R Workshop

Prabhanjan N. T
For
Department of Statistics
Bangalore University

6 March, 2010

R Basics

- Simple Arithmetic and Important Functions
- Data Handling and Manipulation
- Descriptive Statistics
- Exploratory Data Analysis
- Probability
- Statistical Inference

Simple Arithmetic

```
> 57 + 89
[1] 146
> 45 - 87
[1] -42
> 60 * 3
[1] 180
> 7CXPS /18
[1] 0.3888889
> 4^4
[1] 256
> 5%%2
[1] 1
> 5%/%2
[1] 2
```

```
> 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
```

Entering Data

```
> 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"
```

R makes a distinction between a row and a column vector:

```
> 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
```

Entering Data

Use “mode” to know wether an R object is numeric, character, etc.

```
> 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"
```

Entering a Vector Manually:

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

Entering Data

Entering a Matrix Manually:

```
> 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
```

More about vectors and matrix operations after 13 slides ...

Logical Operators

Symbol	Meaning
!	logical NOT
&	logical AND
|	logical OR
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	logical equals (double =)
!=	not equal
&&	AND with IF
||	OR with IF
xor(x,y)	exclusive OR
isTRUE(x)	an abbreviation of identical(TRUE,x)

Reading Data From External Files

- Data is rarely entered in a specialized software
- Generally stored in some external files like DAT, XLS, CSV, etc.

```
> (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
```

Reading from ".csv" files

```
> megadata <- read.csv("/bu/dos/Multivariate_Statistics/anderson.csv",header=T)
```

Reading Data From External Files

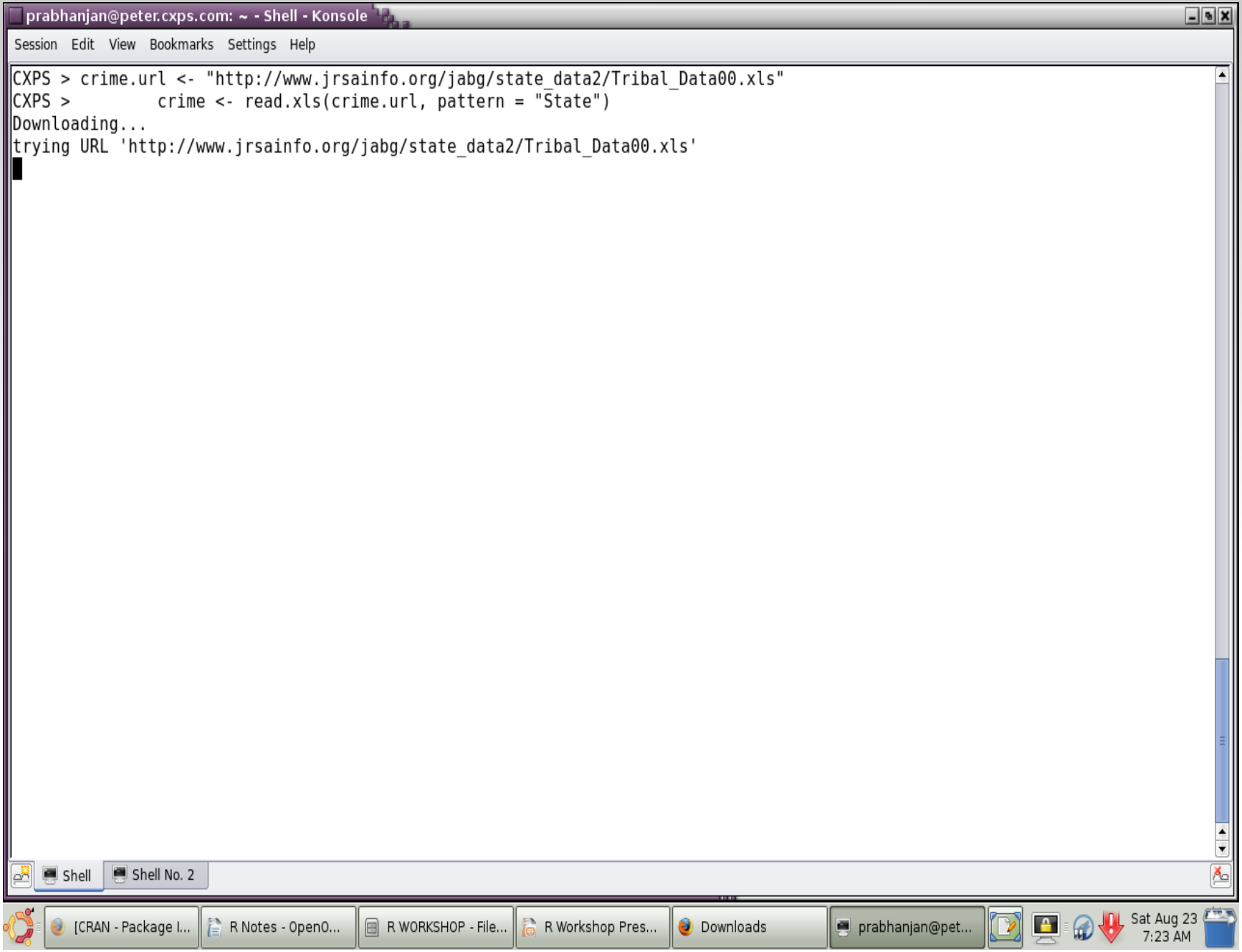
To read from Excel, we need add-on package “gdata”:

```
> library(gdata)
> megadata <-
read.xls(“/bu/dos/Multivariate_Statistics/anderson.csv”,header=T,sheet=5)
```

