A Short Course in Statistics with R*

Prabhanjan N T[1], Suresh R[2], and Manjunath B G[3]

1 CustomerXPs, Bangalore

2 Department of Statistics, Bangalore University, Bangalore

3 FB Mathematik, Universität Siegen, Siegen, Germany

Version 0.1

* Based on a workshop conducted for M.Sc III students conducted on 23 and 30 August, 2008

CONTENTS

ACKNOWLEDGEMENTS

Prof. P. Rajalakshmi : for providing us all the facilities in the Department of Statistics, Bangalore University with lot of cooperation and smiles

Prof. J.V. Janhavi : encouraged us through each horror sessions of the workshop

Sri. Shakun Gupta : for having accepted and delivered a very useful talk on the importance of Open Source Software on 23 August, 2008

Profs. T. Srivenkataramana, S.M. Manjunath, K. Harishchandra, G. Nanjundan, V. Srinivas, and P.V. Pandit : for a very useful masters course and beyond that too

DEDICATION

Prof. H.J. Vaman

for being the most important lamp

Notes

We have made an attempt to keep this short course as self-contained as possible. Each chapter lists, in the beginning, all the data sets and packages used in it. Also accompanied with the short course is the file "R_Codes_for_ASCSWR.txt", where ASCSWR is abbreviation of the title of this book "A Short Course in Statistics With R". Further, we compliment this book with a zip file "Data_Sets.zip" which contains all the data sets used in it.

Remark about the R codes. The book naturally contains lots of R codes. To distinct it from the flow of text, the spacings have been reduced to "single-line" spacing, whereas the text has "one-and-half-line" spacing. Further, the R code to be entered in the console are in blue color while the resulting output are in black color. We hope that this enhances the readability of the text. Thanks.

1. Why R?

Package(s): base

Data set(s) :

1.1 Why R?

In his Presidential address to the 37th Indian Science Congress, 1950, the legendary Prasanta Chandra Mahalanobis delivered one of the most important essays in the annals of statistics, namely, "Why Statistics?" Some of the key-points in that essay addresses issues such as statistics as a legitimate branch of science, the concept of random samples, importance of random sampling, etc. This essay has been reprinted in the monthy journal *Resonance,* 1999, published by the Indian Academy of Sciences, and makes a very pleasant reading even today. Sixty years later we ask ourselves the question "Why R?" Of course, the magnitude of the question is oriented in a completely different and (probably) insignificant way, and we hope the reader will excuse us for this idiosyncracy. Whereas Prof. Mahalanobis put that question to the Indian scientific community the importance of Statistics as a main stream, and as such our perspective for asking this question about R is a bit different and to put more emphasis on the use of this Open Source software which has been mainly developed by Statisticians. Unlike the subject which deals about uncertainty arising in most of the real experiments, R is not the only alternative for the purpose of the related statistical computations. The natural question that arises is about the main advantages of this software package over the available ones.

R is easily one of the fastest growing software in the academics and industry alike. It is worthwhile to note that many applied statistics books are being written with computational programs being developed in R. The software itself is accompanied with very useful notes written by Venables and Ripley, who have themselves contributed to it from its infant stages. Dalgaard (2002, 2008) is perhaps the first introductory stat book blended with R, and is arguably the best place to start with. Faraway (2002) wrote one of the earliest set of notes, this can be freely circulated without any restrictions on copyrights, with a more dedicated focus on Regression and ANOVA. It is perhaps impudent to note that the publishing house, Springer, has started an exclusive series on books written with R software for computational genesis which has been labeled as "Use R!" From a regional point of view, we also welcome the recent book by Deshmukh, Purohit, and Gore (2008).

The reader will be more aware of the subject-specific R books as she transgresses through the rest of this set of notes.

Strengths of R. "Free software" is one of the easily understandable merit of R.

Some limitations. R is slow in running loops**. Also, in comparison with other software, the help files are not friendly.**

A Non-technical note: Most of the commands/programs have been written and executed on the R version installed on Ubuntu OS. R has been invoked from the terminal, and as such the presentation in these notes is from this perspective.

1.2 R Installation

The web-site http://cran.r-project.org/ consists all versions of R available for a variety of Operating Systems. CRAN stands for Comprehensive R Archive Network. A incidental fact is that R had been developed on Internet only.

The **R** software can be installed on a variety of platforms such as Linux, Windows, and Macintosh, among others.

http://ubuntuforums.org/showthread.php?t=639710 (for linux ubuntu users)

http://freshmeat.net/articles/view/2237/

1.3 Data Sets in R and Internet

R consists of many data sets and more often than not each package contains many data sets. The command *try(data(package= "")*) enlists all the data sets contained in that package. For example, if we need to find the data sets in the package, say *repolr,* execute the following:

```
> try(data(package="repolr"))
        HHSpain              Harris Hip Pain Scores
        achilles             Achilles Tendon Rupture
> try(data(package="gee"))
        no data sets found
```

The function for reading these data sets will be given in the next chapter. It has also been observed that authors of many book have created packages containing all the data sets from their book and released it for the benefit of the programmers. For example, Faraway (2002) and Everitt and

Hothorn (2006) have created packages titled “faraway” and “HSAUR” respectively, which may be easily downloaded from http://cran.r-project.org/web/packages/.

Another major reason for a student to familiarise herself with a software is that rarely do practical settings have small data sets (n < 100, to be more precise). The last time we verified, all the web-sites listed below contained real/practical data sets. This era really requires the statistician to shy away with the ordinary calculators and flirt as much as she can with the real data sets. This is probably one of the few occasions where the *janta* does not mind more of extrovertism.

List of Web-sites Containing DATA SETS

Practical data sets are available in plenty on world wide web. It is impossible for anybody to give an exhaustive list of all the web-sites containing data sets, and we have just listed below what we have found useful and convinced that the user will benefit from the web-sites. The list is also not in any particular order of priorities.

http://lib.stat.cmu.edu/datasets/

http://www.blackwellpublishing.com/rss/

http://www.commondataset.org/

http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/ (for time series)

http://inforumweb.umd.edu/econdata/econdata.html

http://www.ucsd.edu/portal/site/Libraries/menuitem.346352c02aac0c82b9ba4310d34b01ca/?vgnextoid=e085d2905bd14110VgnVCM10000045b410acRCRD

http://www.amstat.org/publications/jse/information.html

http://www.statsci.org/datasets.html

http://exploringdata.cqu.edu.au/datasets.htm

http://www.amstat.org/publications/jse/datasets/rawlings/index.html

http://archive.ics.uci.edu/ml/datasets.html

We are positive that the above list will benefit the user and also encourage her to find more such sites according to her requirements.

1.4 cran.r-project.org

The website is the source of all versions of R across all the platforms such as Windows, Linux, Mac. The current version of R is 2.10.0. We have observed that the software releases improved

version every three months.

The sites page **http://cran.r-project.org/web/views/** contains various branches of the subject for which various add-on packages are available.

We strongly recommend to the user subscribtion of the R-Mailing list at www.r-project.org/mail.html. It is indeed possible for a user that even after years of mastering R, she may not yet be aware of many of the important methods of the software. Moreover, there is no point in struggling with a problem for days together when the help is there just a mail away. We do believe in the wise words of Samuel Johnson who once said, "There are two types of knowledge. One is knowing a thing. The other is knowing where to find it." The R-Mailing list is the other type of knowledge.

The entire R core development team is actively involved in replying all the queries, a partial list consists of the names given at http://www.r-project.org/contributors.html. Apart from them, any person on the mailing list can also answer the query, and thereby increasing the probability that your querry is answered. In the next section, we statistically show how useful this mailing list can be.

1.5 Explosion of R Books in the Literature

Thanks to the user-friendliness of the software, many books are available in the literature with a "R-specific" focus. The first manual which deserves a mention is the notes of Venables and Ripley (2006), the first version of which came probably in 1997. Such is the importance of these notes that it comes along with the R software and may be easily assessed. Its very readable and lucid in flow and we have defenitely benefitted by it. Without demeaning all those who have written on R, we can probably say that every author is guilty of repeating Venables and Ripley and have thereby greatly reduced our guilt of the same.

Dalgaard (2002) is probably the first exclusive book on the software, and its simplicity is amazing. Though Dalgaard has come with a second edition (released just this month, 2008), the spirit and value is nevertheless unchanged. A freely circulated notes on Regression and ANOVA using R is due to Faraway (2002). Faraway has promptly followed these set of notes with two books, Faraway (2006) and Faraway (2006). Maindonald and Braun (2003) is an early exposition to data analysis methods and graphics. Crawley (2007) book on R, a 950 page length encylopaedic book, covers many of the topics and may also be used to develop a 6-packs.

It is important to note here that though Nolan and Speed (2000) have not written in the R-text book mould, they have developed very many R programs.

* * * * *

2. The R BASICS

Package(s) : base, MASS, gdata, Matrix, stats

Data set(s) :

There is an intrinsic need to explain the way R works. A major programming problem is the failure (the unawareness) of the knowledge about how the functions work. A simple way to know the structure of a function is to simply key in the command at the R console. For example, if one needs to know the constitution of the command "*lm*", simply keying in "lm" at the console will give its constitution.

In this chapter, as far as possible, we have attempted to familarise the reader with basic R functions which we believe are useful towards data preparation and will provide a strong footing for further data manipulation.

The first thing that we will change about the boring console ">" is by replacing it with "CXPS > ":

```
> options(prompt="CXPS > ") # results in a new prompt as below
```

2.1 Simple Arithmetics

Basic mathematical operations can be directly performed at the console ">". Addition, substraction, mulitiplication, division, and exponentiation can be performed in the routine way as in calculators as given below:

```
> 57 + 89
[1] 146
> 45 - 87
[1] -42
> 60 * 3
[1] 180
> 7CXPS /18
[1] 0.3888889
> 4^4
[1] 256
```

Two additional operators that we will find useful for two objects *x* and *y* are x %% y and x %/% y, where the operator "%%" computes "*x* mod *y*", and "%/%" computes integer division:

```
> 5%%2
[1] 1
```

```
> 5%/%2
[1] 2
```

The above mentioned operators are stored in “Arithmetic” part of the *base* package.

Exponentiation, logarithm, trignometric transformations, etc, may be easily performed:

```
> sqrt(100)
[1] 10
> exp(pi)
[1] 23.14069
> log(100)
[1] 4.60517
> sin(90)
[1] 0.8939967
> cos(30)
[1] 0.1542514
> tan(45)
[1] 1.619775
> cos(0)
[1] 1
```

The above operations may be found in “Trig” and “Math” sections of the *base* package. The enthusiastic reader may find more details by entering “?Math” at the R console. Since it is almost impossible to perform all the required calculations directly at the console level, there is an intrinsic need to create new variables and asssign them the values as needed by the programmer. This task will be taken in the next section.

2.2 Entering and Reading Data from External Files

2.2.1 Data Entering

One of the commands without which it is very difficult to imagine doing any R program is the “c” command. Here, *c* stands for concatenation, or coercing, or combining various R objects in a single object. The following simple examples should help the reader:

```
> a <- c(1:10)
> a
 [1]  1  2  3  4  5  6  7  8  9 10
> b <- c(8:3, 9:11)
> b
[1]  8  7  6  5  4  3  9 10 11
> D<-c(a,b)
> D
 [1]  1  2  3  4  5  6  7  8  9 10  8  7  6  5  4  3  9 10 11
> aa<-c("statisticians", "physicists", "mathematicians", "bayesians")
> aa
```

```
[1] "statisticians"  "physicists"     "mathematicians" "bayesians"
> ab<-c("YES","NO")
> ab
[1] "YES" "NO"
```

In the above lines, we have deliberately not created an object named as “c”. This is to avoid a collision of the variable names with the R commands/functions which may lead to future bugs. A good practice of keeping the bugs to a minimum is not to create them in the first place itself.

Note that this software is case sensitive, and its very important to get a clear understanding of this. For example, “C” is itself another R function which sets contrasts for a factor, and if care is not taken we may end up creating contrasts for an object when we intend to create a new object.

R makes a distinction between a row and a column vector, which can be seen by the example:

```
> c(1:5)
[1] 1 2 3 4 5
> t(c(1:5))
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
```

The length of a vector, say x = c(-3, 8), is found as follows:

```
> x = c(-3:8)
> length(x)
[1] 12
```

Use “*mode*” to know wether an R object is numeric, character, etc. The reader may verify the following:

```
> mode(a<-c(1:10))
[1] "numeric"
> mode(a<-pi)
[1] "numeric"
> mode(a<-c("joker"))
[1] "character"
> mode(TRUE)
[1] "logical"
> mode(a <- c(TRUE, FALSE, 4))
[1] "numeric"
> mode(a <- c(1,7,"joker", 4))
[1] "character"
```

Is there a surprise in the previous two statements?

A word of caution. We will soon see in the near future that R functions such as “mean” and “median” compute the mean and median of given numeric objects. However, mode determines

wether an R object is numeric, character, or logical object.

If we want to enter the values for a vector manually, it may be carried out using the function "scan". Suppose, we want to enter four values for a vector "y", we can do this as given below:

```
> y=scan()
1: 420
2: 840
3: 1680
4: 3360
5:
Read 4 items
> y
[1]  420  840 1680 3360
```

A matrix can be similarly entered as:

```
> X=matrix(data=NA,nrow=2,ncol=2)
> X[,1]=scan()
1: 1
2: 2
3:
Read 2 items
> X[,2]=scan()
1: 3
2: 4
3:
Read 2 items
> X
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

2.2.2 Reading Data from External Files

Data can rarely be entered in a software, and as such is generally stored in some external files like DAT, XLS, CSV, etc. It is then important for us that we read/extract data from these files. We now demonstrate the functions that enable us to read data from external files.

We first show how the function "read.table" works:

```
> (trial<-read.table("trial.dat"))
        V1      V2
1       10      17
2       14      21
3       17      26
4       22      32
5       30      42
```

6	38	53
7	43	58
8	52	68

2.3 Some Basic R Functions

Some functions that need to be explained in more details are the following.

c : We have already used this function. This is a generic function which combines its arguments into a single vector or a list.

gl : The function "gl" generates factor levels. Such a function is very useful in *Design of Experiments*. The arguments needed in this function include the number of observations *n*, and the number of replications, *say k.* A value-added flexibility of this function includes labeling the factors by certain names as desired by the user.

Examples:

```
> gl(2,5)
 [1] 1 1 1 1 1 2 2 2 2 2
Levels: 1 2
> gl(3,4)
[1] 1 1 1 1 2 2 2 2 3 3 3 3
Levels: 1 2 3
> gl(3,2,labels=c("CONTROL","TREATMENT A","TREATMENT B"))
[1] CONTROL     CONTROL     TREATMENT A TREATMENT A TREATMENT B TREATMENT B
Levels: CONTROL TREATMENT A TREATMENT B
```

is, is.na, and **rm**: We first remark here that the function "is.na" is a particular case of the function "is". The function "is" derives its nomenclature from the common usage of that word. Particularly, we can use it to know if the class of a given object is numeric, logical, character, vector, matrix, function, etc.

Examples:

```
> is(5)
[1] "numeric" "vector"
> is(mean)
[1] "function" "OptionalFunction" "PossibleMethod"
> is(TRUE)
[1] "logical" "vector"
> is("XYZ")
[1] "character" "vector" "data.frameRowLabels"
```

"na" is the abbreviation for "not available". If the value of an object is "NA", it indicates that the data is missing. Missing data is a very common phenomenon in the practical world, and especially in large data set. Suppose that the third value of a vector of 5 elements is not available. We can include this information in an R object as follows:

```
> (x=c(1,2,NA,4,5))
[1]  1  2 NA  4  5
```

We can use the command is.na to find the missing values of an R object:

```
> is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
```

This trick can be used to handle missing values effectively in a R program. The use of the function "rm" will be illustrated below and the details may be obtained by the enthusiastic reader:

```
> mean(x)
[1] NA
> mean(x,na.rm=TRUE)
[1] 3
```

Of course, we also advise the reader to ensure that each time he starts a new session, she can better start with

```
> rm(list=ls())
> ls()
character(0)
```

where, ls() includes all the objects in the current session, and not the libraries invoked, and *list* is a function which returns a list or dotted pair list composed of its arguments with each value either tagged or untagged, depending on how the argument was specified.

sort : This function can be used a numeric/complex R object in increasing or decreasing order. Suppose, we want to generate $n = 10$ *spacings* from the standard Uniform distribution. A simple way to do this would be the following:

```
> sort(runif(10))
 [1] 0.08818853 0.12351143 0.19478985 0.26230742 0.41002627 0.41843742
 [7] 0.46687220 0.62715594 0.73156014 0.85986082
```

sample : Suppose, we want to obtain a sample of size 3 from the set $S = \{1, 2, 3, ..., 6\}$. If we want to sample with out replacement, we can achieve this using the "sample" function:

```
> sample(c(1:6),3)
[1] 3 5 2
```

We can sample with replacement using

```
> sample(c(1:6),3,replace=TRUE)
[1] 6 1 6
```

window : The "window" command for an R object extracts elements from the specified between two times 'start' and 'end'. Suppose we want to compute the moving average of a series of observations. Consider the following simple program:

```
> x=c(1:10); max=c(1:8)*0
> for(i in 1:8) { max[i]=mean(window(x,i,i+2))]
> max
[1] 2 3 4 5 6 7 8 9
```

The small and compact, yet powerful, book of Spector (2008) gives many good practices and tools in R for data manipulation as required by the experimenter. Here, the data manipulation does not really mean changing the realised values of the observations.

subset : Data preparation takes most of the time in industrial settings. For example, we may have a data file related to some clinical trial which contains all the observations. Suppose, there are many treatment groups and we are interested only in the set of individuals which received drug *A*. In such case, we would like to prepare an R data frame containing information of related interested as follows:

```
> qaldata=read.csv("/home/CT/CT_007.csv",header=T)
> tr_A=subset(qaldata,qaldata$rxgroup=="A")
> is.data.frame(tr_A)
[1] TRUE
```

Thus, we can use the command "subset" very effectively in numerous ways and get the desired data frame.

table : A simple use of this function is to obtain frequencies of a given factor. The following example illustrates this.

```
> x=c(1:5,8:4,13:6,4:9)
> table(x)
x
 1  2  3  4  5  6  7  8  9 10 11 12 13
 1  1  1  3  3  3  3  3  2  1  1  1  1
```

In general, this function uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.

seq : A sequence of points in a given interval with a specified gap can be generated using this command.

```
> seq(0,1,.2)
[1] 0.0 0.2 0.4 0.6 0.8 1.0
```

```
> seq(1,0,-.2)
[1] 1.0 0.8 0.6 0.4 0.2 0.0
```

solve : The following system of equations can be easily solved in R

$$3x + 4y = 12$$

$$2x + 8y = 20$$

The small R program will be the following:

```
> A=matrix(c(3,2,4,8),ncol=2)
> b=c(12,20)
> x=solve(A,b)
> x
[1] 1.00 2.25
```

integrate : One happy occasion in using R is that it can easily handle integration. The following examples clearly demonstrate

```
> integrate(dnorm,-1.96,1.96)
0.9500042 with absolute error < 1.0e-11
> integrate(dnorm,-3,3)
0.9973002 with absolute error < 9.3e-07
> integrate(sin,0,pi/2)
1 with absolute error < 1.1e-14
> new=function(x) {x^2}
> integrate(new,0,1)
0.3333333 with absolute error < 3.7e-15
> integrate(new,0,1)$value
[1] 0.3333333
```

We will explain more about "$" in the appendix.

2.4 Matrix Computations

Gentle (2007) is a modern account on matrix computations using software though there is more emphasis on the theory. However in his book, for many of the matrix analysis, Gentle prefers R and Fortran. The most recent handbook on matrix analysis with more dedication towards statistics is in Seber (2008).

First we explain the beginning steps for analysis. Matrices are also easily created in R. We first have a look at the working of "matrix" function:

```
> matrix
function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
```

Suppose, we want to create an empty 5 by 6 matrix, named A. The R command

```
> A<-matrix(nrow=5,ncol=6)
```

creates a matrix which reads as follows:

```
> A
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   NA   NA   NA   NA   NA   NA
[2,]   NA   NA   NA   NA   NA   NA
[3,]   NA   NA   NA   NA   NA   NA
[4,]   NA   NA   NA   NA   NA   NA
[5,]   NA   NA   NA   NA   NA   NA
```

whereas,

```
> A<-matrix(c(1:12),nrow=2,ncol=6)
```

gives us the matrix

```
> A
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    3    5    7    9   11
[2,]    2    4    6    8   10   12
```

We know that R is intelligent as

```
> A<-matrix(c(1:12),nrow=2,ncol=2)
> A
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

It is sometimes for the advantage of the user if she needs to assign names for the subscripts of a vector which may be accomplished as under:

```
> X<-matrix(round(rnorm(10),4),nrow=2)
> rownames(X)<-rownames(X,do.NULL=FALSE,"Sl.No.")
> colnames(X)<-colnames(X,do.NULL=TRUE)
> colnames(X)<-c("Suresh","Mahesh","Ganesh","Lokesh","Rupesh")
> X
          Suresh  Mahesh  Ganesh  Lokesh  Rupesh
Sl.No.1   0.9190  0.0746  0.6198 -0.1558 -0.4782
Sl.No.2   0.7821 -1.9894 -0.0561 -1.4708  0.4179
```

The reader may find more details about “case.names” and “variable.names”.

Matrix computations are straightforward here. Crossproducts between two matrix, say A and B, may be performed as follows:

```
> A<-matrix(nrow=2,ncol=2)
> data.entry(A)
> A
     var1 var2
[1,]    4    5
[2,]    8   13
> B<-matrix(nrow=2,ncol=3)
```

```
> data.entry(B)
> B
     var1 var2 var3
[1,]   10   15    8
[2,]    4    6    9
> A%*%B
     var1 var2 var3
[1,]   60   90   77
[2,]  132  198  181
```

For any R object, say E, the function data.entry(E) pops-up a data editor where in we enter the values for the elements of E. As usual, we leave it to the reader for finding more about the function by using "?data.entry" at the R console. The operator "%*%" carries out cross-product between two matrices. Unsurprisingly, we also get the following as expected:

```
> B%*%A
Error in B %*% A : non-conformable arguments
```

The inverse of a matrix, after loading the package MASS, is obtained as

```
> library(MASS)
> ginv(A)
              [,1]          [,2]
[1,]  1.0833333   -0.4166667
[2,] -0.6666667    0.3333333
> round(ginv(B),4)
          [,1]         [,2]
[1,]  0.0477    -0.0424
[2,]  0.0716    -0.0637
[3,] -0.0690     0.1724
```

Eigen values, eigen vectors, and singular value decomposition for appropriate matrices may be easily computed using the in-built R commands:

```
> eigen(A)
$values
[1] 16.2620873  0.7379127
$vectors
              [,1]          [,2]
[1,] -0.3775776  -0.8375173
[2,] -0.9259779  0.5464109
> eigen(B)
Error in eigen(B) : non-square matrix in 'eigen'
> svd(A)
$d
[1] 16.5370324  0.7256441
$u
              [,1]          [,2]
```

```
[1,] -0.3850756  -0.9228850
[2,] -0.9228850   0.3850756
$v
                [,1]            [,2]
[1,] -0.5396000  -0.8419215
[2,] -0.8419215   0.5396000
```

We conclude the sub-section with Cholesky decomposition, the importance of which can't be justified in a very small volume such as this one, and we simply give the example in the R package:

```
> ( m <- matrix(c(5,1,1,3),2,2) )
     [,1] [,2]
[1,]    5    1
[2,]    1    3
>      ( cm <- chol(m) )
              [,1]          [,2]
[1,] 2.236068   0.4472136
[2,] 0.000000   1.6733201
```

Note the difference the extra paranthesis do when executed.

