dta' file into the working folder and then use the function 'read.dta()' as we did above.

1.14.3 Reading data from Excel

MSExcel is a very popular spread sheet software for entering data. Most of the users come with the data entered into **Excel** and want data analysis done in **R**. Following steps are required:

`1. Enter your data into Excel's spread sheet.`

2. Save it as comma delimited values, say "D1.csv".

3. Read it into R as: D1<-read.csv("D1.csv")

4. Save it into R as: write.csv(D1,file="D2.csv",row.names=F)

Note that when you enter data into Excel's spread sheet, the first row should be the variable names. These names should not be fractured names. For example `Plant Height` is a fractured nams, it should be `PlantHeight`. As an example, consider the following data need be entered:

```
Height Weight Gender
165     55    Male
160     52    Female
178     70    Male
166     65    Male
169     60    Male
163     58    Female
```

We will enter into Excel and save it as `D1.csv` file, and then read it by using data frame object `D1`.

```
D1<-read.csv("D1.csv")
D1
```

```
##   Height Weight Gender
## 1    165     55   Male
## 2    160     52 Female
## 3    178     70   Male
## 4    166     65   Male
## 5    169     60   Male
## 6    163     58 Female
```

```
write.csv(D1,file="D2.csv",row.names=FALSE)
D2<-read.csv("D2.csv")
D2
```

```
##   Height Weight Gender
## 1    165     55   Male
## 2    160     52 Female
## 3    178     70   Male
## 4    166     65   Male
## 5    169     60   Male
## 6    163     58 Female
```

```
# get summary of data
summary(D2)

##      Height          Weight         Gender
##  Min.   :160.0   Min.   :52.00   Length:6
##  1st Qu.:163.5   1st Qu.:55.75   Class :character
##  Median :165.5   Median :59.00   Mode  :character
##  Mean   :166.8   Mean   :60.00
##  3rd Qu.:168.2   3rd Qu.:63.75
##  Max.   :178.0   Max.   :70.00
```

1.14.4 Reading a data from internet

One can read a data which is available on internet and its path is `"https://goo.gl/UDv12g"`. One can read this data as:

```
DF1<-read.csv("https://goo.gl/UDv12g")
head(DF1)

##   iProdSAT iSalesSAT Segment iProdREC iSalesREC
## 1        6         2       1        4         3
## 2        4         5       3        4         4
## 3        5         3       4        5         4
## 4        3         3       2        4         4
## 5        3         3       3        2         2
## 6        4         4       4        5         4
```

This a sales data. Similarly, a data available with `"http://www.jaredlander.com/data/Tomato%20First.csv"` can be read into **R** as:

```
theUrl<-"http://www.jaredlander.com/data/Tomato%20First.csv"
tomato<-read.csv(file=theUrl)
head(tomato)

##   Round             Tomato Price      Source Sweet
## 1     1          Simpson SM  3.99 Whole Foods   2.8
## 2     1  Tuttorosso (blue)  2.99     Pioneer   3.3
## 3     1 Tuttorosso (green)  0.99     Pioneer   2.8
## 4     1     La Fede SM DOP  3.99   Shop Rite   2.6
```

```
## 5     2       Cento SM DOP  5.49  D Agostino   3.3
## 6     2      Cento Organic  4.99  D Agostino   3.2
##   Acid Color Texture Overall Avg.of.Totals
## 1  2.8   3.7     3.4     3.4          16.1
## 2  2.8   3.4     3.0     2.9          15.3
## 3  2.6   3.3     2.8     2.9          14.3
## 4  2.8   3.0     2.3     2.8          13.4
## 5  3.1   2.9     2.8     3.1          14.4
## 6  2.9   2.9     3.1     2.9          15.5
##   Total.of.Avg
## 1         16.1
## 2         15.3
## 3         14.3
## 4         13.4
## 5         15.2
## 6         15.1

#If we want to use read.table(), then
tomato1<-read.table(file=theUrl,header = TRUE,sep=",")
head(tomato1,n=3)

##   Round             Tomato Price      Source Sweet
## 1     1         Simpson SM  3.99 Whole Foods   2.8
## 2     1  Tuttorosso (blue)  2.99     Pioneer   3.3
## 3     1 Tuttorosso (green)  0.99     Pioneer   2.8
##   Acid Color Texture Overall Avg.of.Totals
## 1  2.8   3.7     3.4     3.4          16.1
## 2  2.8   3.4     3.0     2.9          15.3
## 3  2.6   3.3     2.8     2.9          14.3
##   Total.of.Avg
## 1         16.1
## 2         15.3
## 3         14.3
```