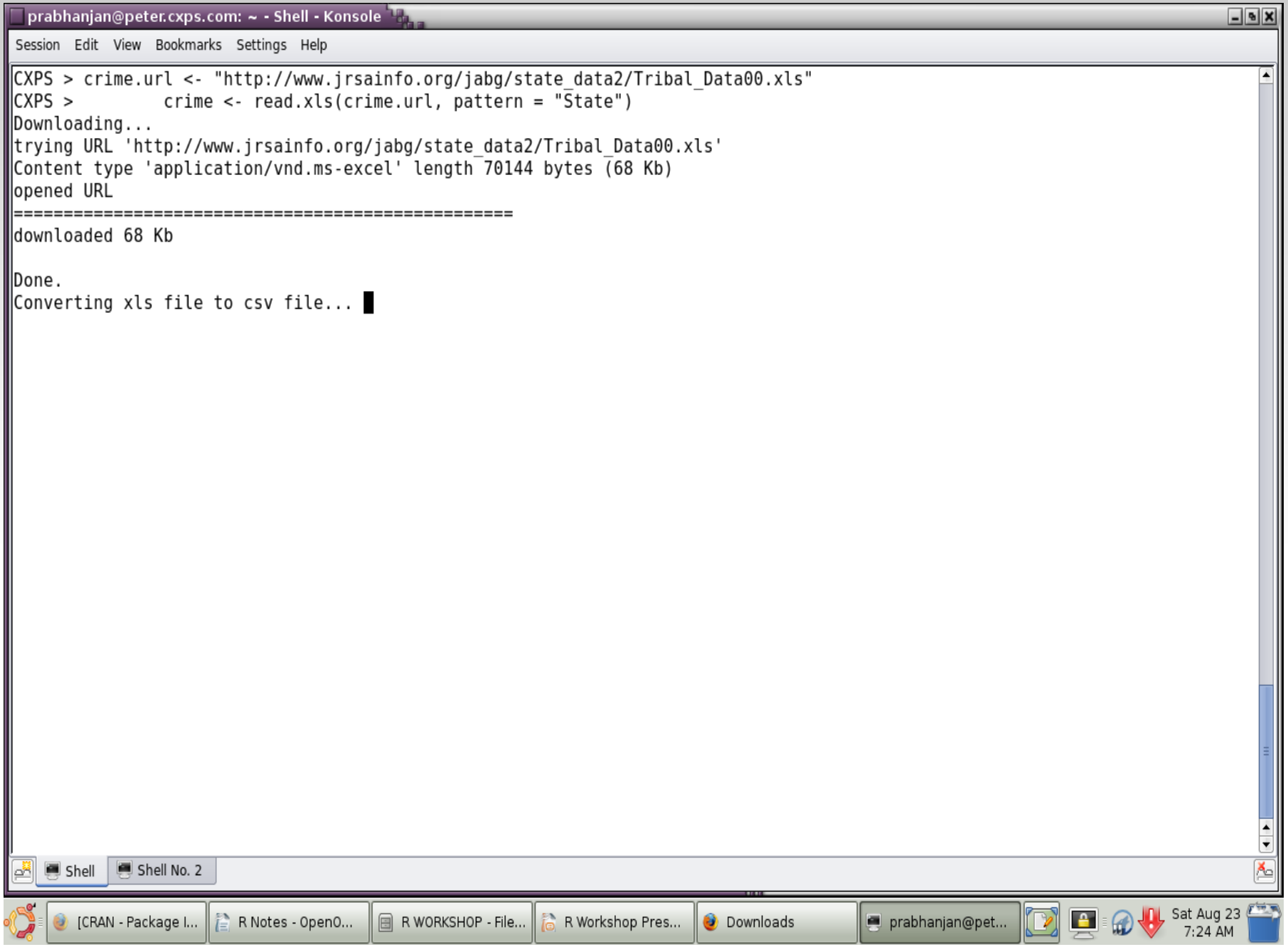
In fact, R first converts the .xls file to .csv, and then imports it.