2.5 Working with packages

It is impossible to have a perfect software package. This problem is overcome by allowing the statistician to download a package of her choice or requirement and install the same. As of date, there are 1590 additional package on the R repository. From 600 add-on packages in 2006, addition of approximately 1.1836 packages per day is anything but spectacular. This abundance of add-on packages ensures that the need of a statistician to write an extensive new program is kept to a minimum.

The packages that are shipped along with the current version of R are base, boot, class, cluster, codetools ,datasets, foreign, graphics, grDevices, grid, KernSmooth, lattice, MASS, methods, mgcv, nlme, nnet, rpart, spatial, splines, stats, stats4, survival, tcltk, tools, and utils. These packages have been found to have more applications and frequency among the users. These packages are stored in the directory /usr/lib/R/library. The add-on packages are stored in the directory /usr/lib/R/site-library.

Most of the packages contain data sets, and a list of which may be obtained by:

```
> try(data(package = "mypackage") )
```

An R package in the repositry may be loaded using either of the commands package() or require(). More frequently, it is possible that the requried package may have dependencies on other packages.

In such cases we also need to install all such dependent packages. If the laptop is connected with internet, the optimal way for installing the package is

> install.packages("mypackage")

The user is first required to specify a CRAN mirror. In this case R will itself identify the dependent packages and download them and complete the installation. If you wish to install more than one package, try this:

> install.packages(c("package1", "package2", ..., "packagen"))

If the user is not always connected with the web, she needs to download the requred R package from http://cran.r-project.org/web/packages/ to the folder /local/myRdownloads. The downloaded packaged may be installed by opening a terminal at that folder and by executing the command

sudo apt-get R CMD INSTALL mypackage.gz

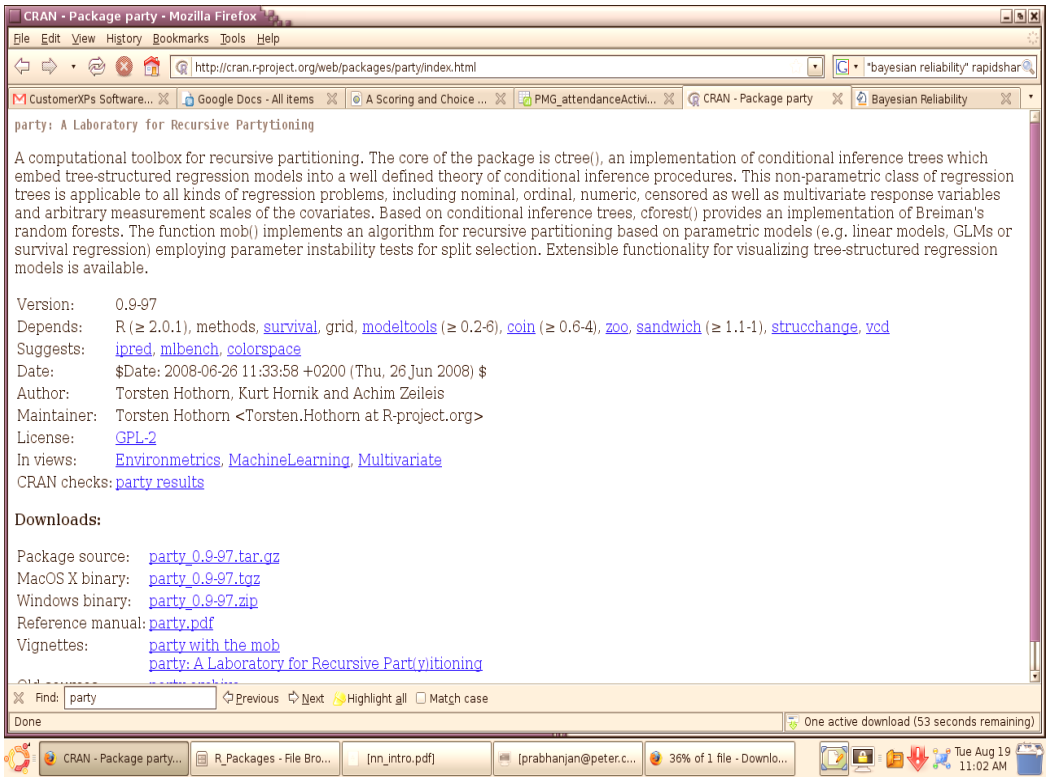


Figure 2.1: A Screenshot of the "Packages" from the CRAN web-page

Ofcourse, there are more details for installing a package than just a simple one-line command. We are helping the reader assuming that she is working on Ubuntu Operating System. If the reader is working on "Linux/Mac/Windows", she should download the package listed below

"Downloads:" in Figure 2.1 from "Package source: / MacOS X binary: / Windows binary:" respectively. It is sometimes possible that despite being able to download a package completely, the installation may not be successfully completed. One reason may be that the current package may depend on other packages and such packages also need to be installed before this one. The related packages are listed on "Depends:" menu on the web page http://cran.r-project.org/web/packages/mypackage/index.html. It may be possible that sometimes one needs additional software such as gfortran, Weka, etc. If the user fails by all resources available to her, she may always write about the specific problem to the R-mailing list, and progress further.

To know all the functions available in a certain package, use the command *library(help=*"mypackage") .

Creators of the packages continuously update their packages, and it may be important for the user to upgrade his version too. In that case, update the package by

> update.packages("mypackage")

The packages are voluntarily developed by statisticians, and in general some help about the same is given along with the package. Some times the statisticians provide "Manual" for the package on their web homepage. It is interesting to know that information about the packages are also published in journals, and in particular, *Journal of Statistical Software* is a dedicated journal towards publishing articles, book reviews, code snippets, and software reviews on the subject of statistical software and algorithms. The authors were even more delighted to find that, like our dear FOSS, all the articles from this journal may be freely downloaded from the web http://www.jstatsoft.org/. This gives us one more reason to stick with FOSS in general, and R in particular.

Finally, one can get a list of all packages:

> available.packages() # list will be available after selection of a CRAN mirror

Further, if you wish to install all the packages (1590 of them!), try this:

> install.packages(available.packages()[,1])

2.6 R Interfaces: Connection to Other Software

Statisticians and Mathematicians will be using some or the other software according to their requirements and familiarities. For instance, a statistician may be using R and a mathematician Matlab. It often happens in the practical world that there arises a need to use some function or tool which is not available in her software, and may be forced to use a new and unfamiliar software. It may even happen that a user in R may be required to use Gibbs sampler which has been well

developed in BUGS (Bayesian Inference Using Gibbs Sampler). Of course, it is not the case that Gibbs sampler can't be developed with the domain of R. A more popular version of BUGS is WinBUGS. Fortunately, *interface* is a very good option for such switchovers.

In general, an **interface** defines the communication boundary between two entities, such as a piece of software , a hardware device, or a user, which refers to an abstraction that an entity provides of itself to the outside. Such interfaces come very handy for a programmer.

R has interfaces to many of the popular softwares, such as Matlab, WinBUGS, WEKA, among others, which are given as add-on packages respectively by *Rmatlab, R2WinBUGS, RWeka.*

2.7 Managing R Sessions

We close this chapter with a discussion on managing R sessions. Though we had a few sessions earlier, the instructions here override the previous sessions. To begin with, a good practice with a session for a new task, it is advised to run the following command at the beginning of a session:

```
> rm(list=ls())
> ls()
character(0)
```

This ensures that we have our current session is fresh and devoid of any existing R objects. The next step is know in which directory we are running our current session, and this may be found as

```
> getwd()
[1] "/home/user"
```

The next step is setting an appropriate directory where our session is stored/saved, which can be accomplished by the command:

```
> (setwd("/home/myRsession))
[1] "/home/myRsession"
```

It is possible to list all the files in the current root directory within the session by executing

```
> list.files()
```

When working from a terminal on a Ubuntu system, a R session starts with executing "R" at the kernel:

```
prabhanjan@peter:~$ R

R version 2.7.1 (2008-06-23)
Copyright (C) 2008 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

Figure 2.2: A Screenshot of an R Beginning Session

We are sure that the user will get tired of seeing the details each time she starts a session. This detail can be easily avoided by executing “R -q”, “R --quiet”, or “R --silent” at the terminal upon which she will have a pleasant start. Interestingly, the command “R -s” does things differently. If this command is executed at the linux kernel, R starts without the console “>”, and when we conclude the session with “q()”, it does not prompt

Save workspace image? [y/n/c]:

After working in a session for a long time, the user may list all the objects using “ls()”. Further, the command “Ctrl + l” clears the previous lines on the screen without deleting any R object. Though these commands may seem trivial, sometimes they are good aesthetic practices.

An R object or output may be saved to a file using

```
> dput(mean(1:10),"my.mean")
> outmean<-dget("my.mean")
> outmean
[1] 5.5
```

R objects, say x and y, may be written to a file as

```
> save(x,y,file="xy.Rdata")
```

Finally, we give the commands which are useful to save a session. The entire session may be saved

as

```
> save.image("mysession")
```

Note that R will overwrite if such a file already exists in the root directory. For the lazy person, the session will be simply saved by choosing “y” after executing

```
> q()
Save workspace image? [y/n/c]:
```

There may be more efficient ways of managing a session, and we have given methods best known to us.

* * * * *

3. Exploratory Data Analysis

Package(s) : datasets

Data set(s) : SIRDS.csv, InsectSprays, sample.csv, AirPassengers, Pete_Sampras.csv

3.1 Introduction: The Tukey's Way of Statistics

The main spirit of Exploratory Data Analysis, abbreviated as EDA, is to allow the data to speak for themselves instead of assuming that there exists some underlying mechanism which generates it and that its mathematical form may be linear, or quadratic, etc.

We feel that an analyst must first have a feel of the data, and this is where EDA scores over the traditional analysis. In fact the methods in this framework are very rich in intuition, simpler to understand, and very easy to interpret the results. However, the reader is cautioned not to use these results to complement the traditional method of analysis.

Many experts and programming geeks have found that R does graphics and **E**xploratory **D**ata **A**nalysis very elegantly and also provides rich flexibility to the user. Tukey believed that data analysis should be carried out by not just using the stringent parametric models, in certain sense, and one should rather holistically allow the data to speak for themselves. It is this philosophy which lead to the birth of EDA and which was richly developed by the Tukey school. Tukey and colleagues did write the earliest work on EDA and which make a very good reading even today. It is also worth noting here that in the year 1962, Tukey wrote an expository article in the *Annals* titled as "The Future of Data Analysis". The reason for mentioning all the above story is that the predecessor of R, namely S, was developed at the Bell Labs to handle EDA effectively, and thus, R is unsurprisingly a very effective tool for the users.

Mosteller and Tukey (1977), Tukey (1977), Dasu and Johnson (2003), Velleman and Hoaglin (1984) are detailed works on EDA.

The driving philosophy of EDA is that data should suggest the true hypotheses and not the other way which one does in a regular course in statistics. Tukey calls the regular methods as *Confirmatory Data Analysis.* The two major approaches of EDA include graphical techniques and quantitative methods. The graphical techniques are effectively used for understanding how the data unfolds, and the quantitative methods may be used for further confirming the understanding as envisioned in the graphs. When the data is displayed in an appropriate way, we may very likely capture some patterns/trends in it which may plausibly go unnoticed in the traditional data analysis

methods.

3.2 Graphical Techniques in EDA

The main strength of EDA is understanding of the data using graphical methods. The most commonly used graphics include

- Box Plot
- Histogram
- Run Chart
- Pareto Chart, and
- Stem-and-Leaf Plot.

3.2.1 Box Plot. The box plot are also known as *box and whisher plots*. This plot is constructed on the five important summaries: sample minimum, lower quartile* (Q1), median (Q2), upper quartile (Q3), and sample maximum.

* quartile: By definition, quartiles divide the sample into four parts with each part containing one fourth of the sample.

Example. **Birth Weights of Infants with Severe Idiopathic Respiratory Distress Syndrome (SIRDS).** Van Vilet and Gupta (1973) studied the birth weight of infants who were having SIRDS. The data set on fifty infants is available in the file "SIRDS.csv". Twenty-seven infants in the study were reported dead.

Our source for this example is from the web-link http://openlearn.open.ac.uk/mod/resource/view.php?id=165503 . We begin reading the data.

```
> sirds=read.csv("/mypath/SIRDS.csv",header=T)
> attach(sirds)
> tiff("sirds.tiff")
> boxplot(Weight~Survived) ->bp
> dev.off()
```

Important summaries are as given below:

```
> bp$names
[1] "No"  "Yes"
> bp$stats
     [,1]  [,2]
[1,] 1.030 1.130
[2,] 1.245 1.740
[3,] 1.600 2.200
```

```
[4,] 2.070 2.765
[5,] 2.730 3.640
> summary(Weight[Survived=="No"])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.030   1.245   1.600   1.693   2.070   2.730
> summary(Weight[Survived=="Yes"])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.130   1.740   2.200   2.308   2.765   3.640
> library(e1071)
Loading required package: class
> skewness(Weight[Survived=="No"])
[1] 0.474189
> skewness(Weight[Survived=="Yes"])
[1] 0.2186368
```

Using “summary” command above we simply verified the “stats” arrived by the boxplot command. The graphical display obtained by boxplot is given in Figure 3.1.

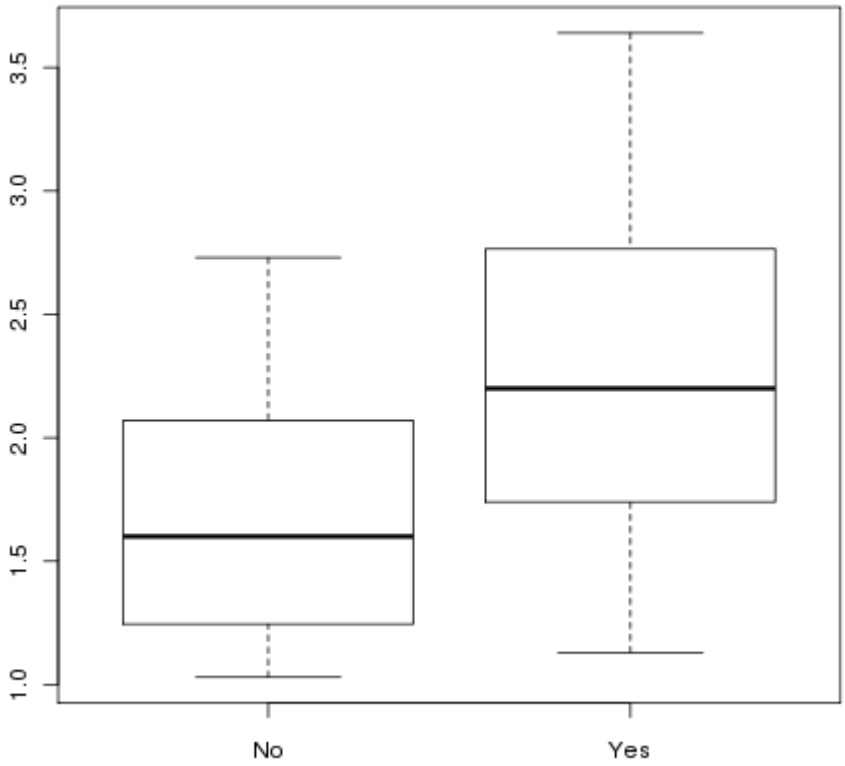


Figure 3.1. The boxplot of infants from the SIRDS.csv file

We see from the summaries we see that for the birth weights (in kg) of the infants who survived, the lower quartile, median and upper quartile are, respectively, 1.72, 2.20 and 2.83,

whereas for the infants who died, the corresponding quartiles are 1.23, 1.60 and 2.20. In both the cases the adjacent values are equal to the sample maxima and minima, so that the whiskers extend to the ends of the sample range.

Inference from the Boxplot. Figure 3.1 shows that the median birth weight of infants who survived is greater than that of those who died. The interquartile ranges, indicated by the lengths of the boxes, are reasonably similar though the overall range of the data set is greater for the surviving infants, which is reflected by the distances between the ends of the two whiskers for each boxplot. Since the upper whiskers for the both the plots appear to be far from the median line, it appears that the data should be right-skewed. The box-plots also indicate that the skewness is larger for the birth weights of the infants who died. This is also verified from the summary produced earlier. Since the median birth weight of infants who died is less than the lower quartile of the birth weights of infants who survived we can (probably) safely say that survival is related to birth weight.

Example 2. The Effect of Insecticides. McNeil (1977). Six insecticides, labeled A-F, were used in an agricultural experiment and the number of insects found after using them are counted. The data set is available as InsectSprays in the package "datasets".

We first count the number of insects death due to each of the insecticides:

```
> countfreq=c(1:6)
> for(i in 1:6){
+ countfreq[i]=sum(count[spray==LETTERS[i]])}
> countfreq
[1] 174 184  25  59  42 200
```

The numbers 174, ..., 200 count the number of insects deaths due to insecticides A, ..., F. Clearly, the frequencies show that the Insecticides A, B, and F are very powerful as compared with C, D, and E. We now plot the boxplot:

```
> tiff("insect.tiff")
> boxplot(count~spray)
> dev.off()
```

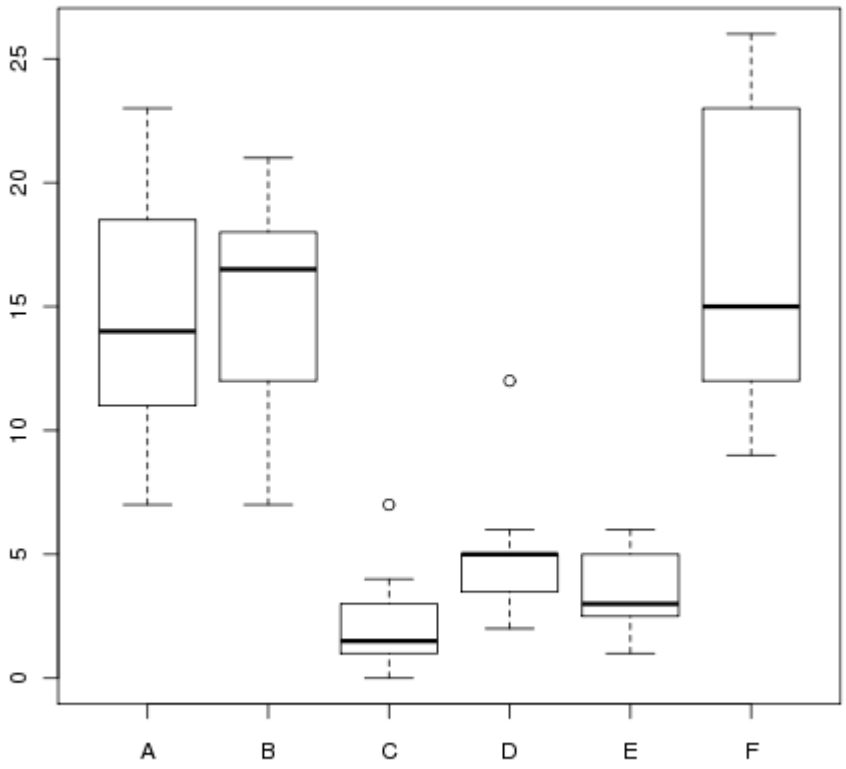


Figure 3.2. Box plot of the counts of Insects for Different Sprays

3.2.2 Histogram. The usuage of histogram is probably as old as data analysis itself, and is one of the oldest method of graphical display. Its origin is too earlier than EDA, and yet it is considered by many EDA experts to be very useful and makes it to the list of one of the very useful practices of EDA.

Example 1. In the file "sample.csv", we have data from five different probability distributions. Towards understanding the plausible distribution of the samples, we plot the histogram and see how useful it is.

```
> sample=read.csv("/mypath/sample.csv",header=T)
> tiff("hist_samples.tiff")
> par(mfrow=c(2,3))
> hist(sample[,1],main="Histogram of Sample 1", xlab="sample1", ylab="frequency")
> hist(sample[,2],main="Histogram of Sample 2", xlab="sample2", ylab="frequency")
> hist(sample[,3],main="Histogram of Sample 3", xlab="sample3", ylab="frequency")
> hist(sample[,4],main="Histogram of Sample 4", xlab="sample4", ylab="frequency")
> hist(sample[,5],main="Histogram of Sample 5", xlab="sample5", ylab="frequency")
> dev.off()
```

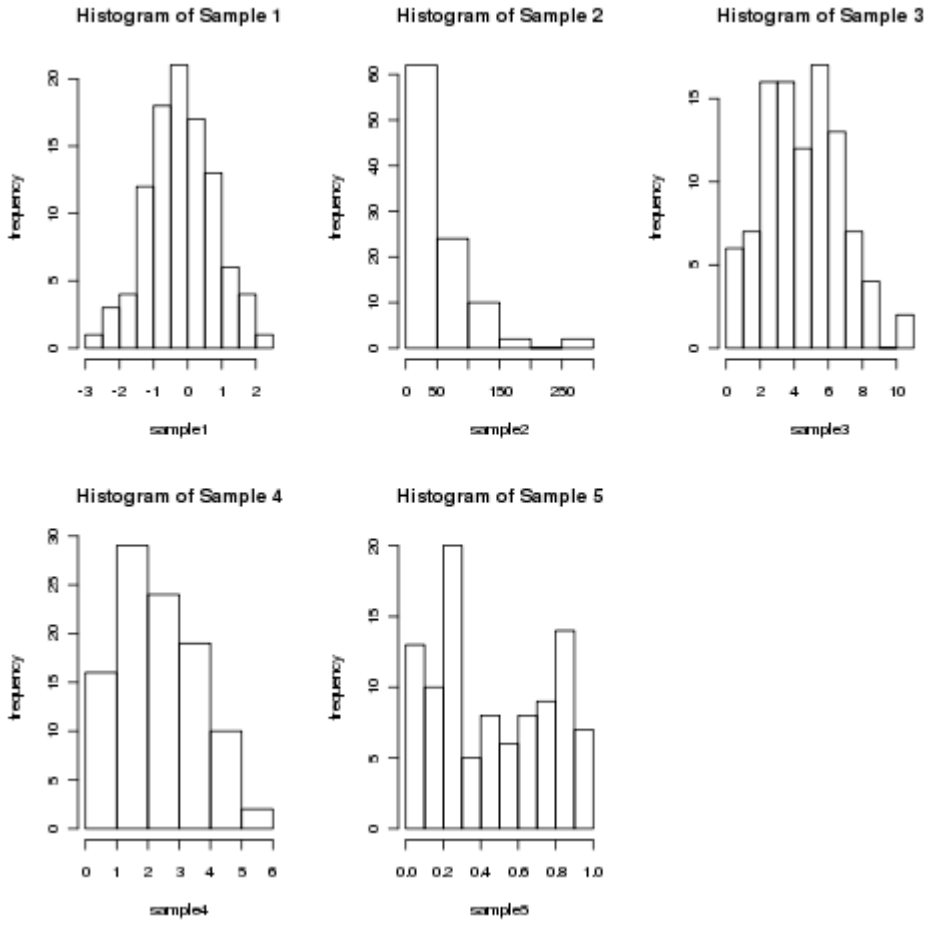


Figure 3.3. Plot of Histograms for Different Samples

Part of the data is displayed below:

Sample_1	Sample_2	Sample_3	Sample_4	Sample_5
0.23	21.97	2	3	0.81
-0.4	9.95	6	2	0.92
-0.45	52.68	3	3	0.23
-1.14	118.98	6	2	0.42
-1.22	1.33	9	3	0.74
0.32	139.24	4	5	0.4
-0.14	90.2	4	2	0.69
-1.8	43.02	7	6	0.23
0.32	36.68	6	4	0.22
-0.05	56.92	4	1	0.22
-0.03	51.71	3	4	0.24
-0.46	43.5	3	3	0.02
1	20.43	2	1	0.28
-0.75	67.87	6	2	0.53
0.37	85.23	4	4	0.14
-1.09	144.52	2	4	0.03
1.19	28.24	3	2	0.27
0.66	72.86	9	4	0.65
-0.69	60.94	1	4	0.88

Important summaries for the sample data is found as below.