Note that when you run these commands, your system must be connected with internet. This is the requirement for generating the documents also. Alternatively, you can save these data set into your working directory and use `read.csv()` to read them.

Chapter 2

Hypothesis testing using R

```
z-test
y~N(mu,sigma^2)
H0: mu=mu0=140
H1: mu not eqaul to 140.
 and sigma^2 is known, the test criteria is z-test. Hwoever, if sigma^2 is not k
z=(ybar-mu0)/(sigma/sqrt(n))~N(0,1)
y<-c(168,170,185,172,169,158,170,175)
sigma=5
Since sigma is known, we have to use z-test.
ztest=(mean(y)-140)/5/sqrt(length(y))
qnorm(.05/2,lower.tail=FALSE) # tabulated value
1.If calculated value is greater than tabulated, we reject H0.
2. If p-value is less than 0.05, we reject H0.
pvalue<-2*(pnorm(abs(z),lower.tail=FALSE))
pvalue=2.62222e-68
which is less than 0.05, hence we reject H0.

3. If sigma is unknown, then we use t-test.

t=(mean(y)-140)/(sd(y)/sqrt(length(y)))
It calculated t is greater thn tabulated t, we reject H0
qt(0.05/2,df=length(y)-1,lower.tail=FALSE) # tabulated value

However, if pvalue is less than 0.05, we reject H0.
pvalue<-2*(pt(abs(t),df=length(y)-1,lower.tail=FALSE))

## ttest for two samples
## t-test for comparing means of two normal populations
```

```
## If y1~N(mu1,sigma1^2)
and y2~N(mu2,sigma2^2)
then to test H0: mu1=mu2
vs
             H1:mu1!=mu2

provided sigma1^2=sigma2^2
then test criteria is t-test

t=(mean(y1)-mean(y2))/(sp*sqrt(1/n1+1/n2)~Student_t(n1-1+n2-1)
where
sp^2=((n1-1)s1^2+(n2-1)s2^2)/((n1-1)+(n2-1))
sp=sqrt(sp^2)
We reject H0 if |t|>qt(alpha/2,df=(n1-1+n2-1),lower.tail=FALSE)
or we reject H0 if
pvalue=2*(pt(abs(t),df=n1-1+n2-1,lower.tail=FALSE))
is less than 0.05.

## write a function to implement t-test for two samples
twot<-function(y1,y2){
#ttest  for two samples
#assume sigma1^2=sigma2^2
# y1 and y2 are two data vectors
n1<-length(y1)
n2<-length(y2)
ybar1=mean(y1)
ybar2=mean(y2)
var1=var(y1)
var2=var(y2)
dferror=n1+n2-2
sp2=((n1-1)*var1+(n2-1)*var2)/dferror
t=(ybar1-ybar2)/sqrt(sp2*(1/n1+1/n2))
pvalue=2*(pt(abs(t),dferror,lower.tail=FALSE))
out<-list(mean1=ybar1,mean2=ybar2,var1=var1,var2=var2,t=t,pvalue=pvalue)
return(out)
}
dump("twot",file="twot.txt")

## Take a data
set.seed(1)
y1<-rnorm(10,8,2) # 10 observations from N(8,sd=2)
```

```
set.seed(1)
y2<-rnorm(12,20,3)# 12 observations from N(20,sd=3)
#Use twot
twot(y1=y1,y2=y2)
## We reject H0 as pvalue is less than 0.05.
## This means on average menas of two populations differ significantly.

## Exercise:
1. Write a function for one sample t-test like above. Use that to test a data di

## Moore D F, Mc Cabe, and Craig(2008): Introduction to the Practice of Statisti
```