Reading data from the web

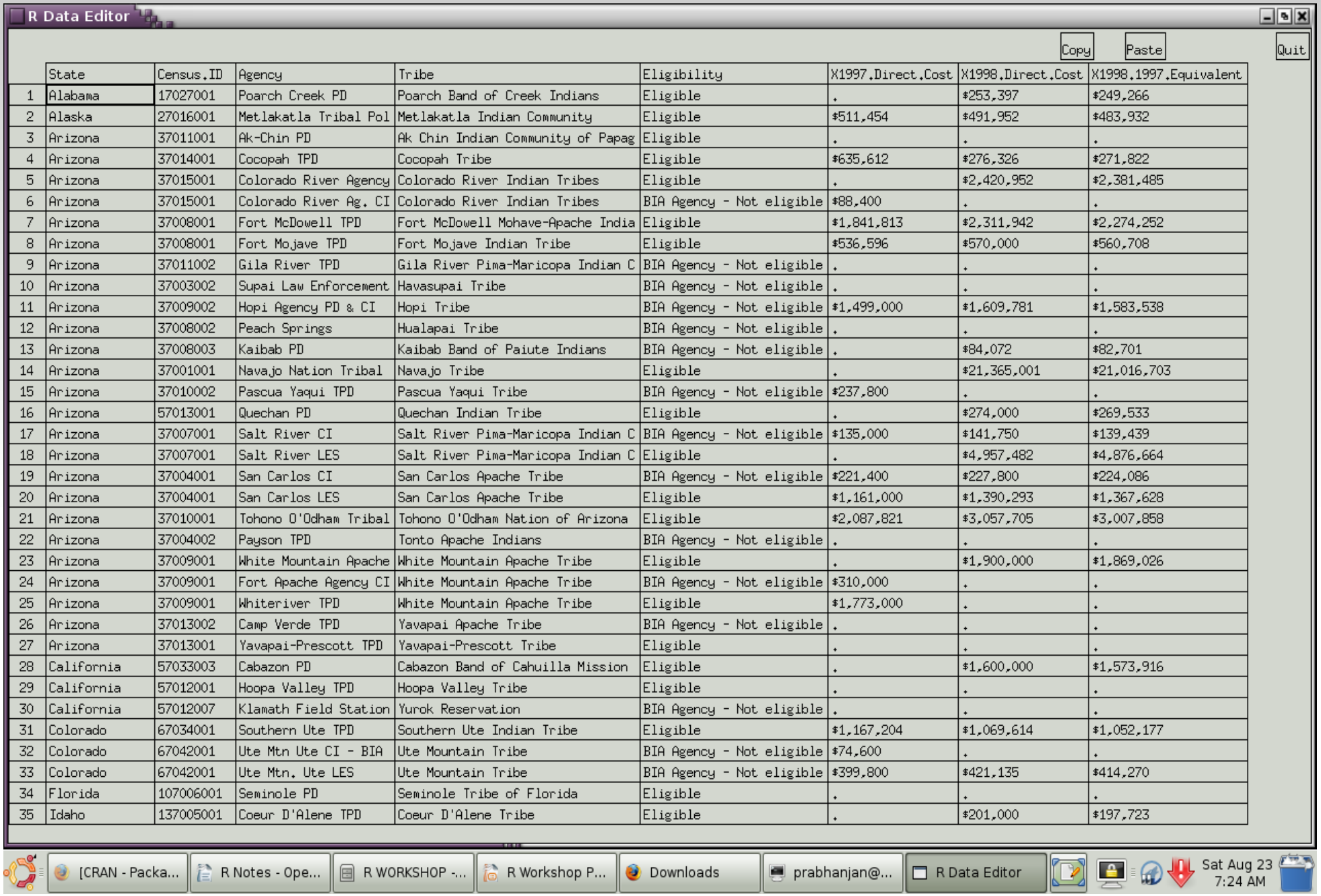
```
> crime.url<-
"http://www.jrsainfo.org/jabg/state_data2/Tribal_Data00.xls"
> crime <- read.xls(crime.url, pattern =
"State")
```