```
> lapply(sample,summary)
$Sample_1
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.6700 -0.8225 -0.1450 -0.1845  0.3950  2.3200
$Sample_2
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.33   14.75   40.33   53.27   71.43  291.40
$Sample_3
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    3.00    5.00    4.98    7.00   11.00
$Sample_4
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    2.00    3.00    2.83    4.00    6.00
$Sample_5
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0200  0.2175  0.4200  0.4643  0.7525  0.9500
```

Yes, standard deviation summary is missing in the above, and which we easily produce below:

```
> lapply(sample,sd)
$Sample_1
[1] 0.9714806
$Sample_2
[1] 54.16076
$Sample_3
[1] 2.282919
$Sample_4
[1] 1.318516
$Sample_5
[1] 0.2981195
```

Remarks on Sample 1. The histogram of sample 1 is a bell-shaped, and its peak (mode) is near 0. The mean and median are respectively -0.1845 and -0.1450, again close to 0. The standard deviation and variance are respectively 0.9714806 and 0.9437745. The shape indicated by the histogram and the summaries very likely indicate that the distribution of the sample may be a *normal distribution*.

Remarks on Sample 2. The histogram of Sample 2 is tailing off very fast after the value of 50. The distribution indicates positive skewness, and also the variance 2933.388 is approximately the square of the mean 53.27. Further, all the values are non-negative which leads us to believe that the sampling distribution may be from *exponential distribution*.

Remarks on Sample 3 and Sample 4. An important feature of these data is that all the values are non-negative integers. This makes us believe that the sampling distribution is a discrete distribution. The mean and variance of Sample 3 is 4.98 and 5.2117, whereas these numbers for Sample 4 is 2.83 and 1.7384. The mean and variance are almost equal which is a characteristic of *Poisson*

distribution and the variance being larger rules out the possibility of the sample being from binomial distribution. Similarly, the variance of Sample 4 being less than its mean is a reflection that this sample maybe from *binomial distribution.*

The reader is left to carry out similar inference about the Sample 5.

We will see more about these samples again in Chapter 4.

3.2.3 Run Chart. A run chart is also known as *run-sequence plot.* In the run chart the data value is simply plotted against its index number. For example, if $x_1, x_2, \ldots, x_t$ is the data, we then plot $x_1, x_2, \ldots, x_t$ against 1, 2, ..., *t.* We can plot the run charts in R using the function *plot.ts* which is very commonly used in time-series analysis.

Example. To illustrate the power of this method, consider the data set of "Airline Passengers" of US for the period 1949-1960, which has been popularized by Box and Jenkins (1972).

```
> library(datasets)
> AirPassengers
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
                          . . .
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
> plot.ts(AirPassengers)
```

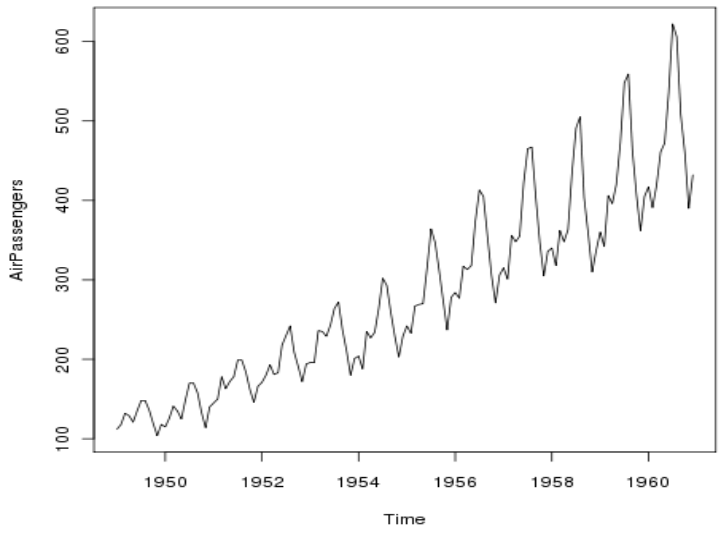


Figure 3.4: Run Chart for the Air Passengers Data

The run chart clearly shows that there is an increasing trend in the number of users. We can also see that there is a seasonal variation whose size is roughly equal to the local mean.

3.2.4 Pareto Chart. The Pareto law, also known as the 80-20 law, in general states that 80% of the resources is used by 20% of the tasks and 20% of the resources is used by 80% of the tasks.

The Pareto chart contains both line and bar graphs, where the bars are displayed in decreasing order, and the line graph indicates the cumulative totals of each category from left to right. It is a very useful tool in *Statistical Quality Control,* abbreviated as SQC. In Chapter 4, Montgomery (2001) lists the Pareto chart as one of the "*Magnificent Seven*" in the tool kit of SQC. As such there is no built-in function for obtaining the Pareto chart. However, we use this opportunity to demonstrate how useful is the *R-mailing list*. One of the R users needed to plot the Pareto chart for some data he had. Jason Turner then posted his querry to the R-mailing list on 31 January, 2001, see https://stat.ethz.ch/pipermail/r-help/2002-January/018406.html . Don Wingate was kind enough to exclusively write an R program for Turner and posted the reply for the rest of world too on 13 February, 2001, see https://stat.ethz.ch/pipermail/r-help/2001-February/011099.html . We produce below the codes provided by Wingate:

```
#################
# pareto. Produces a Pareto plot of effects.
# Parameters:
#       effects - vector or matrix of effects to plot.
#       names   - vector of names to label the effects.
#       xlab    - String to display as the x axis label.
#       ylab    - String to display as the y axis label.
#       perlab  - Label for the cumulative percentage label.
#       heading - Vector of names for plot heading.
#
pareto <- function(effects, names=NULL, xlab=NULL, ylab="Magnitude of Effect",
indicate.percent=TRUE, perlab="Cumulative Percentage", heading=NULL)
{
   # set up graphics parameters, note: set las=2 for perpendicular axis.
      oldpar <- par( mar=c(6, 4, 2, 4) + 0.1 , las=3)
      on.exit(par(oldpar))
      if( ! is.matrix(effects)) effects<-as.matrix( effects )
      for( i in 1:ncol(effects) )
      {
            if( i==2 ) oldpar$ask<-par(ask=TRUE)$ask
            # draw bar plot
            eff.ord <- rev(order(abs(effects[,i])))
            ef <- abs(effects[eff.ord,i])
            # plot barplot
```

```
        ylimit<-max(ef) + max(ef)*0.19
        ylimit<-c(0,ylimit)
        par( mar=c(6, 4, 2, 4) + 0.1 , las=3)
        x<-barplot(ef, names.arg=names[eff.ord], ylim=ylimit,
                   xlab=xlab, ylab=ylab, main=heading[i])
        if( indicate.percent == TRUE ){
            # get cumulative sum of effects
            sumeff <- cumsum(ef)
            m<-max(ef)
            sm<-sum(ef)
            sumeff <- m * sumeff/sm
            # draws curve.
            lines(x, sumeff, lty="solid", lwd=2, col="purple")
            # draw 80% line
            lines( c(0,max(x)), rep(0.8*m,2) )
            # draw axis labling percentage.
            at <- c(0:5)* m/5
           axis(4, at=at,labels=c("0","20","40","60","80","100"), pos=max(x)+.6)
            # add axis lables
            par(las=0)
            mtext(perlab, 4, line=2)
        }
    } # end for each col
}
```

Example. For a chart whose codes have been obtained from the internet, we resort to internet again for a data set. Click on the web-link http://www.otago.ac.nz/sas/qc/chap26/sect4.htm . From this page, we copy and paste the below table. We assume that the reader has run the above codes for the Pareto chart in R.

Obs	cause	COUNT	PERCENT
1	Contamination	14	45.1613
2	Corrosion	2	6.4516
3	Doping	1	3.2258
4	Metallization	2	6.4516
5	Miscellaneous	3	9.6774
6	Oxide Defect	8	25.8065
7	Silicon Defect	1	3.2258

```
> freq=c(14,2,1,2,3,8,1)
 >names(freq)<-  c("Contamination","Corrosion","Doping","Metallization","Miscellaneous",
 "Oxide Effect","Silicon Effect")
>pareto(freq)
```

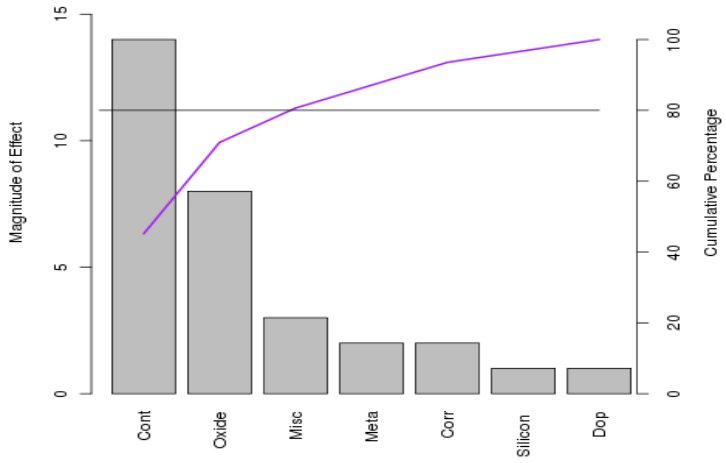


Figure 3.5. Pareto Chart for the Failure Causes

3.2.5 Stem-and-Leaf Plot. Velleman and Hoaglin (1984) describe the basic idea of stem-and-leaf display as allowing the digits of the data values to do the sorting of the data into numerical order and display the same. The steps for constructing stem-and-leaf are listed below.

Step 1. Select an appropriate pair of adjacent digits position in the data and split each observation between the adjacent digits. The digits selected on the left side of the data are called *leading digits.*

Step 2. Sort all possible leading digits in ascending order. The all possible leading digits are called as *stems.*

Step 3. Write the first digit of each data value aside its stem values. The first digit is referred as *leaf.*

In Step 2, we list all possible stems irrespective of wether they occur in the given data set or not.

We illustrate this method using the information on the wins and losses of Pete Sampras during the period 1988-2002.

Example. Pete Sampras Win-Loss Record Data. The number of win-loss for Pete Sampras during the period 1988-2002 is given in the below table.

This data is available in the file "Pete_Sampras.csv". Let us first focus on the wins of Sampras during the period under consideration. The minimum number of wins in a year for him is 10 and the maximum is 85.

Year	1988	1989	1990	1991	1992	1993	1994	1995
Wins	10	18	51	52	72	85	77	72
Loss	10	19	17	19	19	16	12	16
Year	1996	1997	1998	1999	2000	2001	2002	
Wins	65	55	61	40	42	35	27	
Loss	11	12	17	8	13	16	17	

Using the default settings in the R command, we get the following display:

```
> pete=read.csv("/mypath/Pete_Sampras.csv",header=T)
> stem(pete$Wins)
The decimal point is 1 digit(s) to the right of the |
 0 | 08
 2 | 75
 4 | 02125
 6 | 15227
 8 | 5
```

The above display means that the observations are 00, 08, 27, 25, 40, 42, 41, 42, 45, 61, 65, 62, 62, 67, 85.
With the small change in the command, we get the below result:

```
> stem(pete$Wins,scale=2)
 The decimal point is 1 digit(s) to the right of the |
  1 | 08
  2 | 7
  3 | 5
  4 | 02
  5 | 125
  6 | 15
  7 | 227
  8 | 5
```

In summary, EDA give very deep insight into the data. For more plots and quantative methods, refer Tukey (1977) and Vellemen and Hoaglin (1984).

* * * * *

4. Probability and Statistical Inference

Package(s) : prob, PASWR, ISwR, LearnBayes

Data set(s) : sleep, energy

4.1 Probability

We introduce probability to the reader using the "*prob"* package developed by Prof. G. Jay Kerns which helps us to neatly visualize the sample space, events, union, and intersection of sets. This further eases the task of introducing probability, conditional probability, and expectations. The only catch here is that illustration can be done using only finite sample space, and thereby ruling out countably finite and continuous sample space.

As with add-on packages, we first load this package with

```
> require(prob)
Loading required package: prob
Attaching package: 'prob'


      The following object(s) are masked from package:base :
       intersect,
       setdiff,
       subset,
       union
```

Figure 4.1: Screenshot of the R Session when loading "prob" package

4.1.1 Sample Spaces

Coin tossing and rolling die are a probabilists favorite example, well, the probability is very high to be more precise, and this package accomplishes this with aplomb:

```
> S = tosscoin(2)
> S
  toss1 toss2
1     H     H
2     T     H
3     H     T
4     T     T
```

```
> (S= rolldie(2))
   X1 X2
1   1  1
2   2  1
3   3  1
4   4  1
5   5  1
6   6  1
7   1  2
  . . .
35  5  6
36  6  6
```

Following are the functions included in this package:

R Command	Function of the command
addrv	Adding Random Variables to a Probability Space
cards	A Standard Set of Playing Cards
countrep	Count Repetitions
empirical	Empirical Summary of a Simulation
euchredeck	A Deck of Playing Cards for Euchre
iidspace	Independent Identical Experiments
intersect	Intersection of Subsets
is.probspace	Testing for a Probability Space
isin	Test Whether One Vector Is In Another Vector
isrep	Is Repeated in a Vector
marginal	Marginal Distributions
noorder	Sort and Merge Probability Space Outcomes
nsamp	Number of Samples from an Urn
permsn	Generate All Permutations of x Elements Taken m at a Time
prob	Probability and Conditional Probability
prob-package	Elementary Probability on Finite Sample Spaces
probspace	Probability Spaces
rolldie	Rolling a Die
roulette	Roulette
setdiff	Set Difference of Subsets
sim	Simulate Draws from a Sample Space
subset	Subsets of Probability Spaces
tosscoin	Tossing a Coin
union	Union of Subsets
urnsamples	Sampling from urns

Table 4.1: List of functions in the *prob* package

The other experiments whose sample spaces may be easily seen using the package are playing cards, roulette and euchredeck.

Urns constitute a different kind of sampling method, and is distinct from coin tossing and rolling die examples wherein the sample space depends on its contents which may be varying. The function *urnsamples* works as follows:

urnsamples(x, size, replace = FALSE, ordered = FALSE, ...)

The four types of sampling that can be carried out using urnsamples are (i) Ordered, With Replacement, (ii) Ordered, Without Replacement, (iii) Unordered, Without Replacement, and (iv)

Unordered, With Replacement. For example, if the number of objects in an urn is 5 and we have to draw a sample of size 3 using either of the mechanism (i)-(iv), the R commands will be the following:

```
> urnsamples(10:14,size=3,replace=TRUE,ordered=TRUE)
> urnsamples(10:14,size=3,replace=FALSE,ordered=TRUE)
> urnsamples(10:14,size=3,replace=FALSE,ordered=FALSE)
> urnsamples(10:14,size=3,replace=TRUE,ordered=FALSE)
```

For the sake of brevity, the sample spaces have not been given here. In the next subsection we handle algebra of sets.

4.1.2 Counting Methods and Set Algebra

Consider the following sets, $A = \{1, 2, ..., 8\}$, $B = \{4, 5, ..., 10\}$, and $D = \{9, 10, ..., 13\}$. We can easily perform set algebras as indicated below:

```
> A=c(1:8); B=c(4:10); D=c(9:13)
> union(A,B) # union of two sets
 [1]  1  2  3  4  5  6  7  8  9 10
> intersect(A,B) # intersection of two sets
[1] 4 5 6 7 8
> intersect(A,D) # check if two sets are disjoint
integer(0)
> setdiff(A,B) # Set difference, that is, A - B
[1] 1 2 3
> setdiff(A,D) # why is the task being repeated?
[1] 1 2 3 4 5 6 7 8
```

4.1.3 Standard Discrete Probability Distributions

We begin with discrete distributions. Many standard probability densities can be studied using built-in R commands. The three-common and useful functions start with the letters p, d, and q, followed by a standard abbreviation of that distribution. Here, p, d, and q stand for distribution function, density, and quantile function respectively. Of course, there is a fourth one too which will be taken for consideration in the next chapter.

Binomial Distribution. Suppose that X denotes the number of heads on a throw of n times of a coin, and let p denote the probability of getting a head. Then X is a binomial random variable with parameters (n,p), and its probability distribution is given by

$$p(x;n,p) = {}^{n}C_{x}\, p^{x}(1-p)^{n-x},\ x = 0,1,2,\ldots,n.$$

Example 1. Suppose that $n = 20$, and $p = 0.35$. The below diagram demonstrates the use of the trio d, p, and q.

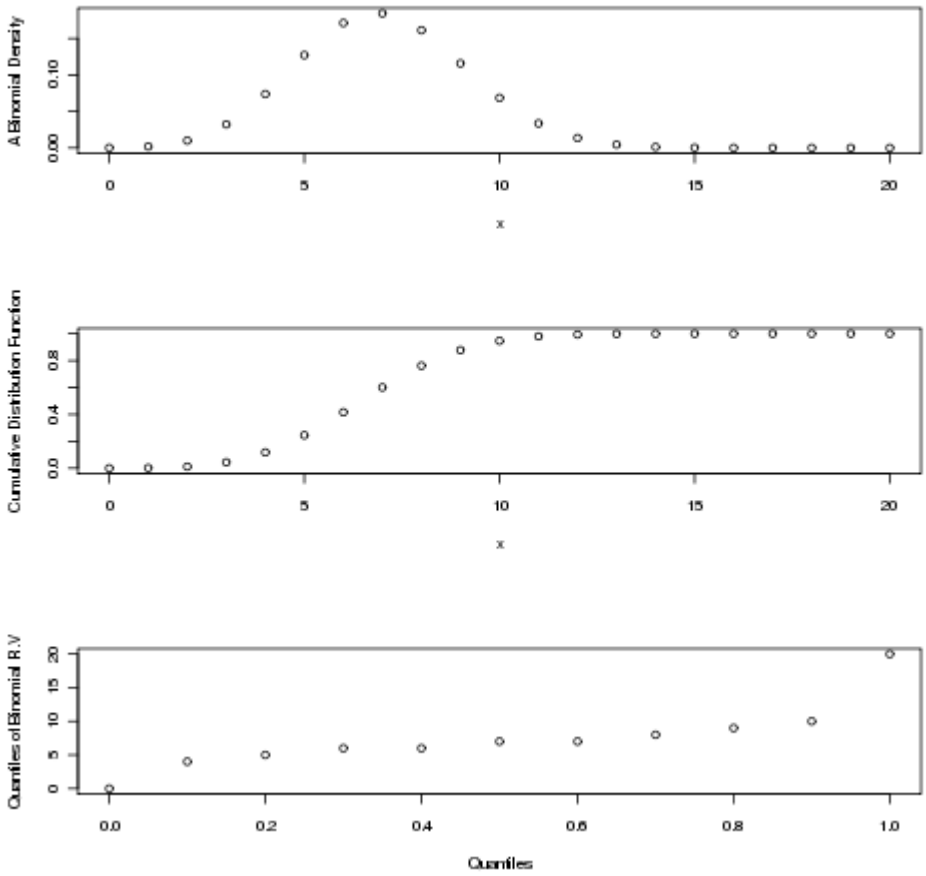


Figure 4.2 The Density, Cumulative Distribution, and Quantile Plot for a Binomial Random Variables

The above plot can be generated using the following codes:

```
> n=20; p=0.35
> tiff("Understanding_Binomial_Distribution.tiff")
> par(mfrow=c(3,1))
> plot(0:20,dbinom(0:n,n,p),xlab="x",ylab="A Binomial Density")
> plot(0:20,pbinom(0:n,n,p),xlab="x",ylab="Cumulative Distribution Function")
>  plot(seq(0,1,.1),qbinom(seq(0,1,.1),n,p),xlab="Quantiles",ylab="Quantiles of Binomial
                R.V")
> dev.off()
```

Example 2. Suppose that $X \sim B(10,.6)$ and $Y \sim B(14,.5)$. Then, the probability *P(X>Y)* can be evaluated by the following program.

```
> pxgy=0       #initializing the value for computation
> for(i in 1:9){          # Since X can't be greater than Y if  Y ≥ 10
+ for(j in i+1:10){              # Accounting for all the values of X greater than Y
+ pxgy=pxgy+dbinom(i,14,.5)*dbinom(j,10,.6)}}
> pxgy
[1] 0.2699851
```

Poisson Distribution. The probability function of a Poisson random variable is given by

$$P(X=x)=\frac{\exp(-\lambda)\lambda^{x}}{(x)!}, x=0,1,2,...$$

It is well known that the mean and variance of Poisson random variable are equal, that is,

$$E(X)=var(X)=\lambda$$

Example 1. Suppose that X follows a Poisson distribution with $\lambda=10$. Then, its percentiles are given by

```
> qpois(seq(0,1,.1),10)
 [1]  0  6  7  8  9 10 11 12 13 14 Inf
```

Further, the probability that *X* is greater than 15 is

```
> 1-ppois(15,10)
[1] 0.0487404
```

Example 2. Suppose that it is known that for a Poisson random variate *P(X=0) = 0.2*. Then *P(X=6)* is given by

```
> lam=-log(.2)  # finds the parameter of the Poisson distribution
> dpois(6,lam)
[1] 0.004827729
```

4.1.4 Standard Continuous Probability Distributions

*Uniform Distribution***.** Let *X* be a uniform random variable on the interval [*a,b*]. That is, the positive density of *X* is

$$f(x)=\frac{1}{b-a}, a\le x\le b$$

and it probability distribution function is

$$F(x)=\frac{x-a}{b-a}, a\le x\le b$$

Example 1. Suppose $X \sim U(10, 20)$, and we want to compute the probabilities $P(10<X< 15)$, $P(12< X<18)$. We can easily do this in the following:

```
> punif(15,min=10,max=20)
[1] 0.5
> punif(18,min=10,max=20)-punif(12,min=10,max=20)
[1] 0.6
```

Beta Distribution. We say that a random variable is *beta* if its probability density function is

$$f(x)=\frac{1}{B(a,b)}x^{a-1}(1-x)^{b-1}, 0\le x\le 1$$

In the above expression, the parameters a and b are nonnegative values, and $B(a,b)=\frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$. Suppose we can to compute probabilities $P(X<x_i)$, where $x_i=.1,.2,.3,.4,.5$, and $a=b=6$. Then

```
> p=c(.1,.2,.3,.4,.5)
> pbeta(p,6,6)
[1] 0.0002957061 0.0116542054 0.0782247910 0.2465018675 0.5000000000
```

gives us the desired probabilities.

Exponential Distribution. We say that the random variable X follows exponential distribution with mean θ if its probability density is

$$f(x,\theta)=\frac{1}{\theta}\exp\frac{-x}{\theta}, x\ge 0$$

If the mean of the exponential distribution is 30, and we desire to compute the percentiles at the points 13, 18, 27, and 45, we get the desired by the following program:

```
> t=c(13,18,27,45)
> pexp(t,1/30)
[1] 0.3516557 0.4511884 0.5934303 0.7768698
```

Suppose, we want to know the quantiles of an exponential distribution with mean time 30 units. We can obtain them by

```
> qexp(seq(0,1,.25),1/30)
[1]  0.000000  8.630462 20.794415 41.588831      Inf
```

Normal Distribution. A random variable *X* is said to posses normal distribution with mean μ and variance σ^2 if its probability density function is given by

$$f(x,\mu,\sigma)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),-\infty<x<\infty$$

Example. It is well-known that for a standard normal variable $(\mu=0,\sigma=1)$, *P(-1.68<X<1.68)=0.90, P(-1.96<X<1.96)=0.95,* and *P(-2.58<X<2.58)=0.99.* Let us verify these statements using *dnorm:*

```
> integrate(dnorm,-1.68,1.68)$value
[1] 0.9070427
> integrate(dnorm,-1.96,1.96)$value
[1] 0.9500042
> integrate(dnorm,-2.58,2.58)$value
[1] 0.99012
```

A comprehensive listing source for probability distributions using R may be fetched at http://cran.r-project.org/web/views/Distributions.html .

4.2 Parametric Statistical Inference

Once a probability model is finalised by the experimenter, she would like to infer from the data collected under the study. Standard inference procedures include estimation, testing and construction of confidence intervals. The standard theory for such problems is given in the classical books of Snedecor and Cochran (1989), Hogg and Craig (1978), and Freedman, Pisani, and Purves (1997). We first consider parametric inference for the suitable models, and then look at the nonparametric methods in the next section.

base and *stats* which are shipped along with the current version of R contain many of the standard statistical methods. We also use *PASWR* package for inference. In this section we focus on testing problems. The reader may refer Sheskin (2004) for a variety of tests.

4.2.1 *Z*-Test

Suppose that $x_1, x_2, \ldots, x_n$ is a random sample from $N(\mu,\sigma^2)$, and we are interested in testing for the mean $H_0: \mu=\mu_0$. If the variance σ^2 is known, we always use *Z*-test. On the other hand if the variance is unknown and the sample size is large, at least *n* =30, we use the Z test. The

Z-test is then given by

$$Z=\frac{\bar{x}-\mu_0}{\sqrt{\sigma^2/n}}$$

When the variance is unknown, we replace the variance by its estimated value.

The z-test can be obtained in the package "PASWR". This is essentially a large-sample test which can be used for one-sample and two-sample. Here, the null hypothesis is a singleton and the alternative may be one-sided or two-sided. The user has control over the value to be tested in null hypothesis as well as the nature of the alternative hypothesis. The level of significance can also be specified as per the requirement. The defaul usage of this test is

```
z.test(x, y = NULL, alternative = "two.sided", mu = 0, sigma.x = NULL,
  sigma.y = NULL, conf.level = 0.95)
```

Example. (Freund and Wilson, 2003). Suppose that the mean weight of peanuts put in jars is required 8 oz. The variance of the weights is known to be 0.03, and the observed weights for 16 jars is 8.08, 7.71, 7.89, 7.72, 8.00, 7.90, 7.77, 7.81, 8.33, 7.67, 7.79, 7.79, 7.94, 7.84, 8.17, 7.87. Here, we are interested in testing $H_0: \mu = 8.0$. We can test this in R as follows:

```
> x=c(8.08, 7.71, 7.89, 7.72, 8.00, 7.90, 7.77, 7.81, 8.33, 7.67, 7.79, 7.79,
7.94, 7.84, 8.17, 7.87)
> z.test(x,mu=8,sigma.x=.03)
        One-sample z-Test
data:  x
z = -14.3333, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 8
95 percent confidence interval:
 7.8778 7.9072
sample estimates:
mean of x
   7.8925
```

Since the *p*-value is very small, we reject the null hypothesis.

4.2.2 Tests for the Means: One- and Two- Sample *t-test*

Assume that $x_1, x_2, ...,x_n$ is a random sample from $N(\mu, \sigma^2)$ with both the parameters being unknown. Suppose we are interested in testing $H_0: \mu = \mu_0$. The parameters μ and σ^2 are respectively estimated using the sample mean and the sample standard deviation. The *t*-test is then given by

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$$

which has *t*-distribution with *n-1* degrees of freedom. The *t*-test in R software may be found in the *stats* package whose constitution is given as

```
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)
```

The following is clear from the above display:

(a) The default function is a one-sample test with a two-sided alternative being tested for $\mu = 0$ and 95% confidence interval as an output;

(b) The user has the options of specifying the nature of alternatives, μ, and the confidence interval level.

We explain the two-sample test in some detail. If $x_1, x_2, \ldots, x_{n_1}$ is a random sample from $N(\mu_x, \sigma_x^2)$, and $y_1, y_2, \ldots, y_{n_2}$ is a random sample from $N(\mu_y, \sigma_y^2)$, one is interested to test $H_0 : \mu_x = \mu_y$. The two-sample t-test is then given by

$$t = \frac{\hat{x} - \hat{y}}{s_{xy}}$$

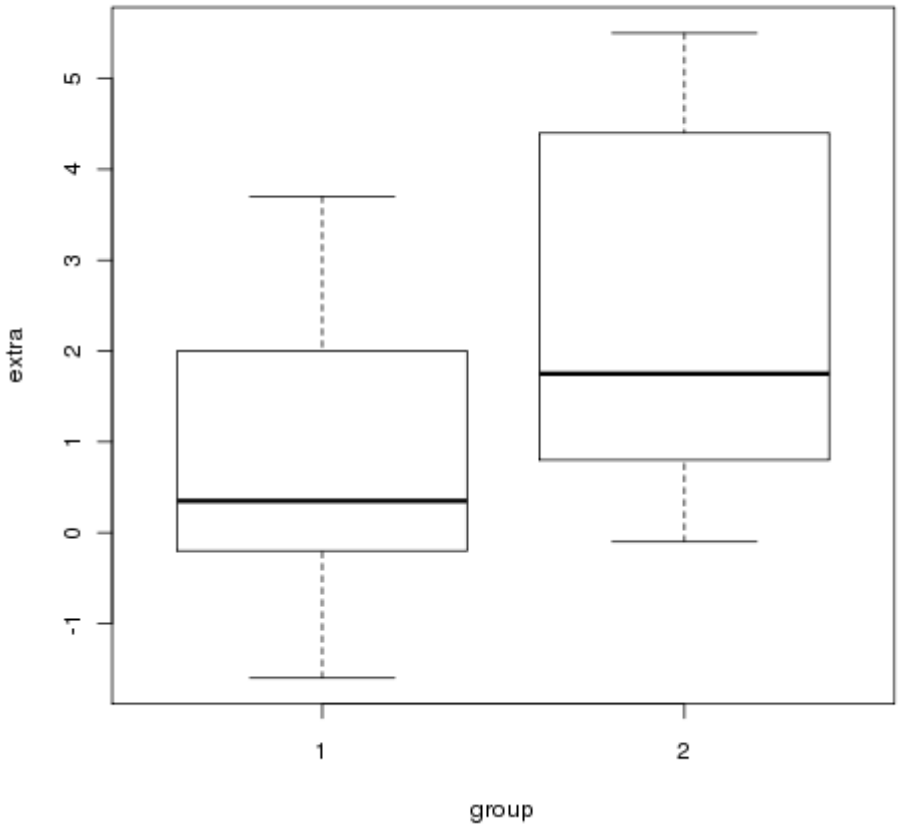
which has *t*-distribution with $n_1 + n_2 - 1$ degrees of freedom, with s_{xy} being the pooled standard deviation.

Example 1. Illustration through "*sleep*" data set in R

This data set is exactly 103 years old, and we also celebrate 100 years of its analysis by Student (Willaim Gosset) in the you-know-famous paper "*The Probable Error of Mean*". The data set is not large and this is compensated by its archiveness. As if Student would have used it if it was large! In this study a sporofic drug and control was given to 10 students each, and their respective extra sleep is noted. The data set "sleep" is given below.

```
> data(sleep)
> sleep
   extra group
1    0.7     1
2   -1.6     1
3   -0.2     1
    . . .
18   1.6     2
19   4.6     2
```

```
20   3.4     2
> plot(extra ~ group, data = sleep)
> t.test(extra ~ group, data = sleep)
      Welch Two Sample t-test
data:  extra by group
t = -1.8608, df = 17.776, p-value = 0.0794
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.3654832  0.2054832
sample estimates:
mean in group 1 mean in group 2
           0.75            2.33
```



Example 2. (Dalgaard, 2002). The objective in this study is to examine if the energy spend by *obese* men and *lean* men are same. The *t*-test is then performed as below.

```
> library(ISwR)
> data(energy)
> energy
```

```
   expend stature
1    9.21   obese
2    7.53    lean
3    7.48    lean
4    8.08    lean
     . . .
20   7.58    lean
21   9.19   obese
22   8.11    lean
```

```
> t.test(energy$expend~energy$stature)
        Welch Two Sample t-test
data:  energy$expend by energy$stature
t = -3.8555, df = 15.919, p-value = 0.001411
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.459167 -1.004081
sample estimates:
 mean in group lean mean in group obese
          8.066154           10.297778
```

Note that the values in "stature" are not numeric in nature. The *t.test* still carries out the comparison, whereas z.test from the PASWR package does not allow one to perform if the *y*-values are not numeric.

4.2.3 Testing for the Variances

The previous two sub-sections considered the problem of testing for the means of the normal distribution. Frequently it is also true that variances of the sampling distribution are not known. In such cases one is first interested to know (i) if the variance is equal to a desired value in the one sample problem, and (ii) if the variances are equal for the two-sample problem.

For a one-sample test of the variance being equal some hypothesized value, refer page 4.57-4.58 of Purohit, et. al. (2008). We consider below the two sample problem.

Example 1. Variance Test for the Two-Sample Problem.

We continue Example 2 from the previous sub-section 4.2.2. Particularly, we are interested in knowing if the variance of the variable "expend" for the two categories of "stature" are equal or not.

```
> var.test(expend~stature)
        F test to compare two variances
data:  expend by stature
F = 0.7844, num df = 12, denom df = 8, p-value = 0.6797
```

```
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.1867876 2.7547991
sample estimates:
ratio of variances
          0.784446
```

Since the p-value is not closer to zero, there is not enough evidence in the data to reject the null hypothesis that the variances are equal.

4.2.4. Testing for Proportions

Let $X_1, X_2, \ldots, X_n$ be Bernoulli trials with some unknown probability p. The test for the hypothesis $H_0 : p = p_0$ is given by

$$Z = \frac{\hat{p} - p_0}{\sqrt{p_0(1-p_0)/n}}$$

where $\hat{p} = \sum X_i / n$.

Example 1. *Testing for a single proportion.* Suppose that 432 heads turn up on a throw of 1000 times of a coin, and we would like to test if the coin is fair. Then, the following one-line command gives us the desired results:

```
> prop.test(432,1000,.5)
        1-sample proportions test with continuity correction
data:  432 out of 1000, null probability 0.5
X-squared = 18.225, df = 1, p-value = 1.963e-05
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.4011223 0.4634066
sample estimates:
   p
0.432
```

Example 2. *Testing for two proportions.* Suppose that India and Pakistan have won 23 and 16 test matches in cricket out of 62 and 58 matches respectively against the formidable Australians. We would like to test if the success percentage of India is the same as Pakistan against the one-sided alternative that the success percentage is higher. The R program takes the following form:

```
> wins=c(23,16)
> attempts=c(62,58)
> prop.test(wins,attempts,alternative="greater")
        2-sample test for equality of proportions with continuity correction
```

```
data:  wins out of attempts
X-squared = 0.8401, df = 1, p-value = 0.1797
alternative hypothesis: greater
95 percent confidence interval:
 -0.0612265  1.0000000
sample estimates:
   prop 1    prop 2
0.3709677 0.2758621
```

4.3 Nonparametric Tests

Nonparametric methods are gaining practical importance and this is reflected in the software too. Most software provide nonparametric tests as standard functions and R too has many inbuilt nonparametric tests. The related theory can be referred in the works of Gibbons and Chakraborti (2003), Wasserman (2006), and Govindrajalu (2007). The toughest nut to crack in this area is the Hajek, Sidak and Sen (1999) monograph, which is an enlarged edition of the classic Hajek and Sidak (1967). In this section we consider three of the most popular nonparametric tests, (i) Kolmogorov-Smirnov test, (ii) Wilcox test, and (iii) Kruskal test.

4.3.1 The Kolmogorov-Smirnov Test

Suppose that $X_1, X_2, \ldots, X_n$ is a random sample from a probability distribution *F,* and $Y_1, Y_2, \ldots, Y_m$ is a random sample from probability distribution *G*. If we are interested in knowing if the two samples arise from a same population, we would like to verify our intuition by testing the hypothesis $H_0 : F = G = F'(say)$. Notice that we are not making any assumptions for the distribution of *F* and *G*. In this case we need resort to nonparametric methods. We reproduce below the example given in R for the Kolmogorov-Smirnov test.

```
> x=rnorm(50);y=runif(30)
> ks.test(x,y)
        Two-sample Kolmogorov-Smirnov test
data:  x and y
D = 0.64, p-value = 9.731e-08
alternative hypothesis: two-sided
```

The reader may verify this program by running “example(ks.test)” at the R console.

4.3.2 The Wilcoxon Signed-Ranks Test

Let $X_1, X_2, .., X_n$ be iid random variables with common absolutely continuous distribution function *F*, assumed symmetric about its median, say $z_{1/2}$. Suppose that we are interested in testing $H_0 : z_{1/2} = z_0$. The Wilcoxon signed-rank test first arranges $|X_1|, |X_2|, ..., |X_n|$ in increasing order of magnitude and assign ranks 1, 2, ..., *n*, keeping track of the original signs of X_i . The test statistics then counts the sum of the ranks of positive and negative $X_i's$, say T^+ and T^- respectively, which under the null hypothesis should be nearly equal. The statistic T^+ is known as the *Wilcoxon statistic*.

If we have bivariate random samples $(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)$, we can proceed as in the one-sample case by ranking the differences $Y_1 - X_1, Y_2 - X_2, ..., Y_n - X_n$. In general, the two-sample Wilcoxon test is based on replacing the data with their group, ignoring their group, and then calculating the sum of ranks in one group.

Example 1. (*Revisiting the Example 2 in Section 4.2.2*)

The Wilcoxon test for the energy spent by "*lean*" and "*obese*" persons in R is given below:

```
> wilcox.test(expend~stature)
        Wilcoxon rank sum test with continuity correction
data:  expend by stature
W = 12, p-value = 0.002122
alternative hypothesis: true location shift is not equal to 0
Warning message:
In wilcox.test.default(x = c(7.53, 7.48, 8.08, 8.09, 10.15, 8.4,  :
  cannot compute exact p-value with ties
```

4.4 Bayesian Inference

Students and beginners in R will find an excellent computational introduction for the Bayes paradigm in the recent book of Albert (2007) titled "Bayesian Computation with R", who is also the creator of the R package "LearnBayes". The learned statistician may directly begin with Marian and Robert (2007) "Bayesian Core" . This section is mainly drawn from Albert's book. An early account of the Bayes theory may be found in the treatises of Berger (1985), Bernardo and Smith (1994), and modern account of it is contained in Robert (2002) and Ghosh, Delampady, and Samanta (2006). Gelman, et. al. (2004) and Carlin and Louis (2000) are very good accounts of the Bayes theory applications. An elementary introduction to data analysis in the Bayesian paradigm is also available

in Silvia (2006). The authors have found the graduate-level book of Bolstad (2007) as the best starting point for learning Bayesian theory. Another reason, of course, is that some computations have been done in this book using R and are available with the package "Bolstad".

We begin this section with the Bayes formula.

4.4.1 The Bayes Formula

Let Ω be the sample space, and let $B_1, B_2, \ldots, B_m$ be a collection of mutually exclusive and exhaustive events from Ω, that is, $B_i \cap B_j = \phi$, and $, \cup_{i=1}^{m} B_i = \Omega$. Suppose $P(A/B_i) > 0, i = 1, 2, \ldots, m$, then the Bayes probability is given by

$$P(B_i/A) = \frac{P(A/B_i)P(B_i)}{\sum_{i=1}^{m} P(A/B_i)P(B_i)}.$$

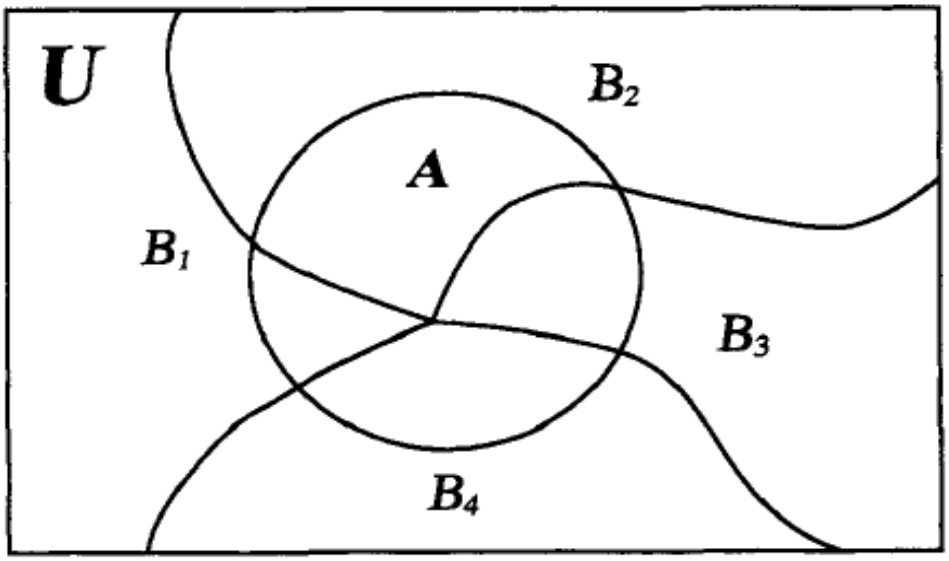


Figure 4.3. Venn Diagram for Bayes Formula

The above formula can be easily proved:

$$\begin{aligned} P(B_i/A) &= P\frac{(B_i \cap A)}{P(A)} \\ &= \frac{P(A/B_i)P(B_i)}{\sum_{i=1}^{m} P(A \cap B_i)} \\ &= \frac{P(A/B_i)P(B_i)}{\sum_{i=1}^{m} P(A/B_i)P(B_i)} \end{aligned}$$

We now illustrate the Bayes formula.

Example. (Horgan, 2008). A manager from the Human Resource Department of an XYZ company notes that 50% of the documents in the company are written in Word, 30% in Latex, and 20% in Html format. She also observes that the proportion in each of these formats exceeding 10 pages for Word, Latex, and Html are respectively 40%, 20% and 20%. We denote the file format categories by A_W, A_L, A_H respectively for the Word, Latex, and Html formats. Thus, $P(A_W)=0.5, P(A_L)=0.3, P(A_H)=0.2$.

Let E be the event that a document selected at random is more than 10 pages. From the information available it is clear that $E=(A_W \cap E)\cup(A_L \cap E)\cup(A_H \cap E)$. Further, $P(E/A_W)=0.4, P(E/A_L)=0.2, P(E/A_H)=0.2$. Thus,

$$\begin{aligned} P(E) &= P(A_W)P(E/A_W)+P(A_L)P(E/A_L)+P(A_H)P(E/A_H) \\ &= (0.5\times 0.4)+(0.3\times 0.2)+(0.2\times 0.2)=0.2+0.06+0.04 \\ &= 0.3 \end{aligned}$$

Though not really required, we have an excuse to run towards R.

```
> p_aw=0.5; p_al=0.3; p_ah=0.2 # probability of causes
> p_e_g_aw=0.4; p_e_g_al=0.2; p_e_g_ah=0.2 #probability of event given
                                          causes
> p_e=p_aw*p_e_g_aw + p_al*p_e_g_al + p_ah*p_e_g_ah # total
                                        probability of the event
> p_e # all for this thug
[1] 0.3
```

Naturally, we can now answer the probabilities of a document exceeding 10 pages is of the file format is Word, or Latex, or Html as follows:

```
> p_aw_g_e = (p_aw * p_e_g_aw)/ p_e # prob of word document given
                                        pages exceeded 10
> p_aw_g_e
[1] 0.6666667
> p_al_g_e = (p_al * p_e_g_al)/ p_e # prob of latex document given pages
                                        exceeded 10
> p_al_g_e
[1] 0.2
> p_ah_g_e = (p_ah * p_e_g_ah)/ p_e # prob of html document given pages
                                        exceeded 10
> p_ah_g_e
[1] 0.1333333
```

4.4.2 The Bayesian Paradigm

Suppose that we have a random sample from a probability distributions $F(.,\theta), \theta\in\Theta$. As mentioned in Robert (2002), page 9, Bayes and Laplace further assume that the uncertainty about

parameter θ can be modeled with some probability distribution π on Θ. The probability distribution π is referred as *prior distribution.* Using the Bayes formula, we get

$$\pi(\theta/x)=\frac{f(x/\theta)\pi(\theta)}{\int f(x/\theta)\pi(\theta)d\theta}.$$

The distribution $\pi(\theta/x)$ is known as *posterior distribution.* We begin with a simple example.

Example 1. (Albert, 2009). Proportion of Heavy Sleepers.

A statistician learns that doctors recommend eight hours of sleep a day. Let p denote the proportion of who sleep at least eight hours. If x persons in a survey of size n get the recommended sleep, the likelihood function is given by

$$L(p)\propto p^{x}(1-p)^{n-x}$$

It is believed that the plausible values of p are .05, .15, .25, .35, .45, .55, .65, .75, .85, .95, and the statisticians belief on the occurance of these values are respectively 1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0, 0.

The belief on the plausible values can be translated into probabilities by dividing each value by their total sum. That is,

```
> p = seq(0.05, 0.95, by = 0.1) # plausible values of the parameter
> prior = c(1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0, 0) # belief in plausible values
> prior = prior/sum(prior) # translating the belief into probability
```

The “pdisc” function from the “LearnBayes” package computes the posterior probabilities. Suppose that it is found in a survey of the statistician that 23 out of 68 interviewed subjects had the required sleep of 8 or more hours.

```
> library(LearnBayes)
>  data = c(23, 45)
>  post = pdisc(p, prior, data)
>  round(cbind(p, prior, post),2)
       p prior post
 [1,] 0.05  0.03 0.00
 [2,] 0.15  0.18 0.00
 [3,] 0.25  0.28 0.21
 [4,] 0.35  0.25 0.71
 [5,] 0.45  0.16 0.08
 [6,] 0.55  0.07 0.00
 [7,] 0.65  0.02 0.00
 [8,] 0.75  0.00 0.00
 [9,] 0.85  0.00 0.00
[10,] 0.95  0.00 0.00
```

Note that two of the maximum posterior probability is concentrated on the values p = .25 and p = .35.

4.4.3 Inference for Binomial Distribition

The binomial probability function is

$$p(x/\theta)=\theta^{x}(1-\theta)^{1-x}$$

The range of the paramater p is the unit interval [0,1]. The *non-informative prior* on p is the standard uniform distribution *U[0,1]*. That is, $\pi(\theta)=1, 0\le p\le 1.$ Thus, if we have n observations from $p(x/\theta),$ the posterior distribution of p is given by

$$\pi(\theta/\boldsymbol{x})\propto p(\boldsymbol{x}/n,\theta)\pi(\theta)$$
$$=\theta^{\sum_i^n x_i}(1-\theta)^{n-\sum_i^n x_i}$$

Example 1. We simulate the data from Bernoulli distribution with probability of success p = 0.65, and analyze the properties of Bayes estimator under different priors in the following program. The theme is essentially to show that data washes away all the prior information as the sample size increases. This is evident as we move across the graphs.