Screenshot 1



Screenshot 2



Screenshot 3

Some Basic Functions

c : a generic function which combines its arguments into a single vector or a list

gl : generates factor levels.
useful in Design of Experiments and Simulation
Arguments: the number of observations n,
the number of replications, say k.
Advantage: label the factors by names as desired

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
```

Some Basic Functions

is, is.na, and rm:
"is.na" is a particular case of the function "is"

- The function "is" derives its nomenclature from the common usage of that word
- Helps find 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
```

Some Basic Functions

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
```

Some Basic Functions

sort : arrange a numeric/complex R object in increasing or decreasing order.
To generate $n = 9$ *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
```

and then take the difference of the two consecutive numbers of the resulting array

sample : need a sample of size 3 from the set S= {1, 2, 3, ..., 6}.
Sample *with out* replacement, use the "sample" function:

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

Sample *with* replacement

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

Some Basic Functions

window : extracts elements from the specified between two times 'start' and 'end'.
Compute the moving average of a series of observations.

```
> 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
```

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
```

Some Basic Functions

table : Useful to obtain frequencies of a given factor.

```
> 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.

Also,

```
> x= c(2,2,2, 3, 3, 4,5); y=c(2,2,3,4,4,6,6)
> table(x,y)
   y
x   2 3 4 6
  2 2 1 0 0
  3 0 0 2 0
  4 0 0 0 1
  5 0 0 0 1
```

seq : gives a sequence of points in a given interval with a specified gap (difference)

```
> 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
```

Some Basic Functions

rep: Repeats the elements of a vector as specified by the user.

```
> rep(c("A","B"),2)
[1] "A" "B" "A" "B"
> rep(c("A","B"),each=2)
[1] "A" "A" "B" "B"
```

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.

```
> 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
```

Loops, if, ifelse, etc

```
for (i in 1:5) print(i^2)

j<-k<-0
for (i in 1:5) {
j<-j+1
k<-k+i*j
print(i+j+k) }

fac1<-function(x) {
f <- 1
if (x<2) return (1)
for (i in 2:x) {
f <- f*i
f }}
```

```
sapply(0:5,fac1)

y = 67
z <- ifelse (y < 0, -1, 1)
z
```

Vector Operations

```
> x= c(2,2,2,3,3,4,5); y=c(2,2,3,4,4,6,6)
> x + y # adds two vectors element-wise
[1]  4  4  5  7  7 10 11
> x - y # substracts two vectors element-wise
[1]  0  0 -1 -1 -1 -2 -1
> x * y # multiplies two vectors element-wise
[1]  4  4  6 12 12 24 30
> x/y # divides two vectors element-wise
[1] 1.0000000 1.0000000 0.6666667 0.7500000 0.7500000 0.6666667 0.8333333
> x/x # what happens now
[1] 1 1 1 1 1 1 1
```

```
> x= c(2,2,2,3,3,4,5); y=c(2,4)
> x+y #observe
[1] 4 6 4 7 5 8 7
Warning message:
In x + y : longer object length is not a multiple of shorter object length
> x-y #observe
[1]  0 -2  0 -1  1  0  3
Warning message:
In x - y : longer object length is not a multiple of shorter object length
> x*y #observe
[1]  4  8  4 12  6 16 10
Warning message:
In x * y : longer object length is not a multiple of shorter object length
```

Matrix Operations

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
```

Matrix Operations

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","Rupes
h")
> 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 Operations

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

Matrix Operations

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
> 1/A # not really the matrix inverse!
      var1       var2
[1,] 0.250 0.20000000
[2,] 0.125 0.07692308
```

Matrix Operations

Eigen values and eigen vectors 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'
```

Singular value decomposition for appropriate matrices :

```
> 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
```

Matrix Operations

We conclude with **Cholesky decomposition**,

```
> ( 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.

Working with packages*

Most of the packages contain data sets, and a list of which may be obtained by:
> try(data(package = "mypackage"))

To install a package

> install.packages(“mypackage”)
Need more:
> install.packages(c(“package1”, “package2”, ..., “packagen”))

And after some time:
> update.packages(“mypackage”)

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])

Managing R Sessions*

Good practice at the beginning of a session:
> rm(list=ls())
To know whats in a session:
> ls()
Your current directory:
> getwd()
And to change the same:
> (setwd("/home/myRsession))

R objects, say x and y, may be written to a file as
> save(x,y,file="xy.Rdata")

> save.image("mysession")

For the lazy person, the session will be simply saved by choosing "y" after executing
> q()

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

Exploratory Data Analysis

- Tukey (1962): “The Future of Data Analysis”
- S was developed at the Bell Labs to specifically handle EDA effectively
- R is unsurprisingly a very effective tool for the users
- Mosteller and Tukey (1977)
- Tukey (1977)
- Velleman and Hoaglin (1984)