```
# R Program for Bayesian Inference of Binomial Distribution Using Different Priors
x=seq(0,1,0.01)
ydata=rbinom(4096,1,.65)
no.of.trials=c(1,2,3,4,8,16,32,64,128,256,512,1024,2048,4096)
successdata=matrix(data=NA,nrow=14,ncol=2)

for(i in 1:14)
 successdata[i,1]=sum(window(ydata,1,no.of.trials[i]))
 successdata[,2]=no.of.trials

flatprior=x*0+1

par(mfrow=c(5,3))
plot(x,flatprior,xlim=c(0,1),ylim=c(0,1),"l")

for(i in 1:14)
{temp=max(dbeta(x,successdata[i,1]+1,successdata[i,2]-successdata[i,1]+1))
plot(x,dbeta(x,successdata[i,1]+1,successdata[i,2] – successdata[i,1]+1),
xlim=c(0,1),ylim=c(0,temp),"l",ylab="posty")
      }

#Comparison with different priors
par(mfrow=c(5,3))
```

```
plot(x,flatprior,xlim=c(0,1),ylim=c(0,1),"l")
       points(x,dbeta(x,2,2),"l",col="red")
       points(x,dbeta(x,.5,.5),"l",col="pink")

for(i in 1:14)
{temp=max(dbeta(x,successdata[i,1]+1,successdata[i,2]-successdata[i,1]+1))
plot(x,dbeta(x,successdata[i,1]+1,successdata[i,2]-successdata[i,1]+1),
xlim=c(0,1),ylim=c(0,temp),"l",ylab="posty")
       points(x,dbeta(x,successdata[i,1]+2+1,successdata[i,2] -successdata[i,1]+2+1),"l",col="red")
      points(x,dbeta(x,successdata[i,1]+.5+1,successdata[i,2]-successdata[i,1]+.5+1),"l",col="pink")
```

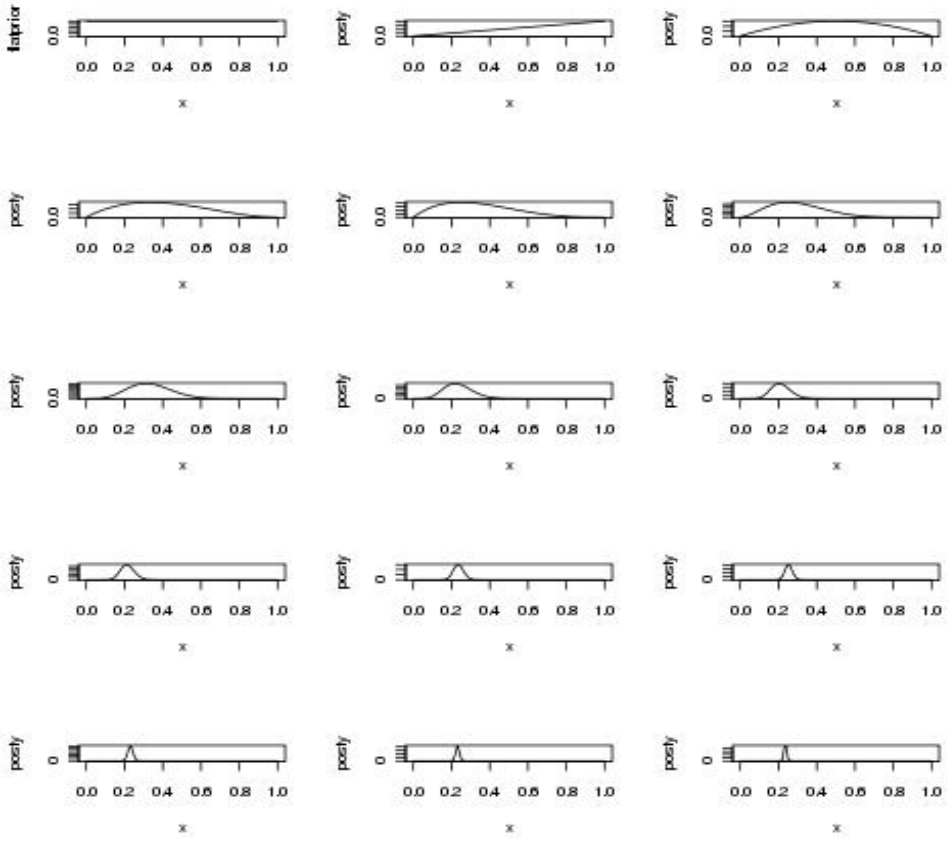


Figure 4.4 Inference for Binomial Distribution Using a Non-informative Prior

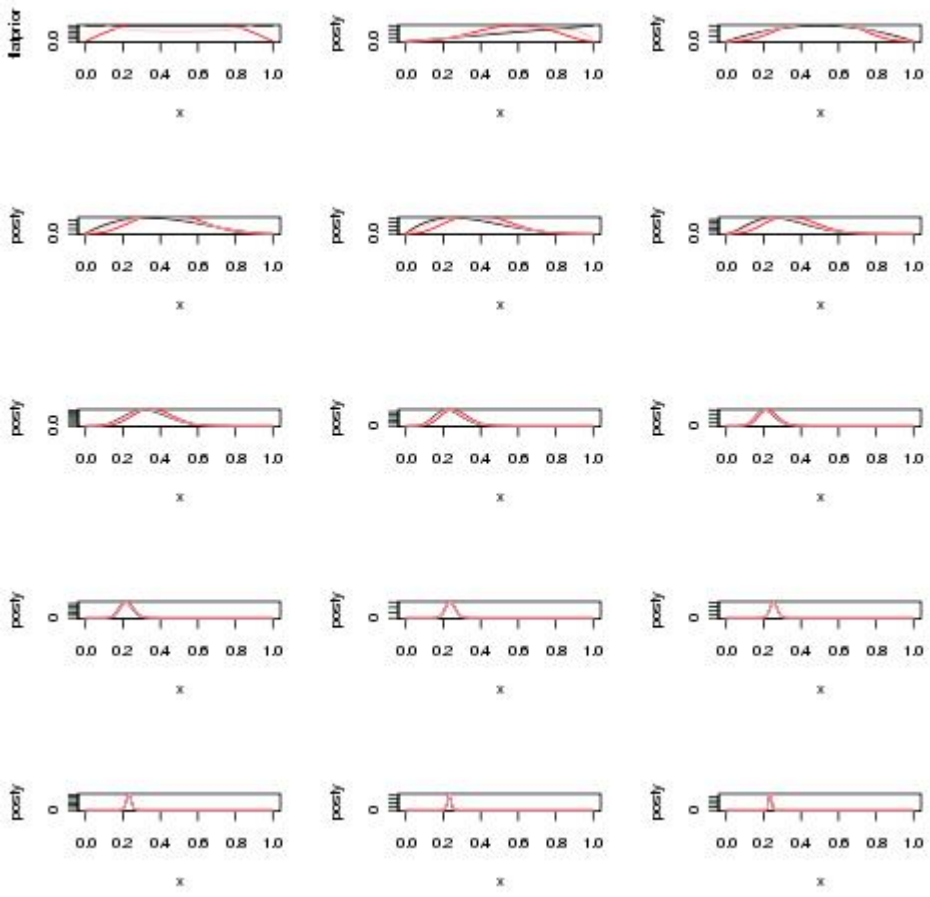


Figure 4.5 Inference for Binomial Distribution Using Different Priors

* * * * *

5. Simulation

Package(s) :

Data set(s) :

5.1 Introduction

Simulation of observations from random experiments has been of profound interest. Generating of random numbers has always been crucial for statisticians, both for theory and applications. The emergence of computers and its speed has greatly benefitted in the endure for the need of random numbers. The truth to be told machines can not really produce random numbers in the sense that the generated sequence will eventually repeat itself identically, which is more formally called as *cycle* of the generator. However this does not restrict their usage for most practical situations if the required random numbers is less than the cycle of the generator. See also the important page : http://www.cran.r-project.org/web/views/Distributions.html

About the function RNGkind and RNGversion, set.seed

George Marsaglia

Ross (2006) Gentle (2009), Gentle (2003), Liu (2002)

5.2 Generating the (Pseudo-)Random Numbers

Generation of random numbers have a lot of applications in real life. The 14-digit number that is written on your mobile recharge voucher, numbers on be-lucky-to-win scratch cards, match your mobile numbers on those given in a series of draws of newspaper, etc, are all some scenarios which require generating random numbers. Thus, there is *that* need of a *random number generator* (RNG) which ensures that the numbers in the draw look like random. This is purely in the sense that there is no systematic pattern in the sequence of generated numbers. Of course, when one of the authors suggested that the last two digits of a phone number from the telephone directory (or yellow pages) can be used as an RNG, the proposal was shoot down citing the practical difficulty of carrying those over 5 kgs book weight while conveniently overlooking the realistic part that they are available almost everywhere.

It used to be a difficult task to generate random numbers three decades ago. Computers have eased this job for us just as they aided in a billion other problems.

A powerful and fairly straight-forward algorithm for generating pseudo-random numbers is the *linear congruential generator,* abbreviated LCG. In this algorithm, we start with an initial number x_0, a multiplier *a*, an incremental value *c*, and a modulo *m*. Using these terms, we can obtain a sequence of pseudo-random numbers x_n, using the form of linear congruential generator:

$$x_n = (a\,x_{n-1} + c) \bmod m,\ 0 \le x_n \le m.$$

For large values of *m*, the sequence $\{x_n\}$ generated by the LCG behaves like a sample from the *uniform distribution* $U(0,m)$. Further, the sequence $\{x_n/m\}$ resembles a random sample from the standard uniform distribution $U(0,1)$. We can see this from the simple program below.

```
> m=2^20; a = 123.89; c = 39.75
> x=c()
> x[1] = 4567
> for(i in 2:10001){
+ x[i]=(a*x[i-1]+c) %% m
+ }
> tiff("LCG_Hist.tiff")
> par(mfrow=c(2,1))
> hist(x,xlab="x values", ylab="Frequency")
> hist(x/m,xlab="normalised x values", ylab="Frequency")
> dev.off()
null device
          1
```

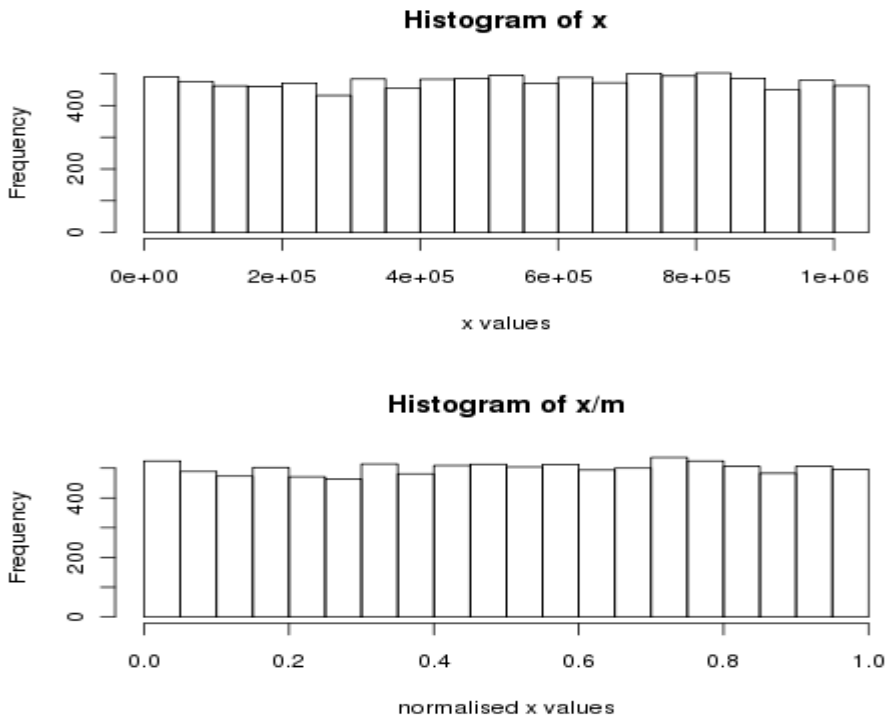


Figure 5.1. Histogram of 10000 Observations Obtained from the LCG

For more details about the LCG, refer and read pages 11-27, Chapter 1 of Gentle (2003). We next move to simulation from standard discrete distributions.

5.3 Simulation from Discrete Distributions

As mentioned earlier in Chapter 4, the wait for the fourth of the quartert (p, d, q, r) for a probability distribution ends here. The letter "r" prefixed with the abbreviation of a probability distribution enlisted in R helps us to generate pseudo-random observations from the target distribution.

*Discrete Uniform Distribution***.** If a random variable X can take one of the N possible distinct outcomes with equal probability, we say that X has a *discrete uniform distribution.* A simple example of such a variate is if one randomly selects a key from a bunch, then the probability of getting the right key for unlocking follows a discrete uniform distribution. Mathematically,

$$P(X=x_i)=\frac{1}{N}, i=1,2,...,N \quad .$$

Example 1. Suppose we want to generate 100 numbers randomly from the sequence 1, 2, ..., 10. Then, the random numbers are given by

```
> dud<-sample(c(1:10),100,replace=T)
> table(dud)
dud
 1  2  3  4  5  6  7  8  9 10
 8  8 12 12 14  5 11  8  9 13
```

Binomial Distribution. Suppose X denotes a binomial random variable with parameters (n,p), we can generate random observations from it using the function *rbinom.* If we want to generate 10 Bernoulli observations of a fair coin, we obtain them as

```
> rbinom(10,1,.5)
 [1] 0 1 0 1 1 1 1 0 1 0
```

Example 2. Suppose that $X \sim B(20,.3)$, and we want to estimate $P(X>13)$ based on 100 simulated observations. We can get the desired in the following:

```
> set.seed(123)
> sum(rbinom(100,20,.3)>13)/100
[1] 0
```

We can verify the approximateness of our answer by actually computing $P(X>13)$:

```
> c(14:20)->x
```

```
> sum(dbinom(x,20,.3))
[1] 0.000261047
```

Since the truth *P(X>13)* is very close to zero, we see that the simulation method of estimating *P(X>13)* just works fine.

Poisson Distribution. We use the command "rpois" for simulation of observations from a Poisson random variable with mean rate λ .

Example 3. Suppose *X ~ P(10),* and *Y ~ P(20).* We can then estimate *P(X>Y)* based on 1000 observations as follows:

```
> x=rpois(1000,10); y=rpois(1000,20)
> sum(x>y)/1000
[1] 0.036
```

We leave as an exercise to verify this result using the function *dpois. (Hint: Stop computations once dpois(x,* λ *) is approximately equal to zero).*

Simulation from standard distributions is no longer a big deal.

5.3 Simulation from Continuous Distributions

We begin with simulation from the uniform distribution which also forms the basis for generating samples from all continuous distributions.

Continuous Uniform Distribution. The probability density function of a uniform random variable over the interval (a,b) is

$$f(x)=\frac{1}{b-a}; a \le x \le b$$

Simulation from this distribution is particularly simpler, and we first look at the syntax of the function *runif:*

```
runif(n, min=0, max=1)
```

By default, we are generating *n* observations from the unit interval, and we can change this using

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

We had seen in Section 2 that the LCG returns us sample from uniform distribution. It is left as an excersize to the reader to figure out the exact algorithm used in "runif". Remember, R is an open source software and you can see every line of code in it.

As an illustration, we simply use the built-in example of R.

```
> example(runif)
runif
> var(runif(10000))  #- ~ = 1/12 = .08333
[1] 0.08353145
```

The Inverse Cumulative Distribution Function (CDF) Method. Simulation of samples from continuous distributions easily follows from the simulated values from the standard uniform distribution thanks to the following result.

Lemma. If *X* is a random variable with continuous CDF F_X and $U =: F_X(X)$, then

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

Consequently, if we need samples from the distribution F_X, we can use the inverse relation

$$X = F_X^{-1}(U)$$

for generating the samples from the required distribution.

Exponential Distribution. A random variable *X* follows exponential distribution with mean θ if its probability density is

$$f(x,\theta) = \frac{1}{\theta} e^{(-\frac{x}{\theta})}, x \geq 0.$$

The CDF of exponential distribution is then

$$F_X(x,\theta) = 1 - e^{(-\frac{x}{\theta})}, x \geq 0.$$

If *U* is an observation from *U*(0,1), we have by the discussion following the previous lemma that

$$F_X^{-1}(U)$$

is a observation from $F_X(x,\theta)$. That is, a pseudo-observation from $F_X(x,\theta)$ may be generated in the following steps:

$$F_X(x,\theta) = U$$
$$1 - e^{(\frac{-x}{\theta})} = U$$
$$e^{(\frac{-x}{\theta})} = 1 - U$$
$$x = -\theta \times \log(1-U)$$

We can examine the correctness of the above steps using the following program:

```
> n = 100; theta = 50
> pseu_unif=runif(n)
```

```
> x=-theta*log(1-pseu_unif)
> tiff("Exponential_Distribution.tiff")
> hist(x,freq=F,ylim=c(0,.012))
> curve(dexp(x,rate=1/theta),add=T)
> dev.off()
null device
          1
```

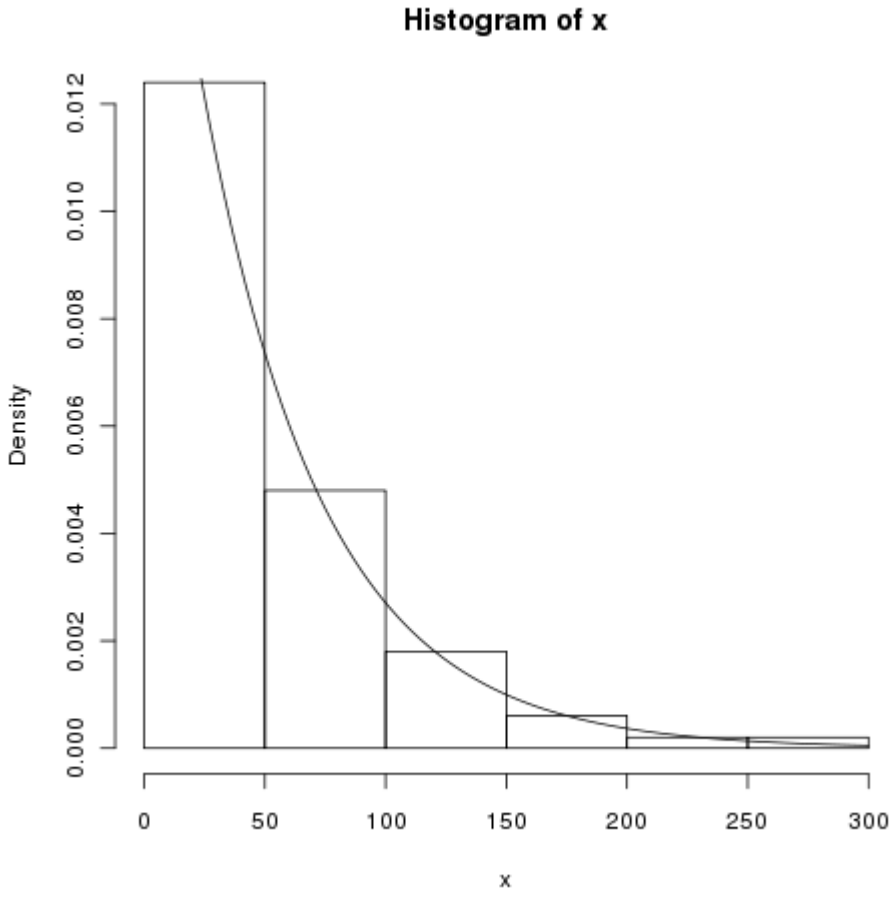


Figure 5.2. Histogram of Simulated Observations from Exponential Distribution and its Density Plot

Of course, we can easily generate observations from the exponential distribution in R using the function "rexp".

```
> (rexp(10,1/theta))
 [1]  9.998896 51.222067 51.393481 14.233423 78.152594  2.104415  4.931544
 [8]  4.928466 14.019258 14.789348
```

5.4 Understanding Limit Theorems Through Simulation

Limit theorems in a course on probability theory is often intimidating in the sense that

though the probability of a person reaching infinity is zero we have to accept the result for the only reason that applying calculus yields the truth answer in infinetly often cases with probability one. Yes, *measure theory* just helps the cause even further and lets refrain ourself from using these infinetly often used techniques. Here, in this section though we can't take the reader to infinity, we give him a very clear picture of what its like to be at infinity. In short, we really show whats happening in the statements of the nature:

$$\begin{gathered} \textit{Let } T_1, T_2, \ldots, T_n, \ldots \textit{ be a random sample from a probability distribution } F_\theta . \\ \textit{We say that the sequence } \{T_n\} \textit{ is uniformly consistent for } \theta \textit{ if} \\ T_n - \theta \to^P 0, \textit{as } n \to \infty . \end{gathered}$$

Example 1. The Poisson Approximation (or just approximations) for Binomial Distribution.

If the number of Bernoulli trials is more than 20, i.e. $n \geq 20,$ *a*nd the probability of success is less than 0.05, the probability of x successes may be well approximated by a Poisson random variable instead of the natural binomial distribution. The thumb rule for this approximation is that $np \approx 1$. This is such a useful approximation that it is bound to be found in almost ever book on the first course in statistics or probability. To see how this approximation works, we first look at the probability mass functions of both binomial distribution and the approximating Poisson distribution.

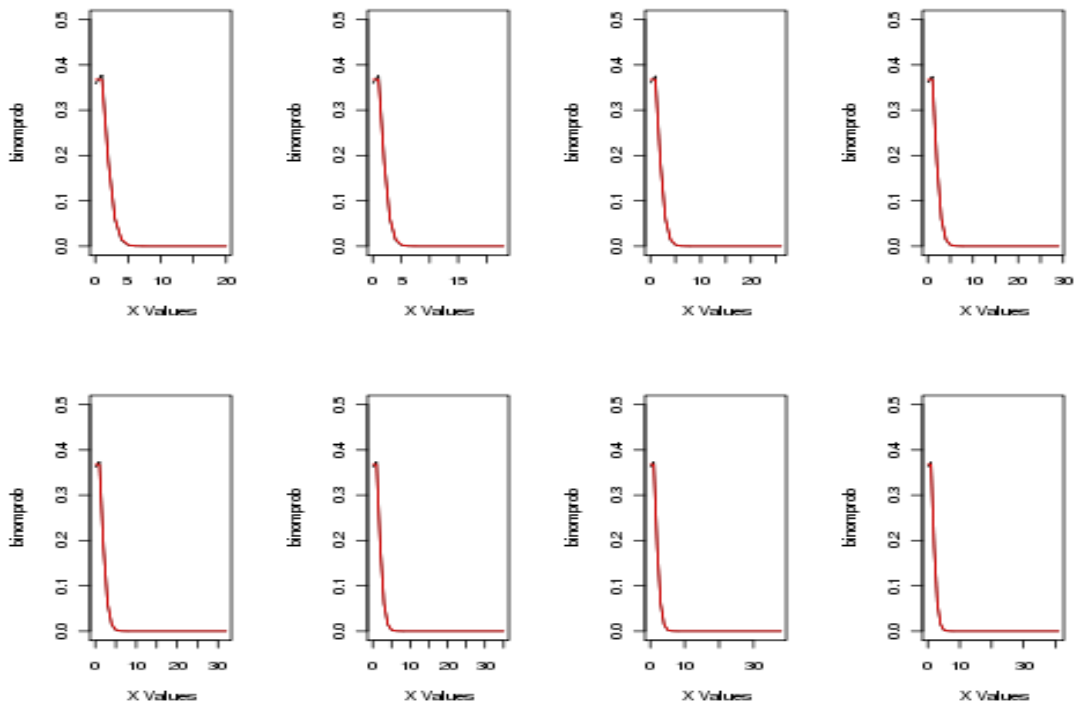


Figure 5.3. Poisson Approximation when np is approximately equal to 1

```
> n=seq(20,41,3)
> p=1/n
> approxdiff=n*0
> par(mfrow=c(2,4))
> for(i in 1:length(n)){
+ binomprob=dbinom(c(0:n[i]),n[i],p[i])
+ poisprob=dpois(c(0:n[i]),n[i]*p[i])
+ plot(c(0:n[i]),binomprob,ylim=c(0,.5),"l")
+ lines(c(0:n[i]),poisprob,ylim=c(0,.5),"l",col="red")
 + approxdiff[i]=sum(binomprob-poisprob)}
> approxdiff # gives the cumulative sum of difference in approximation
[1] -3.768953e-17 -1.769346e-17 -2.677794e-17  1.163295e-17  4.641127e-17
[6]  4.823248e-17  5.245738e-17  1.226275e-17
```

As the above plot reveals, there is hardly any difference in the probability curves of the binomial distribution and its approximating Poisson distribution. The scenario when *np* is approximately equal to 1 is clear. However, we need to address what happens when this is not the case.

In the above program, the parameter of the Poisson distribution was kept agonisingly closer to 1. Now, we deviate it a little in the range of 0.25 to 0.5 and re-plot the whole thing (we just change *p* to 10*(1/n) in the above program, and this program is left as an excersise to the reader).

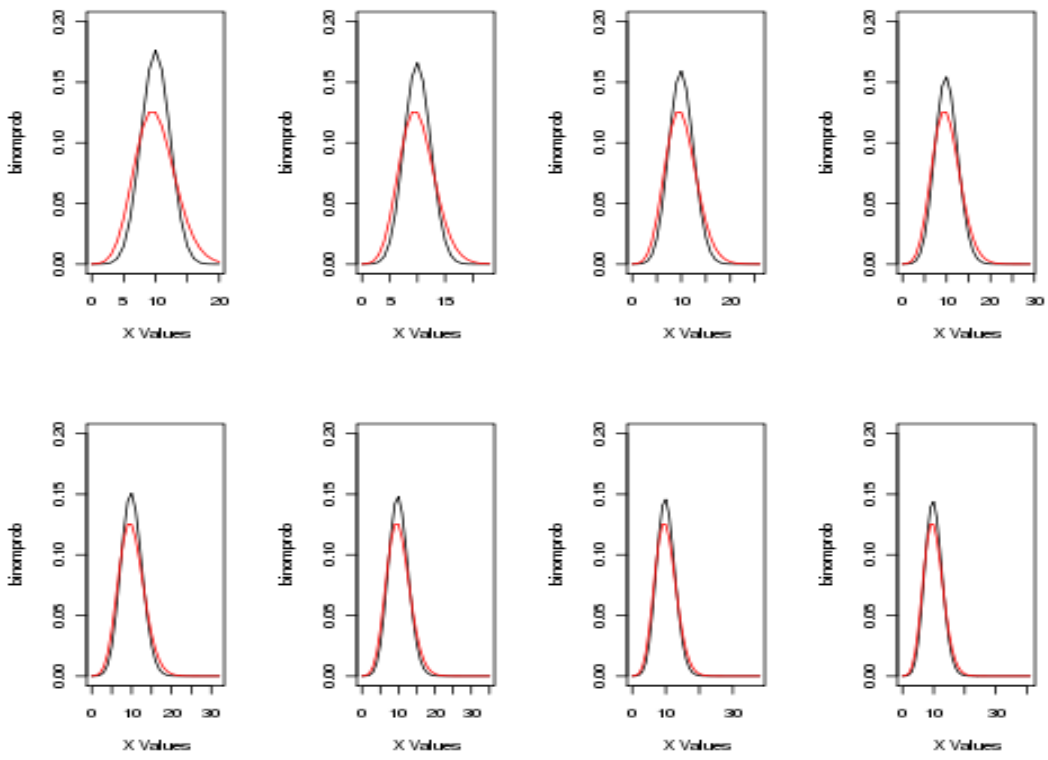


Figure 5.4. Poisson Approximation for *p* in the range (0.25, 5)

In this changed scenario too, we see that the approximations continue to hold good.

Example 2. Convergence of Uniform Minima

Let $X_1, X_2, \ldots, X_n$ be a random sample from a uniform distribution $U(0,\theta)$. Suppose, we are interested in the asymptotic behavior of $nX_{(1)}$, where $X_{(1)} = min(X_1, X_2, \ldots, X_n)$ as a function of *n*.

Theoretically, the survival function (complement of the cumulative distribution function) is given by

$$\begin{aligned} P(nX_{(1)} > x) &= P(X_{(1)} > x/n) \\ &= \prod_{i=1}^{n} P(X_i > x/(n\theta)) \\ &= \left(1 - \frac{x/\theta}{n}\right)^n \end{aligned}$$

Taking limits on both sides, the asymptotic distribution of the survival function of $n\,X_{(1)}$ is thus the following:

$$\begin{aligned} \lim_{n\to\infty} P(nX_{(1)} > x) &= \lim_{n\to\infty} \left(1 - \frac{x/\theta}{n}\right)^n \\ &= \exp(-\frac{x}{\theta}) \end{aligned}$$

The expression on the second line is the survival function of exponential distribution with mean θ.

We conduct a simulation study to understand the same using the following algorithm.

Step 1. Consider sample sizes *n = 100, 200, ..., 800.*

Step 2. For a given sample size, generate *n* observations from $U(0,\theta)$ and note the minimum of this sample and calculate $n\,X_{(1)}$.

Step 3. Repeat Step 2 a large number of times, say 20.

Step 4. Plot the empirical cumulative distribution function for each sample size *n,* and also plot the empricial distribution of exponential random variable with mean θ.

The graphs generated gives us a clear understanding of asymptotic distribution.

We see from the Figure 5.5 that as the sample size increases, the convergence of the minima of uniform distribution is closer to that exponential distribution.

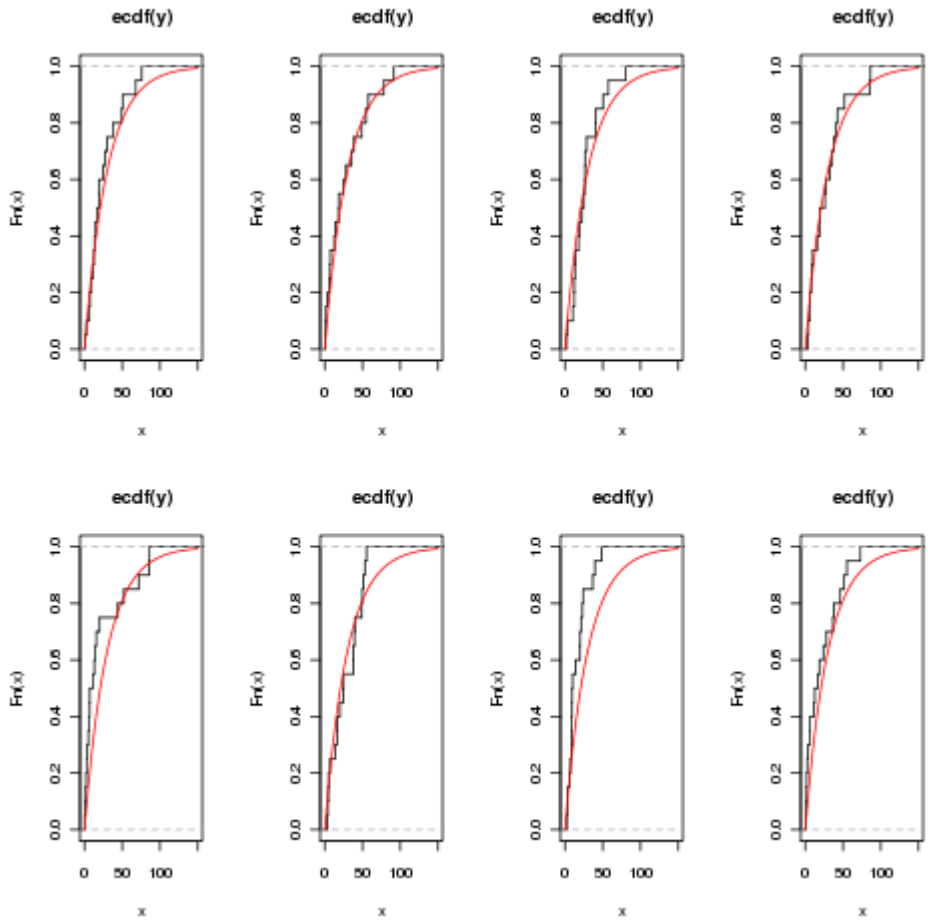


Figure 5.5. Simulation Study for Convergence of Uniform Minima

The R codes producing the above diagram is listed below.

```
con <- function () {
> theta=30
> nsimul=20
> ss=c(10,20,50,100,200,400,600,800)
> xpoints=seq(0,100,5)
> par(mfrow=c(2,4))
+         for(i in 1:length(ss)) {
+                                 y=c()
+                                  for(j in 1:nsimul) {
+                                     y[j]=ss[i]*min(runif(ss[i],0,theta))
+                                                    }
+                           plot(ecdf(y),verticals=T,do.points=F,xlim=c(0,150))
+                           y=y[order(y)]
+                           z=seq(0,150,5)
+                           lines(z,pexp(z,1/theta),type="l",col="red")
+                           }
+                           }
> con()
```

Example 3. *Understanding the Weak Law of Large Numbers (WLLN)*. The WLLN states that if $X_{1,}\ldots,X_n$ are i.i.d. Random variables with $E(X_i)=\mu$, and $Var(X_i)=\sigma^2<\infty$, then as $n\to\infty$

$$\frac{1}{n}S_n=\frac{1}{n}\sum_{i=1}^{n}X_i\to^P\mu.$$

In the above expression $\to^P$ indicates convergence in probability.

To understand the WLLN through simulation, we consider three disbutions, viz., normal, exponential, and gamma, for which the variance is theoretically known to be finite. We then generate 10000 values from each of these distributions with means respective at 5, 6.5, and 8. The below program along with the graph gives a clear understanding of how true the WLLN holds in reality.

```
> n=10000
> xnorm=rnorm(n,5,1); xexp=rexp(n,1/6.5); xgamma=rgamma(n,4,1/2)
> tiff("wlln_simul.tiff")
> plot(1:n,cumsum(xnorm[1:n])/1:n, type="l", main=substitute("Convergence of sample
mean to " *mu,list(1)),col = "red",ylim=c(4,9))
> lines(1:n,cumsum(xexp[1:n])/1:n,type="l",col="blue")
> lines(1:n,cumsum(xgamma[1:n])/1:n,type="l",col="green")
> abline(h=c(5,6.5,8),lty=2)
> dev.off()
null device
          1
```

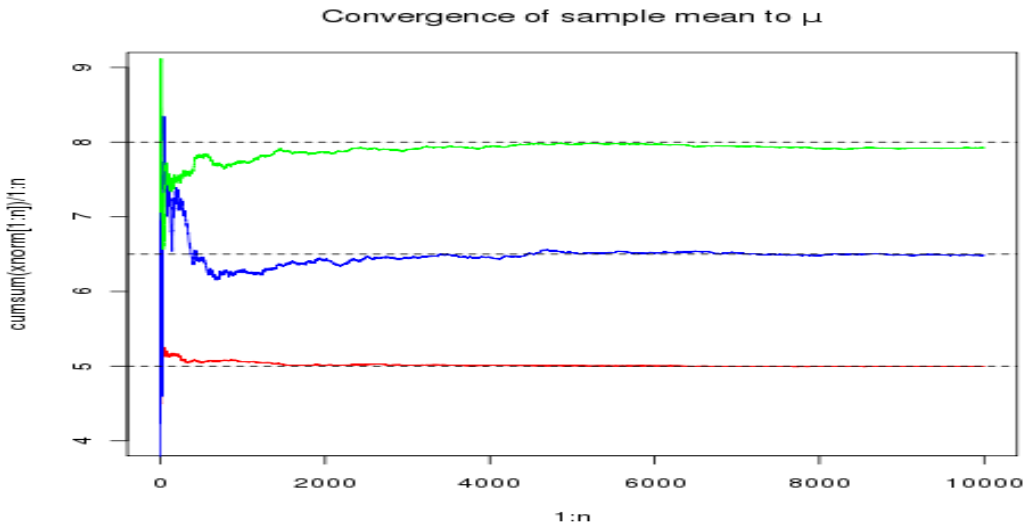


Figure 5.6. A Simulation Study for Understanding WLLN

* * * * *

6. Regression Models

Package(s) : faraway

Data set(s) : rocket_propellant.csv, anscombe, uscrime.csv

6.1 Introduction

Faraway (2002) is probably the first detailed account on the use of R for linear models. Interestingly, this book is allowed to be freely circulated and one may also print it and sell at a cost covering the cost of print. This book makes an elegant reading even today for the R 2.8.1 version, though it was written when the R version was in the early 1.x versions. Faraway (2006) is an extended version which considers the *generalized linear models* which we deal with in Section 6. 8. Fox (2002) deals with regression problems in both R and S-plus. Sheather (2009) is also a very recent account of the use of R for analysis of linear models, and SAS users will also find it easier to use this book as it also gives parallel programs of it. Ritz and Streibig (2008) is dedicated to the applications of *nonlinear regression models* using R.

6.2 Linear Regression Models: Simple Regression

Suppose that Y is the variable of interest and that it is dependent on some covariates $\boldsymbol{X}=(X_1,\ldots,X_k)$. The covariates are also sometimes called as explainatory variables, or regressors, or predictors. In general the covariate is an independent variable. The output Y is called as regressand. The general linear regression model is given by

$$Y=\beta_0+\beta_1 X_1+\ldots+\beta_k X_k+\epsilon. \tag{6.1}$$

That is, the regressors are assumed to have a linear effect on the regressand. The vector $\boldsymbol{\beta}=(\beta_0,\beta_1,\beta_2,\ldots,\beta_k)$ is known as the vector of *regression coefficients.* If k = 1 the model is known as *simple regression model,* and if $k > 1$ it is called as a *multiple regression model. The* values of the $\beta_j,\ j=1,\ldots,k$ are completely unspecified and take values on the real line. A positive value of them indicates positive correlation and a negative value indicates negative correlation. If all the covariates have value 0, β_0 is the average value of *Y.* We refer β_0 as the intercept term. In a simple regression model, the regression coefficients has this interpretation: If the regressor is changed by one unit, the change in *Y* is β. We illustrate the concepts in this section using a simple regression model.

For the *i*-th individual in the study, the model is given by

$$Y_i = \beta_0 + \beta_1 X_{i1} + ... + \beta_k X_{ik} + \epsilon_i.$$

The errors ϵ_i are assumed to be independent and identically distributed as normal distribution $N(0, \sigma^2)$, with unknown σ^2. We assume that we are having a sample of size *n*. In this section we focus on the simple regression model.

Example 1. The Rocket Propellant Data. We draw this data set from Montgomery, et al. (2003). In this study a rocket motor is manufactured by bonding together an igniter propellant and a sustainer propellant. The variable of interest here is the shear strength (measured in psi units) of the bond between these two propellants. The manufacturer is interested to know the effect of the age of the rocket propellant *X* on the shear strength of a material *Y*. In simple words, the model being studied is

Shear Strength = Baseline + (Effect of Age of Propellant)* (Age of Propellant) + Error

and in symbols it is

$$Y = \beta_0 + \beta_1 X + \epsilon$$

The data from Montgomery, et al. (2003) is given in the file "rocket_propellant.csv", and we promptly read it in R:

```
> rp=read.csv("rocket_propellant.csv",header=T)
```

We first plot the scatter plot:

```
> tiff("rocket.tiff")
> plot(Age_of_Propellant, Shear_Strength)
> dev.off()
```

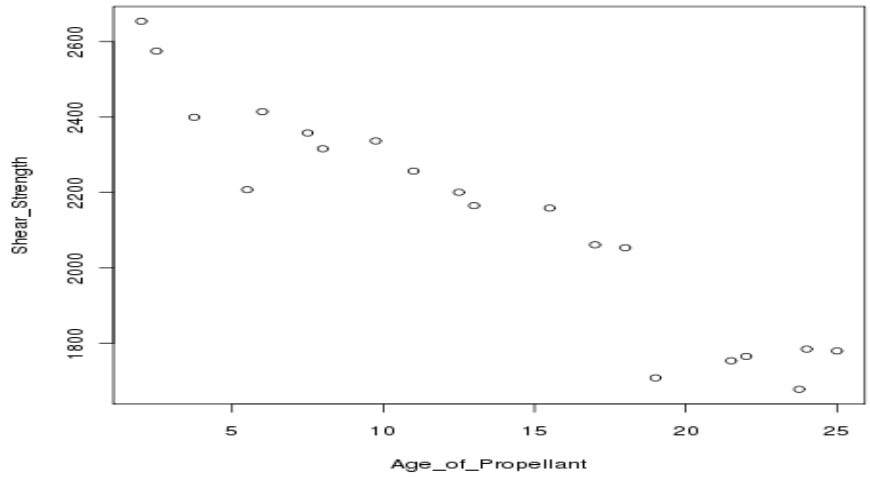


Figure 6.1. Scatter plot of Shear Strength vs Age of Propellant

It is apparent from the scatter plot that as the shear strength increases the age in propellant is decreasing, and thereby indicating a negative correlation.

The main problem in a simple linear regression model is estimation of the unknown vector regression coefficients $\boldsymbol{\beta}=(\beta_0,\beta_1)$ and the variance of the error term σ^2. Once we have estimated the parameters, we can use the regression model for prediction purposes.

For many statistical reasons, the parameters are estimated using the *least squares method.* The least-squares criterion is

$$S(\beta_0,\beta_1)=\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i)^2$$

Differentiation the above equation with respect $\boldsymbol{\beta}=(\beta_0,\beta_1)$ and equating them to zero gives us the least-squares normal equations, and solving them further gives us estimators for $\boldsymbol{\beta}=(\beta_0,\beta_1)$:

$$\hat{\beta}_1=\frac{S_{xy}}{S_{xx}}, \tag{6.2}$$

$$\hat{\beta}_0=\bar{y}-\hat{\beta}_1\bar{x} \tag{6.3}$$

where

$$S_{xx}=\sum_{i=1}^{n}(x_i-\bar{x})^2,$$
$$S_{xy}=\sum_{i=1}^{n}y_i(x_i-\bar{x}).$$

Of course, S_{xx} is called as *sum-of-squares* and S_{xy} as the *sum-of-cross-products*. Towards finding an estimate of the variance σ^2 , we proceed as follows. We first define *residuals* as the difference between the observed value y_i and the corresponding fittted value $\hat{y}_i$, that is, for *i = 1 , 2, ..., n:*

$$e_i=y_i-\hat{y}_i$$
$$=y_i-\beta_0-\beta_1 x_i$$

Define the residual or error sum of squares:

$$SS_{Res}=\sum_{i=1}^{n}e_i^2=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2. \tag{6.4}$$

Since the residuals are based on n observations, and the parameters β_0 and β_1 are estimated from it, the degrees of freedom associated with SS_{Res} is n – 2. An unbiased estimator of σ^2 is given by

$$\hat{\sigma}^2 = \frac{SS_{Res}}{n-2} = MS_{Res} \tag{6.5}$$

It is thus meaningful that an estimator of the variance is the *residual mean square*. Once the estimator of variance is avaiable, we can carry out tests for the parameters of the regression line. Mathematically, the expressions for the variance of $\hat{\beta}_0$ and $\hat{\beta}_1$ are respectively:

$$Var(\hat{\beta}_0) = \sigma^2 (\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}})$$

and

$$Var(\hat{\beta}_1) = \frac{\sigma^2}{S_{xx}}.$$

Using the above expressions and the estimator of the variance of the error term, we can estimate the standard error of $\hat{\beta}_0$ and $\hat{\beta}_1$ using:

$$se(\hat{\beta}_1) = \sqrt{\frac{MS_{Res}}{S_{xx}}} \tag{6.6}$$

and

$$se(\hat{\beta}_0) = \sqrt{(MS_{Res}(\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}}))} \tag{6.7}$$

Thus, if we are interested in testing $H_0 : \beta_1 = \beta_{10}$, a useful test statistic is

$$t_0 = \frac{\hat{\beta}_1 - \beta_{10}}{\sqrt{\frac{MS_{Res}}{S_{xx}}}} \tag{6.8}$$

which is distributed as a *t*-distribution with *n-2* degrees of freedom. Thus, a α - level test would be to reject the null hypothesis if

$$|t_0| > t_{\frac{\alpha}{2},n-2}$$

where $t_{\alpha/2,n-2}$ is the upper $\alpha/2$ percentile point of the t_{n-2} distribution.

Similarly, the test statistic for the hypothesis $H_0 : \beta_0 = \beta_{00}$ is

$$t_0 = \frac{\hat{\beta}_0 - \beta_{00}}{\sqrt{MS_{Res}(\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}})}} \tag{6.9}$$

which is again distributed as a *t*-distribution with *n-2* degrees of freedom, and the test procedure

parallels the testing of β_1.

Confidence Intervals. Using the null distributions of $\hat{\beta}_0$ and $\hat{\beta}_1$ the $100(1-\alpha)$ percent confidence intervals of the slope and intercept are given by

$$\hat{\beta}_1 - t_{\alpha/2,n-2}\, se(\hat{\beta}_1) \le \beta_1 \le \hat{\beta}_1 + t_{\alpha/2,n-2}\, se(\hat{\beta}_1) \tag{6.10}$$

and

$$\hat{\beta}_0 - t_{\alpha/2,n-2}\, se(\hat{\beta}_0) \le \beta_0 \le \hat{\beta}_0 + t_{\alpha/2,n-2}\, se(\hat{\beta}_0). \tag{6.11}$$

Finally, we state that a $100(1-\alpha)$ percent confidence interval for σ^2 is

$$\frac{(n-2)MS_{Res}}{\chi^2_{\alpha/2,n-2}} \le \sigma^2 \le \frac{(n-2)MS_{Res}}{\chi^2_{1-\alpha/2,n-2}}. \tag{6.12}$$

We will illustrate all the above concepts for the rocket propellant data.

Example. The Rocket Propellant Data. (contd.)

Note that its very simple to obtain S_{xx}, S_{xy} in R:

```
> attach(rp)
> sxx=sum((Age_of_Propellant-mean(Age_of_Propellant))^2)
> sxx
[1] 1106.559
> sxy=sum(Shear_Strength*(Age_of_Propellant-mean(Age_of_Propellant)))
> sxy
[1] -41112.65
> beta1=sxy/sxx
> beta1
[1] -37.15359
> beta0=mean(Shear_Strength)-beta1*mean(Age_of_Propellant)
> beta0
[1] 2627.822
> n=length(Shear_Strength)
> sst=sum(Shear_Strength^2)-n*(mean(Shear_Strength)^2)
> sst
[1] 1693738
> ssres=sst-beta1*sxy
> ssres
[1] 166254.9
> (sigma2=ssres/(n-2))
[1] 9236.381
> msres=ssres/(n-2)
> (sebeta1=sqrt(msres/sxx))
[1] 2.889107
```