Exploratory Data Analysis

Graphical Methods

- Boxplot
- Histogram
- Stem-and-leaf Plot
- The *x*-*y* Plot

Quantitative Techniques in EDA

- Trimean
- Letter Values
- Resistant Values
- Median Polish
- Resistant Smooth, and
- Rootogram

EDA: Graphical Methods
The Boxplot

- The box plot are also known as *box and whisker plots*
- The five important summaries
  - Sample Minimum
  - Lower Quartile (Q1)
  - Median (Q2)
  - Upper Quartile (Q3)
  - Sample Maximum

The Box Plot

Example 1. Birth Weights of Infants with Severe Idiopathic Respiratory Distress Syndrome (SIRDS)

http://openlearn.open.ac.uk/mod/resource/view.php?id=165503

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

The Box Plot

```
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
```

The Box Plot

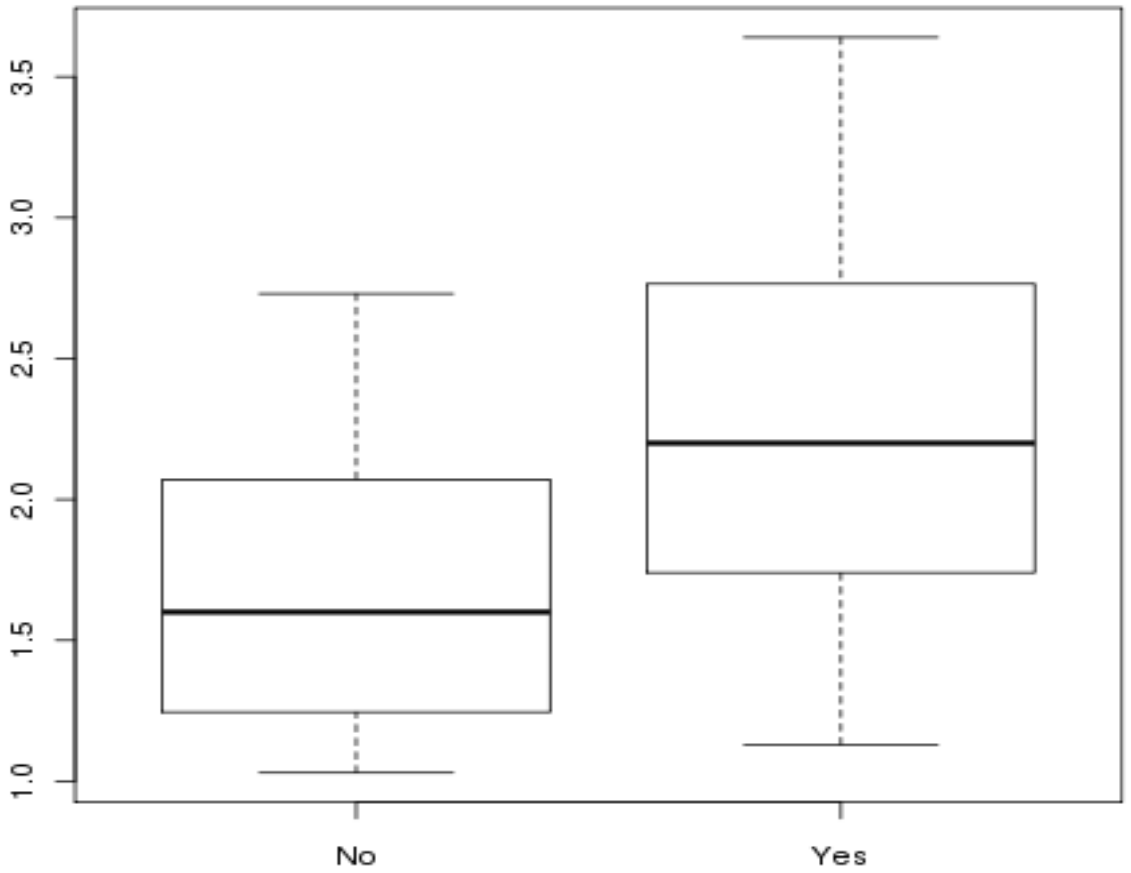


Figure 1. The boxplot of infants from the SIRDS.csv file

The Box Plot

Inference from the Boxplot.

- the median birth weight of infants who survived is greater than that of those who died
- interquartile ranges, indicated by the lengths of the boxes, are reasonably similar
- the overall range of the data set is greater for the surviving infants
- 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.
- 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.

The Box Plot

Example 2. The Effect of Insecticides. McNeil (1977).

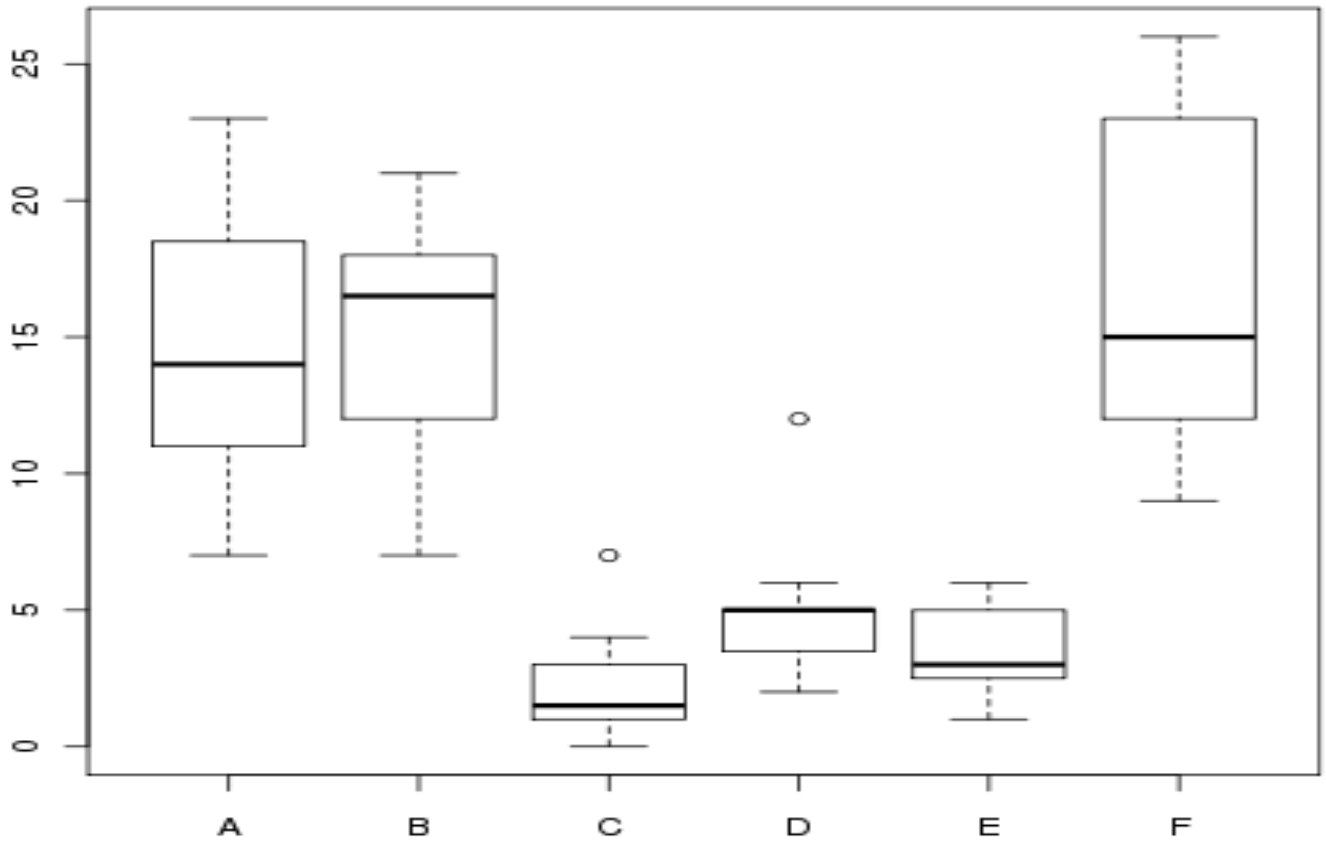
Six insecticides, labeled A-F

the number of insects found dead after using them are counted.

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 Box Plot



Clearly, the frequencies show that Insecticides A, B, and F are very powerful as compared with C, D, and E.

The Histogram

- The usuage of histogram is probably as old as data analysis itself
- 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.