```
> sebeta0=sqrt(msres*(1/n + (mean(Age_of_Propellant)^2)/sxx))
> sebeta0
[1] 44.18391
> (testbeta1=beta1/sebeta1)
[1] -12.85989
> (abs(qt(0.025,18)))  # returns the value of t-dist with 18 d.f at alpha=.05
[1] 2.100922
```

Since the absolute value of the test statistics for the slope term -12.85989 is larger than the value specified for the *t*-distribution 2.100922, we reject the null hypothesis and conclude that there is a linear relationship between the shear strength and the age of the propellant.

The computations related to the confidence intervals are as follows:

```
> (lclbeta1 = (beta1 - abs(qt(.025,18))*sebeta1)) #gives lower confidence limit for beta1
[1] -43.22338
> (uclbeta1 = (beta1 + abs(qt(.025,18))*sebeta1)) #gives upper confidence limit for beta1
[1] -31.08380
> (lclbeta0 = (beta0 - abs(qt(.025,18))*sebeta0)) #gives lower confidence limit for beta0
[1] 2534.995
> (uclbeta0 = (beta0 + abs(qt(.025,18))*sebeta0)) #gives upper confidence limit for beta0
[1] 2720.649
> (lclsigma2 = (n-2)*msres/qchisq(1-.025,18)) #gives lower confidence limit for sigma2
[1] 5273.516
> (uclsigma2 = (n-2)*msres/qchisq(1-.975,18)) #gives upper confidence limit for sigma2
[1] 20199.24
```

Yes, we know that the reader is asking to put the above calculations in more formal way. So here we report them. The 95% confidence intervals for the parameters are as follows:

$$-43.22338 \le \beta_1 \le -31.08380,$$
$$2534.995 \le \beta_0 \le 2720.649,$$

and

$$5273.516 \le \sigma^2 \le 20199.24\,.$$

The Analysis of Variance (ANOVA)

Gelman (2005) has done a comprehensive review on ANOVA and explains why it is still relevant and very useful tool after almost eighty years of its invention. ANOVA may be used to test the significance of the regression model.

The fundamental technique for ANOVA is partitioning of the total sum of squares into components used in the model. That is

$$SS_{Total} = SS_{Treatments} + SS_{Error}$$

Mathematically, the above is equivalent to

$$\sum_{i=1}^{n}(y_i-\bar{y})^2=\sum_{i=1}^{n}(\hat{y}_i-\bar{y})^2+\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 \tag{6.13}$$

where

$$SS_{Total}=SS_T=\sum_{i=1}^{n}(y_i-\bar{y})^2$$
$$SS_{Treatments}=SS_R=\sum_{i=1}^{n}(\hat{y}_i-\bar{y})^2$$
$$SS_{Error}=SS_{Res}=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

The *degrees of freedom* (df) are similarly partitioned as

$$df_{Total}=df_{Treatments}+df_{Error}$$

The explicit values of df is obtainted as

$$df_{Total}=df_{Treatments}+df_{Error}$$
$$n-1=1+(n-2)$$

which gives us $df_{Total}=n-1, df_{Treatments}=1, df_{Error}=n-2.$

Under the null hypothesis, SS_R has a χ^2_1 distribution. Since the distributions of SS_{Res} is a χ^2_{n-2} distribution, and SS_R is independent of SS_{Res}, we can use F-test for testing significance of the covariates. That is

$$F_0=\frac{SS_R/df_R}{SS_{Res}/df_{Res}}=\frac{SS_R/1}{SS_{Res}/(n-2)}=\frac{MS_R}{MS_{Res}} \tag{6.15}$$

is distributed as an *F*-distribution with 1 and n – 2 degrees of freedom. We can summarize the ANOVA procedure in an ANOVA table:

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0
Regression	$SS_R=\hat{\beta}_1 S_{xy}$	1	MS_R	MS_R/MS_{Res}
Residual	$SS_{Res}=SS_T-\hat{\beta}_1 S_x$	n -2	MS_{Res}	
Total	SS_T	n -1		

Example. The Rocket Propellant Data. (contd.)

The ANOVA table calculations for the problem are given below:

```
> (srs=beta1*sxy)
[1] 1527483
> ssres
[1] 166254.9
> sst
```

```
[1] 1693738
> (msrs=srs/1)
[1] 1527483
> msres
[1] 9236.381
> (f0=msrs/msres)
[1] 165.3768
```

The above values can be put in the table form:

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0
Regression	1527483	1	1527483	165.3768
Residual	166254.9	18	9236.381	
Total	1693738	19		

R^2: The Coefficient of Determination

An important measurement of the performance of the regression model is given by its coefficient of determination R^2 and it is defined by

$$R^2 = \frac{SS_R}{SS_T} = 1 - \frac{SS_{Res}}{SS_T}$$

The value of R^2 explains the proportion of variation in y explained by the regressor x.

Example. The Rocket Propellant Data. (contd.) The coefficient of determination value for this data set is 0.9018414. This is to say that approximately 90% of the variation in the Shear Strength of the bonding is explained by the Age of Propellant. Thus, it seems the linear model is a very good fit for the data.

Of course, we have put the reader through a lot of unnecessary R codes. Linear regression is after all one of the very important methods in statistical data analysis. The main idea of doing this exercise is to give a programming guide through R, and also parallelly understand the underlying theory with their computations. We now see how the "lm" command can be used for regression analysis.

Example. The Rocket Propellant Data. (contd.)

```
> rplm=lm(Shear_Strength ~ Age_of_Propellant)
```

```
> summary(rplm)
Call:
lm(formula = Shear_Strength ~ Age_of_Propellant)
Residuals:
    Min      1Q  Median      3Q     Max
-215.98  -50.68   28.74   66.61  106.76
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       2627.822     44.184   59.48  <2e-16***
Age_of_Propellant  -37.154      2.889  -12.86 1.64e-10***
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1
Residual standard error: 96.11 on 18 degrees of freedom
Multiple R-squared: 0.9018,     Adjusted R-squared: 0.8964
F-statistic: 165.4 on 1 and 18 DF,  p-value: 1.643e-10
> anova(rplm)
Analysis of Variance Table
Response: Shear_Strength
                  Df  Sum Sq Mean Sq F value  Pr(>F)
Age_of_Propellant  1 1527483 1527483  165.38 1.643e-10 ***
Residuals         18  166255    9236
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1
```

The reader may confirm the values here from those obtained earlier. In fact we may do a lot more in R than just obtain the summary and ANOVA of the regression model. We just give below a screenshot of the available contents of fitted linear model:

```
> rplm$
rplm$assign       rplm$coefficients  rplm$effects      rplm$model        rplm$rank
rplm$terms        rplm$call         rplm$df.residual
rplm$fitted.values rplm$qr          rplm$residuals    rplm$xlevels
```

R also provides a variety of plots for a linear model, such as, fitted-vs-residuals plot, normal q-q plot, Scale-location plot, and Residuals-vs-Leverage plots. The below commands produce these plots, and we reproduce below it those plots.

```
> tiff("rp.tiff")
> par(mfrow=c(2,2))
> plot(rplm)
> dev.off()
```

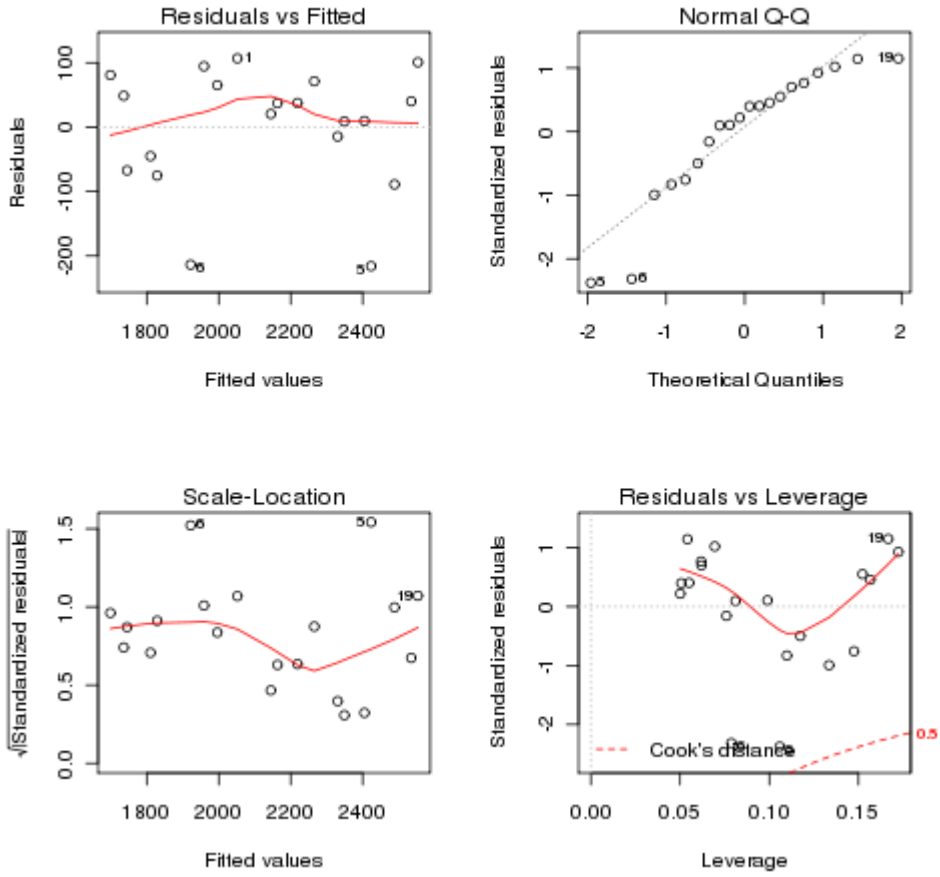


Figure 6.2. Residual Plots for the Regression Model of Shear Strength Data

A final note before we close this section on simple regression model. The least-squares estimates are very sensitive to outliers, and the entire study can go for a toss in its presence. We superficially change one value in the regressor and repeat the exercise of the fitting a linear model. Notice the drastic fall in the value of R-square. Of course, in the next section we highlight some more limitation using one of the very popular data sets in statistics.

```
> Age_of_Propellant[19]=20000000
> rplm1=lm(Shear_Strength~ Age_of_Propellant)
> summary(rplm1)
Call:
lm(formula = Shear_Strength ~ Age_of_Propellant)
Residuals:
    Min      1Q  Median      3Q     Max
-425.69 -320.36   58.11  217.35  471.16
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      2.104e+03  6.412e+01  32.812   <2e-16 ***
Age_of_Propellant 2.752e-05  1.434e-05   1.919    0.071 .
```

Signif. codes: 0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1

Residual standard error: 279.5 on 18 degrees of freedom
Multiple R-squared: 0.1699, Adjusted R-squared: 0.1238
F-statistic: 3.684 on 1 and 18 DF, p-value: 0.07094

6.3 The Anscombe Warnings

Anscombe (1974) presented four data sets which have same values in the mean, variance, correlation, regression line, R-square value, *p*-values, etc. This data set is available in R as "anscombe", and we reproduce it below for the benefit of the reader.

x1	x2	x3	x4	y1	y2	y3	y4
10	10	10	8	8.04	9.14	7.46	6.58
8	8	8	8	6.95	8.14	6.77	5.76
13	13	13	8	7.58	8.74	12.74	7.71
9	9	9	8	8.81	8.77	7.11	8.84
11	11	11	8	8.33	9.26	7.81	8.47
14	14	14	8	9.96	8.1	8.84	7.04
6	6	6	8	7.24	6.13	6.08	5.25
4	4	4	19	4.26	3.1	5.39	12.5
12	12	12	8	10.84	9.13	8.15	5.56
7	7	7	8	4.82	7.26	6.42	7.91
5	5	5	8	5.68	4.74	5.73	6.89

Table 6.1. Anscombes Quartet Data

We reproduce here select summaries:

```
> summary(anscombe)
       x1          x2          x3          x4          y1          y2          y3          y4
 Median : 9.0   9.0   9.0   8   7.580   8.140   7.11   7.040
 Mean   : 9.0   9.0   9.0   9   7.501   7.501   7.50   7.501
```

Also, the ANOVA table for the four data-sets are very identical:

```
Response: y1
          Df  Sum Sq Mean Sq F value   Pr(>F)
x1         1 27.5100 27.5100   17.99 0.002170 **
Residuals  9 13.7627  1.5292
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1
Response: y2
          Df  Sum Sq Mean Sq F value   Pr(>F)
x2         1 27.5000 27.5000  17.966 0.002179 **
Residuals  9 13.7763  1.5307
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1
Response: y3
          Df  Sum Sq Mean Sq F value   Pr(>F)
```

```
x3          1 27.4700 27.4700  17.972 0.002176 **
Residuals  9 13.7562  1.5285
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1
Response: y4
          Df  Sum Sq Mean Sq F value   Pr(>F)
x4          1 27.4900 27.4900  18.003 0.002165 **
Residuals  9 13.7425  1.5269
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1
```

Next, we reproduce the summaries of the linear models.

```
> summary(lm.1)
Call:
lm(formula = ff, data = anscombe)
Residuals:
    Min      1Q  Median      3Q     Max
-1.92127 -0.45577 -0.04136  0.70941  1.83882
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0001     1.1247   2.667  0.02573 *
x1            0.5001     0.1179   4.241  0.00217 **
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1
Residual standard error: 1.237 on 9 degrees of freedom
Multiple R-squared: 0.6665,     Adjusted R-squared: 0.6295
F-statistic: 17.99 on 1 and 9 DF,  p-value: 0.002170
> summary(lm.2)
Call:
lm(formula = ff, data = anscombe)
Residuals:
   Min     1Q  Median     3Q    Max
-1.9009 -0.7609  0.1291  0.9491  1.2691
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.001      1.125   2.667  0.02576 *
x2             0.500      0.118   4.239  0.00218 **
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1

Residual standard error: 1.237 on 9 degrees of freedom
Multiple R-squared: 0.6662,     Adjusted R-squared: 0.6292
F-statistic: 17.97 on 1 and 9 DF,  p-value: 0.002179
> summary(lm.3)
Call:
lm(formula = ff, data = anscombe)
Residuals:
   Min     1Q  Median     3Q    Max
-1.1586 -0.6146 -0.2303  0.1540  3.2411
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)   3.0025     1.1245   2.670  0.02562 *
x3            0.4997     0.1179   4.239  0.00218 **
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1
Residual standard error: 1.236 on 9 degrees of freedom
Multiple R-squared: 0.6663,     Adjusted R-squared: 0.6292
F-statistic: 17.97 on 1 and 9 DF,  p-value: 0.002176
> summary(lm.4)
Call:
lm(formula = ff, data = anscombe)
Residuals:
      Min        1Q    Median        3Q       Max
-1.751e+00 -8.310e-01  1.258e-16  8.090e-01  1.839e+00
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0017     1.1239   2.671  0.02559 *
x4            0.4999     0.1178   4.243  0.00216 **
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1
Residual standard error: 1.236 on 9 degrees of freedom
Multiple R-squared: 0.6667,     Adjusted R-squared: 0.6297
F-statistic:    18 on 1 and 9 DF,  p-value: 0.002165
```

Despite the striking similarities, there is a drastic difference in the four data sets which is reflected in the scatter plots produced below.

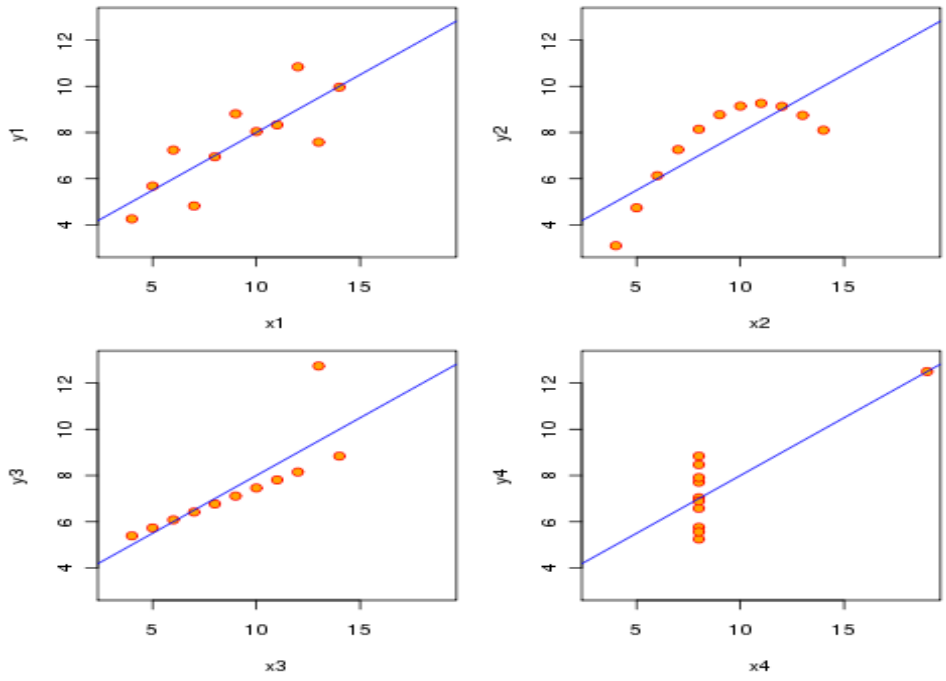


Figure 6.3. Scatter Plots of Anscombe's Quartet

The above plot can be obtained using "example(anscombe)" at the R console.

We now move to the more general regression model in which we have more than one regressor.

6.4 Linear Regression Models: Multiple Regression

Suppose that $\boldsymbol{x}_i=(x_{i1}, x_{i2}, \ldots, x_{ip})$ is a vector of independent observations which influences the outcome of a variable of interest $y_i, i=1,2,\ldots,n$. A popular example of a regression study is understanding the effect of fathers height, say x, on that of the son, denoted y. If there is just one explainatory variable, we call it as *simple regression problem,* and if there are more than one explainatory variable, we refer that as *multiple regression problem.* For example, the researcher may be interested in modeling the effect of heights of father, x_1, and mother, x_2, on the height of son.

Example. US Crime Data

Data is available on the crime rates across 47 states in USA, and we have additional information on 13 more explainatory variables. This data set has been used and illustrated in the Der and Everitt (2002) book on the use of the SAS software. The data is given in the file US_Crime.csv. We explain the variables, as detailed in Der and Everitt, as below.

R: Crime rate: the number of offences known to the police per 1,000,000 population
Age: Age distribution: the number of males aged 14 to 24 years per 1000 of total state population
S: Binary variable distinguishing southern states (S = 1) from the rest
Ed: Educational level: mean number of years of schooling × 10 of the population 25 years old and over
Ex0: Police expenditure: per capita expenditure on police protection by state and local governments in 1960
Ex1: Police expenditure: as Ex0, but for 1959 LF Labour force participation rate per 1000 civilian urban males in the age group 14 to 24 years
M: Number of males per 1000 females
N: State population size in hundred thousands
NW: Number of non-whites per 1000
U1: Unemployment rate of urban males per 1000 in the age group 14 to 24 years
U2: Unemployment rate of urban males per 1000 in the age group 35 to 39 years
W: Wealth, as measured by the median value of transferable goods and assets or family income (unit 10 dollars)
X: Income inequality: the number of families per 1000 earning below one half of the median income

We first read the data by the command:

```
> usc=read.csv("/mypath/uscrime.csv",header=T)
```
and check that the data has been read properly:
```
> dim(usc)
[1] 47 14
> names(usc)
[1] "R"   "Age" "S"   "Ed"  "Ex0" "Ex1" "LF"  "M"   "N"   "NW"  "U1"  "U2"
[13] "W"   "X"
```

6.4.1 Scatter Plots: A First Look

In the case of a simple regression problem, the user may first study the nature of relationship using the scatter plot. For the multiple regression data, we use the R graphics method called "pairs". For the US Crime data, we obtain the plot as follows.

```
> tiff("US_Crime.tiff")
> pairs(usc)
> dev.off()
```
After having stored the output to a tiff file, we paste below the graph:

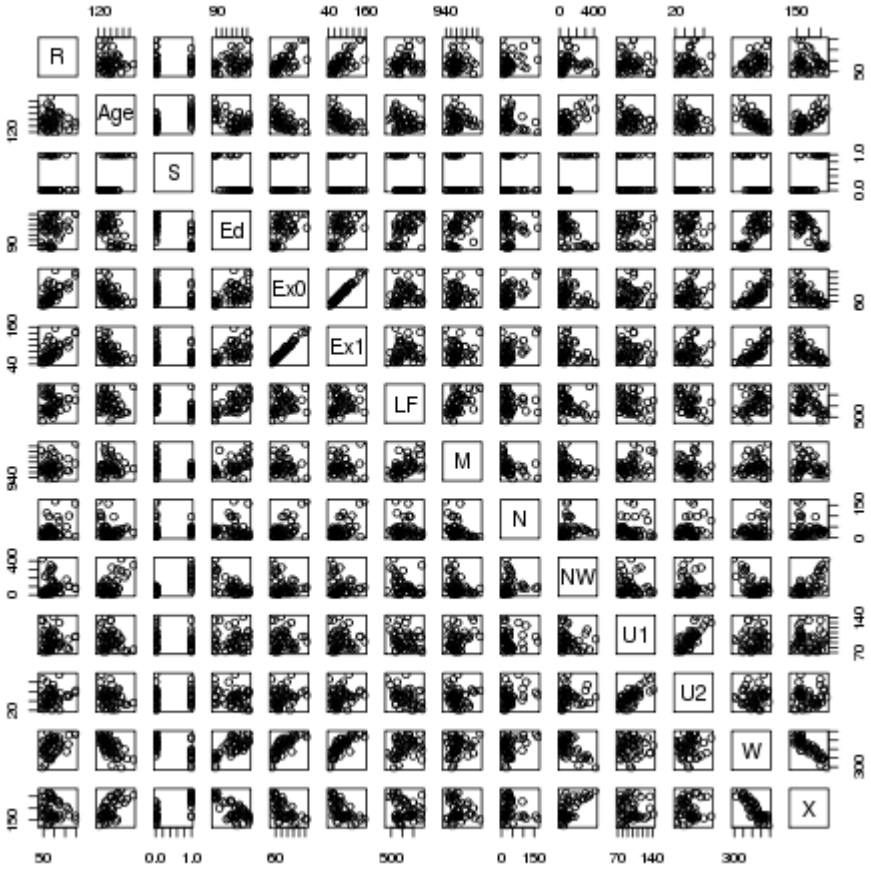


Figure 6.4. Scatter Plot of US Crime Data

From the first row of the above scatter plot we see that the crime rate is weekly related to each of the explanatory variables. The scatter plot also reveals strong relationship between the variables Ex0 and Ex1, indicating *multicollinearity*, and a negative relationship between the variables W and X, that is *high correlationship*.

6.4.2 Fitting a Multiple Regression

Given n observations, the model (6.1) can be written in a matrix form as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{6.16}$$

where $\boldsymbol{y}' = [y_1, y_2, \ldots, y_n]$, $\boldsymbol{\beta}' = [\beta_1, \beta_2, \ldots, \beta_k]$, $\boldsymbol{\epsilon}' = [\epsilon_1, \epsilon_2, \ldots, \epsilon_n]$, and

$$\boldsymbol{X} = \begin{matrix} 1 & x_{11} & x_{12} & \ldots & x_{1k} \\ 1 & x_{21} & x_{22} & \ldots & x_{2k} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{nk} \end{matrix}$$

The least-squares normal equations for the model (6.16) is given by

$$\boldsymbol{X}'\boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{y} \tag{6.17}$$

which leads to the least-squares estimator

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y} \tag{6.18}$$

Example. US Crime Data. (contd..)

```
> model1=lm(R~Age+S+Ed+Ex0+Ex1+LF+M+N+NW+U1+U2+W+X,usc)
> summary(model1)

Call:
lm(formula = R ~ Age + S + Ed + Ex0 + Ex1 + LF + M + N + NW +
    U1 + U2 + W + X, data = usc)
Residuals:
   Min     1Q Median     3Q    Max
-34.884 -11.923  -1.135  13.495  50.560
Coefficients:
             Estimate   Std. Error  t value  Pr(>|t|)
(Intercept) -6.918e+02  1.559e+02  -4.438   9.56e-05 ***
Age          1.040e+00  4.227e-01   2.460   0.01931 *
S           -8.308e+00  1.491e+01  -0.557   0.58117
Ed           1.802e+00  6.496e-01   2.773   0.00906 **
Ex0          1.608e+00  1.059e+00   1.519   0.13836
Ex1         -6.673e-01  1.149e+00  -0.581   0.56529
LF          -4.103e-02  1.535e-01  -0.267   0.79087
M            1.648e-01  2.099e-01   0.785   0.43806
N           -4.128e-02  1.295e-01  -0.319   0.75196
```

```
NW          7.175e-03   6.387e-02   0.112     0.91124
U1          -6.017e-01  4.372e-01   -1.376    0.17798
U2          1.792e+00   8.561e-01   2.093     0.04407 *
W           1.374e-01   1.058e-01   1.298     0.20332
X           7.929e-01   2.351e-01   3.373     0.00191 **
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1

Residual standard error: 21.94 on 33 degrees of freedom
Multiple R-squared: 0.7692,     Adjusted R-squared: 0.6783
F-statistic: 8.462 on 13 and 33 DF,  p-value: 3.686e-07
```

6.4.3 Variance Inflation Factor

We revisit the role of scatter plots in the context of multiple regression model before embarking on the problem of variance inflation factor (VIF) in the present section and model selection in the next one. Scatter plot is essentially a grid of scatterplots for each pair of variables. Such a display is often useful in assessing the general relationships between the variables, identifying possible outliers, and highlighting potential multicollinearity problems amongst the explanatory variables (i.e., one explanatory variable being essentially predictable from the remainder). Multicollinearity is best understood by spliting its spelling as "multi-col-linearity" implying linear relationship among the multiple columns of the design matrix. Highly correlated explanatory variables, multicollinearity, can cause several problems when applying the multiple regression model. For instance, in the presence of multicollinearity, the design matrix may not be of full rank.

Recollect from Figure 6.4 that the individual relationships of crime rate with each of the explanatory variables shown in the first column of the plot do not appear to be particularly strong, apart perhaps from Ex0 and Ex1. The scatterplot matrix also clearly highlights the very strong relationship between these two variables. Highly correlated explanatory variables, multicollinearity, can cause several problems when applying the multiple regression model.

Spotting multicollinearity amongst a set of explanatory variables might not be easy. The obvious course of action is to simply examine the correlations between these variables, but whilst this is often helpful, it is by no means foolproof, more subtle forms of multicollinearity may be missed. An alternative and generally far more useful approach is to examine what are known as the variance inflation factors of the explanatory variables.

The *variance inflation factor* VIF_j for the j-th variable is given by

$$VIF_j = \frac{1}{(1-R_j^2)}$$

where R_j^2 is the square of the multiple correlation coefficient from the regression of the j-th explanatory variable on the remaining explanatory variables. The variance inflation factor of an explanatory variable indicates the strength of the linear relationship between the variable and the remaining explanatory variables. A rough rule of thumb is that variance inflation factors greater than 10 gives some cause for concern.

Example. US Crime Data. (contd..)

In the previous sub-section, we have fit a full model for the US crime data. We now compute Variance Inflation Factors(VIF) of the 13 explanatory variables using the below codes. We require the R function "vif" for this purpose which is in the library "faraway", hence we inculde it in the present R session by the following code.

```
> library(faraway)
> uscrimewor=uscrime[,-1]
> vif(uscrimewor)
     Age        S       Ed      Ex0      Ex1       LF        M        N
2.698021 4.876751 5.049442 94.633118 98.637233 3.677557 3.658444 2.324326
      NW       U1       U2        W        X
4.123274 5.938264 4.997617 9.968958 8.409449
```

Concentrating for now on the variance inflation factors in the above output,we see that those for Ex0 and Ex1 are well above the value 10. As a consequence, we simply drop variable Ex0 from consideration and now regress crime rate on the remaining 12 explanatory variables using the following code:

```
> model2=lm(R~Age+S+Ed-Ex0+Ex1+LF+M+N+NW+U1+U2+W+X,uscrime) # Note how
                                        the variable Ex0 is removed from the model1#
> summary(model2)
Call:
lm(formula = R ~ Age + S + Ed - Ex0 + Ex1 + LF + M + N + NW +
    U1 + U2 + W + X, data = uscrime)
Residuals:
    Min      1Q  Median      3Q     Max
-43.372 -15.258   2.249  12.903  45.055
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -739.89065  155.54826  -4.757 3.54e-05 ***
Age            1.08541    0.42967   2.526 0.016355 *
```

```
S           -8.16412   15.19508  -0.537 0.594569
Ed           1.62670    0.65153   2.497 0.017542 *
Ex1          1.02965    0.27202   3.785 0.000597 ***
LF           0.00509    0.15331   0.033 0.973707
M            0.18686    0.21341   0.876 0.387413
N           -0.01639    0.13092  -0.125 0.901122
NW           -0.00213    0.06478  -0.033 0.973964
U1          -0.61879    0.44533  -1.390 0.173709
U2           1.94296    0.86652   2.242 0.031577 *
W            0.14739    0.10763   1.369 0.179866
X            0.81550    0.23908   3.411 0.001685 **
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1

Residual standard error: 22.35 on 34 degrees of freedom
Multiple R-squared: 0.7531,     Adjusted R-squared: 0.666
F-statistic: 8.643 on 12 and 34 DF,  p-value: 3.295e-07
```

The square of the multiple correlation coefficient is 0.75, indicating that the 12 explanatory variables account for 75% of the variability in the crime rates of the 47 states.

Now we shall compute VIF of the 12 explanatory variables excluding Ex0

```
> vif(uscrimewor[,-4])
    Age       S      Ed     Ex1      LF       M       N      NW
2.684410 4.876553 4.890755 5.325942 3.533571 3.640926 2.287111 4.085332
     U1      U2       W       X
5.934319 4.930479 9.930128 8.375840
```

The variance inflation factors are now all less than 10.The main features of interest in the output are the F-test and the parameter estimates.

In the former, the F-test is for the hypothesis that all the regression coefficients in the regression equation are zero. Here, the evidence against this hypothesis is very strong (the relevant P-value < 0.0001). In general, however, this overall test is of little real interest because it is most unlikely in general that none of the explanatory variables will be related to the response. The more relevant question is whether a subset of the regression coefficients is zero, implying that not all the explanatory variables are informative in determining the response. It might be thought that the nonessential variables can be identified by simply examining the estimated regression coefficients and their standard errors as given in the above summary output, with those regression coefficients significantly different from zero identifying the explanatory variables needed in the derived regression equation, and those not different from zero corresponding to variables that can be

omitted. Unfortunately,this very straightforward approach is not in general suitable, simply because the explanatory variables are correlated in most cases. Consequently, removing a particular explanatory variable from the regression will alter the estimated regression coefficients (and their standard errors) of the remaining variables. The parameter estimates and their standard errors are conditional on the other variables in the model. A more involved procedure is thus necessary for identifying subsets of the explanatory variables most associated with crime rate. In Regression Analysis it is popularly called as **Variable Selection or Model Selection,** which is taken up in the next section.

6.4.4 Model Selection

A number of methods are available, including: (i) Backward elimination, (ii) Forward selection, and (iii) Stepwise regression.

Backward elimination. This is the simplest of all variable selection procedures and can be easily implemented without special software. In situations where there is a complex hierarchy, backward elimination can be run manually while taking account of what variables are eligible for removal. This method starts with a model containing all the explanatory variables and eliminates variables one by one, at each stage choosing the variable for exclusion as the one leading to the smallest decrease in the regression sum of squares. Once again, an F-type statistic is used to judge when further exclusions would represent a significant deterioration in the model.

1. Start with all the predictors in the model
2. Remove the predictor with highest p-value greater than alpha critical.
3. Refit the model and goto 2
4. Stop when all p-values are less than alpha critical.

The alpha critical is sometimes called the p-to-remove and does not have to be 5%. If prediction performance is the goal, then a 15-20% cut-off may work best, although methods designed more directly for optimal prediction should be preferred.

Forward selection. This just reverses the backward method.This method starts with a model containing none of the explanatory variables and then considers variables one by one for inclusion. At each step, the variable added is one that results in the biggest increase in the regression sum of

squares. An F-type statistic is used to judge when further additions would not represent a significant improvement in the model.

1. Start with no variables in the model.
2. For all predictors not in the model, check their p-value if they are added to the model. Choose the one with lowest p-value less than alpha critical.
3. Continue until no new predictors can be added.

Stepwise regression. This method is, essentially, a combination of forward selection and backward elimination. This addresses the situation where variables are added or removed early in the process and we want to change our mind about them later. Starting with no variables in the model, variables are added as with the forward selection method. Here, however, with each addition of a variable, a backward elimination process is considered to assess whether variables entered earlier might now be removed because they no longer contribute significantly to the model.
We discuss below computations related to backward elimination method only.

Example. US Crime Data Continued.
We illustrate the backward method - at each stage we remove the predictor with the largest p-value over 0.05. Sine the p-value of regressor NM in the above summary output of model2 is large, we shall eliminate it first.

```
> model3=update(model2,.~.-NW)
> summary(model3)
Call:
lm(formula = R ~ Age + S + Ed + Ex1 + LF + M + N + U1 + U2 +
    W + X, data = uscrime)
Residuals:
   Min     1Q Median     3Q    Max
-43.27 -15.14   2.09  13.01  44.94
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -7.405e+02  1.524e+02  -4.859 2.45e-05 ***
Age          1.081e+00  4.005e-01   2.699 0.010639 *
S           -8.404e+00  1.315e+01  -0.639 0.526798
Ed           1.630e+00  6.344e-01   2.569 0.014609 *
Ex1          1.026e+00  2.397e-01   4.279 0.000138 ***
LF           3.502e-03  1.434e-01   0.024 0.980655
M            1.887e-01  2.031e-01   0.929 0.359349
N           -1.637e-02  1.290e-01  -0.127 0.899773
```

```
U1          -6.214e-01  4.321e-01  -1.438 0.159319
U2           1.944e+00  8.539e-01   2.276 0.029073 *
W            1.482e-01  1.032e-01   1.435 0.160056
X            8.152e-01  2.354e-01   3.462 0.001430 **
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1

Residual standard error: 22.03 on 35 degrees of freedom
Multiple R-squared: 0.7531,     Adjusted R-squared: 0.6755
F-statistic: 9.705 on 11 and 35 DF,  p-value: 1.015e-07
```

Since the p-value of LF is very high, we shall eliminate it now.

```
> model4=update(model3,.~.-LF)
> summary(model4)
Call:
lm(formula = R ~ Age + S + Ed + Ex1 + M + N + U1 + U2 + W + X,
    data = uscrime)
Residuals:
    Min      1Q  Median      3Q     Max
-43.291 -14.997   2.118  12.982  44.926

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -741.2494   146.7795  -5.050 1.29e-05 ***
Age            1.0804     0.3945   2.739 0.009526 **
S             -8.5427    11.6790  -0.731 0.469231
Ed             1.6342     0.6027   2.711 0.010203 *
Ex1            1.0243     0.2304   4.445 8.07e-05 ***
M              0.1912     0.1721   1.111 0.273988
N             -0.0158     0.1252  -0.126 0.900222
U1            -0.6258     0.3865  -1.619 0.114168
U2             1.9450     0.8399   2.316 0.026384 *
W              0.1484     0.1014   1.464 0.151820
X              0.8166     0.2250   3.629 0.000877 ***
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1

Residual standard error: 21.72 on 36 degrees of freedom
Multiple R-squared: 0.7531,     Adjusted R-squared: 0.6845
F-statistic: 10.98 on 10 and 36 DF,  p-value: 2.935e-08
```

Since p-value of the regressor N is large, we shall eliminate it, and redo the above step.

```
>  model5=update(model4,.~.-N)
>  summary(model5)
Call:
lm(formula = R ~ Age + S + Ed + Ex1 + M + U1 + U2 + W + X, data = uscrime)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-42.779 -14.823   2.647  12.471  44.630
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -747.10655  137.39262  -5.438 3.62e-06 ***
Age            1.08103    0.38914   2.778 0.008540 **
S             -8.26275   11.31302  -0.730 0.469760
Ed             1.63501    0.59459   2.750 0.009170 **
Ex1            1.01223    0.20672   4.897 1.94e-05 ***
M              0.20110    0.15123   1.330 0.191743
U1            -0.62956    0.38022  -1.656 0.106224
U2             1.94184    0.82830   2.344 0.024536 *
W              0.14596    0.09813   1.487 0.145381
X              0.80639    0.20723   3.891 0.000401 ***
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1

Residual standard error: 21.43 on 37 degrees of freedom
Multiple R-squared: 0.753,      Adjusted R-squared: 0.6929
F-statistic: 12.53 on 9 and 37 DF,  p-value: 7.984e-09
```

Since p-value of the regressor S is large, we shall eliminate it.

```
> model6=update(model5,.~.-S)
> summary(model6)
Call:
lm(formula = R ~ Age + Ed + Ex1 + M + U1 + U2 + W + X, data = uscrime)
Residuals:
    Min      1Q  Median      3Q     Max
-42.054 -14.879   1.649  12.997  46.943
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -744.54750  136.50217  -5.454 3.19e-06 ***
Age            1.00497    0.37264   2.697 0.010376 *
Ed             1.68655    0.58675   2.874 0.006596 **
Ex1            1.00069    0.20484   4.885 1.90e-05 ***
M              0.21061    0.14975   1.406 0.167725
U1            -0.57438    0.37034  -1.551 0.129203
U2             1.81849    0.80590   2.256 0.029876 *
W              0.14563    0.09752   1.493 0.143617
X              0.75692    0.19464   3.889 0.000392 ***
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1

Residual standard error: 21.3 on 38 degrees of freedom
Multiple R-squared: 0.7494,     Adjusted R-squared: 0.6967
F-statistic: 14.21 on 8 and 38 DF,  p-value: 2.591e-09
```

Since the p-value of the regressor M is greater , we will eliminate it.

```
> model7=update(model6,.~.-M)
> summary(model7)
Call:
lm(formula = R ~ Age + Ed + Ex1 + U1 + U2 + W + X, data = uscrime)
Residuals:
    Min      1Q  Median      3Q     Max
-48.2637 -13.5527   0.1152  11.9836  49.1757
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -631.70033  111.80637  -5.650 1.59e-06 ***
Age           1.14639    0.36329   3.156 0.003083 **
Ed            2.01335    0.54549   3.691 0.000682 ***
Ex1           1.00043    0.20740   4.824 2.18e-05 ***
U1           -0.32881    0.33067  -0.994 0.326168
U2            1.52886    0.78886   1.938 0.059878 .
W             0.16887    0.09731   1.735 0.090583 .
X             0.83049    0.18982   4.375 8.78e-05 ***
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1

Residual standard error: 21.57 on 39 degrees of freedom
Multiple R-squared: 0.7364,     Adjusted R-squared: 0.6891
F-statistic: 15.56 on 7 and 39 DF,  p-value: 1.556e-09
```

Since the regressor U1 has very high p-value, we will eliminate it.

```
> model8=update(model7,.~.-U1)
> summary(model8)
Call:
lm(formula = R ~ Age + Ed + Ex1 + U2 + W + X, data = uscrime)
Residuals:
    Min      1Q  Median      3Q     Max
-43.5053 -12.7711  -0.4111  11.0408  51.7528
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -636.60125  111.68203  -5.700 1.25e-06 ***
Age           1.14006    0.36318   3.139 0.003180 **
Ed            1.78983    0.49695   3.602 0.000864 ***
Ex1           1.07467    0.19346   5.555 1.99e-06 ***
U2            0.87907    0.44184   1.990 0.053508 .
W             0.18420    0.09607   1.917 0.062344 .
X             0.86175    0.18717   4.604 4.14e-05 ***
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1
```

```
Residual standard error: 21.56 on 40 degrees of freedom
Multiple R-squared: 0.7297,     Adjusted R-squared: 0.6892
F-statistic:    18 on 6 and 40 DF,  p-value: 5.523e-10
```

Since W has very high p-value, we will eliminate it.

```
> model9=update(model8,.~.-W)
> summary(model9)
Call:
lm(formula = R ~ Age + Ed + Ex1 + U2 + X, data = uscrime)
Residuals:
    Min      1Q  Median      3Q     Max
-51.7326 -11.7795  -0.8152  10.9109  61.1339
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -528.8557    99.6162  -5.309 4.13e-06 ***
Age            1.0184     0.3691   2.759 0.008619 **
Ed             2.0363     0.4955   4.110 0.000185 ***
Ex1            1.2973     0.1597   8.124 4.49e-10 ***
U2             0.9901     0.4521   2.190 0.034263 *
X              0.6463     0.1545   4.183 0.000148 ***
---
Signif. codes:  0 â***â 0.001 â**â 0.01 â*â 0.05 â.â 0.1 â â 1

Residual standard error: 22.26 on 41 degrees of freedom
Multiple R-squared: 0.7049,     Adjusted R-squared: 0.6689
F-statistic: 19.58 on 5 and 41 DF,  p-value: 6.354e-10
```

Since none of the p-values are greater than 0.05, the final removal of the W variable is a close call. We now stop backward selection process. Notice that the R^2 for the full model of 0.7531 is reduced only slightly to 0.7049 in the final model. Thus, the removal of eight predictors causes only a minor reduction in fit.

* * * * *

BIBLIOGRAPHY

Albert, J. (2007). *Bayesian Computations with R.* Springer

Albert, J. (2009). *Bayesian Computations with R,* 2nd ed. Springer

Anscombe, F. J. (1973) Graphs in statistical analysis. *American Statistician*, **27**, 17-21.

Berger, J.O. (1985). *Statistical Decision Theory*, 2e. Springer.

Bernardo, J. and Smith, A. (1994). *Bayesian Theory*. John Wiley, New York.

Bolstad, W. M. (2007). *Introduction to Bayesian Statistics*, 2e. Wiley.

Carlin, B. and Louis, T. (1996). *Bayes and Empirical Bayes Methods for Data Analysis*. Chapman and Hall, New York.

Crawley, M. (2007). *The R Book*. Wiley.

Dalgaard, P. (2002), *Introductory Statistics with R* , New York, USA: Springer- Verlag.

Dalgaard, P. (2008), *Introductory Statistics with R* , 2e, New York, USA: Springer- Verlag.

Dasu, J., and Johnson, T. (2003). *Exploratory Data Mining and Data Cleaning*. J. Wiley

Der, G., and Everitt, B.S. (2002). *A Handbook of Statistical Analysis using SAS*, 2e. CRC.

Everitt, B.S., and Hothorn, T. (2006). *A Handbook of Statistical Analyses Using R*. CRC

Faraway, J. (2002). *Practical Regression and Anova using R*. Freely distributed notes.

Faraway, J. (2006). *Extending the Linear Model with R*. CRC.

Fox, J. (2002). *An R and S-Plus Companion to Applied Regression*. Sage.

Freedman, D., Pisani, R., and Purves, R. (1997). *Statistics*, 3e. Norton.

Freund, R.J., and Wilson, W.J. (2003). *Statistical Methods*, 2e. Academic Press.

Gelman, A., Carlin, J., Stern, H., and Rubin, D. (2001). *Bayesian Data Analysis*. Chapman and Hall, New York, second edition.

Gentle, J.E. (2009). *Computational Statistics*. Springer-Verlag.

Gentle, J.E. (2003). *Random Number Generation and Monte Carlo Methods*, 2e. Springer-Verlag.

Gentle, J.E. (2007). *Matrix Algebra, Theory, Computations, and Applications in Statistics*. Springer.

Gelman, A., Carlin, J., Stern, H., and Rubin, D. (2001). *Bayesian Data Analysis*. Chapman and Hall, New York, second edition.

Ghosh, J.K., Delampady, M., and Samanta, T. (2006). *Introduction to Bayesian Analysis*. Springer.

Gibbons, J.D., and Chakraborti, S. (2003). *Nonparametric Statistical Inference*, 4e. Marcel Dekker.

Govindrajalu, Z. (2007). *Nonparametric Inference*, World Scientific.

Hajek, J., Sidak, Z., and Sen, P. K. (1967). *Theory of Rank Tests*, 2e. Academic Press, New York.

Hajek, J., and Sidak, Z. (1967). *Theory of Rank Tests*. Academic Press, New York.

Hogg, R.V., and Craig, A.T. (1978). *Introduction to Mathematical Statistics*, 4e. Macmillan.

Horgan, J. (2008). *Introduction to Probability With R*. J. Wiley.

Liu, J.S. (2002). *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag.

Mahalanobis, P.C. (1999). Why Statistics? *Resonance*.

Maindonald, J. and Braun, J. (2003). *Data Analysis and Graphics Using R*. Cambridge, UK: Cambridge University Press.

Marian, J-M., and Robert, C.P. (2007). *Bayesian Core: A Practical Approach to Computational Bayesian Statistics*. Springer.

Montgomery, D.C., Peck, E.A., and Vining, G.G. (2003). *Introduction to Linear Regression Analysis*. Wiley.

Mosteller, F. and Tukey, J. W. (1977). *Data Analysis and Regression*. Reading, MA:Addison-Wesley.

Murrell, P. (2005). *R Graphics* , Boca Raton, Florida, USA: Chapman & Hall/CRC.

Nolan, D., and Speed, T. (2000). *Stat Labs – Mathematical Statistics Through Applications*. Springer.

Purohit, S.G., Gore, S.D., and Deshmukh, S.R. (2008). *Statistics Using R*. Narosa, India.

R Development Core Team (2009a), *An Introduction to R*, R Foundation for Statistical Computing, Vienna, Austria, URL http://www.R-project.org, ISBN 3-900051-12-7.

Ritz, C., and Streibig, J.C. (2008). *Nonlinear Regression with R*. Springer-Verlag

Robert, C. (2001). *The Bayesian Choice*. Springer-Verlag, New York, second edition.

Ross, S.M. (2006). *Simulation*, 4e. Elsevier.

Sarkar, D. (2008). *Lattice: Multivariate Data Visualization with R*, New York, USA: Springer-Verlag.

Seber, G.A.F. (2008). *A Matrix Handbook for Statisticians*. Wiley.

Sheskin, D.J. (2004). *Handbook of Parametric And Nonparametric Statistical Procedures*, 3e. CRC.

Sheather, S.J. (2009). *A Modern Approach to Regression with R*. Springer-Verlag

Sivia, D.S. With J. Skilling. (2006). *Data Analysis – A Bayesian Tutorial*. Oxford.

Snedecor, G.W., and Cochran,W. G. (1980). *Statistical Methods*, 7th ed. Ames, IA: Iowa State Univ. Press.

Spector, P. (2008). *Data Manipulation With R*. Springer.

Tukey, J.W. (1962). The Future of Data Analysis. *Ann. Statist*. 1-67.

Tukey, J. W. (1977). *Exploratory Data Analysis*. Reading, MA: Addison-Wesley.

Venables and Ripley (2006)

Velleman, P.F., and Hoaglin, D.C. (2004). *ABC of Exploratory Data Analysis*. Duxbury Press, Boston. Republished in 2004 by The Internet-First University Press.

Wasserman, R. (2006). *All of Nonparametric Statistics*. Springer-Verlag.

* * * * *