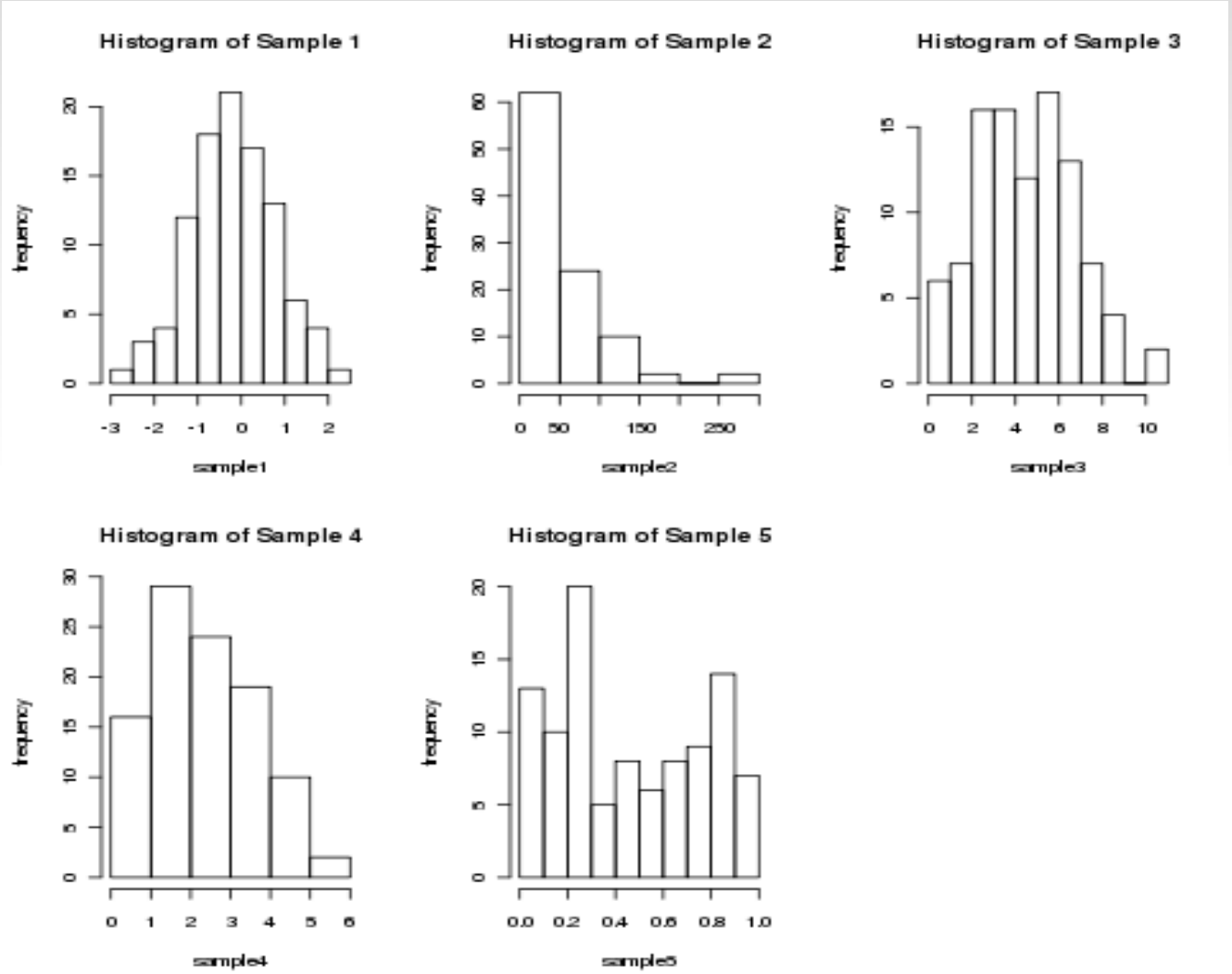
Example 3. In the file “sample.csv”, we have data from five different probability distributions. Towards understanding the plausible distribution of the samples, we plot the histogram and see how useful it is.

```
> sample=read.csv("/mypath/sample.csv",header=T)
> tiff("hist_samples.tiff")
```

The Histogram

```
 > 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()
```

The Histogram



The Histogram

```
> 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
```

The Histogram

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*.

The Histogram

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.

The Rootogram

- Generally, in histograms, bar height varies more in bins with long bars than in bins with short bars.
- In frequency terms, the variability of the counts increases as their typical size increases
- A re-expression can approximately remove the tendency for the variability of a count to increase with its typical size
- **Rootogram** arises on taking the re-expression as the square-root of the frequencies

The Rootogram Plot for the Militiamen Chest Data

Example. The "***Militiamen Chests***" data set is one of the famous data sets and dates back to 1846.

- Of course, the data is not *completely available* to us.
- Bin values and the respective frequencies is available
- Thanks to Velleman and Hoaglin, and Internet of course
- http://lib.stat.cmu.edu/DASL/Datafiles/MilitiamenChests.html

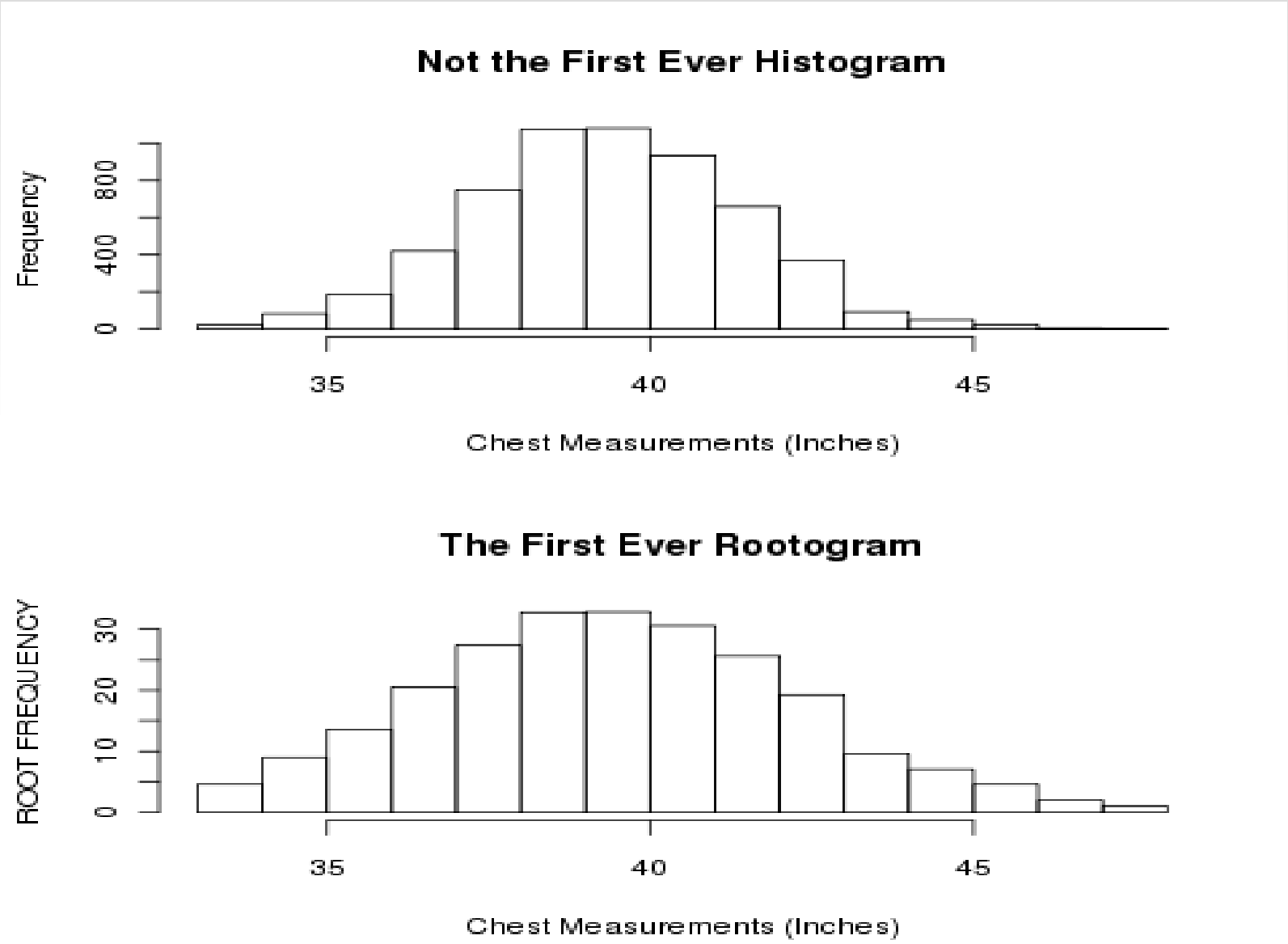
Let us reconstruct the data.

```
> chest=read.csv("Chest_VH.csv",header=T)
> attach(chest)
> names(chest)
[1] "Chest" "Count"
> militiamen=rep(Chest,Count)
> length(militiamen)
[1] 5738
```

The Rootogram Plot for the Militiamen Chest Data

```
bins=seq(33,48)
bin.mids=(bins[-1]+bins[-length(bins)])/2

par(mfrow=c(2,1))
h=hist(militiamen,breaks=bins,xlab="Chest Measurements (Inches)",main="Not
the First Ever Histogram")

h$counts=sqrt(h$counts)

plot(h,xlab="Chest Measurements (Inches)",ylab="ROOT
FREQUENCY",main="The First Ever Rootogram")
```

The Rootogram Plot for the Militiamen Chest Data



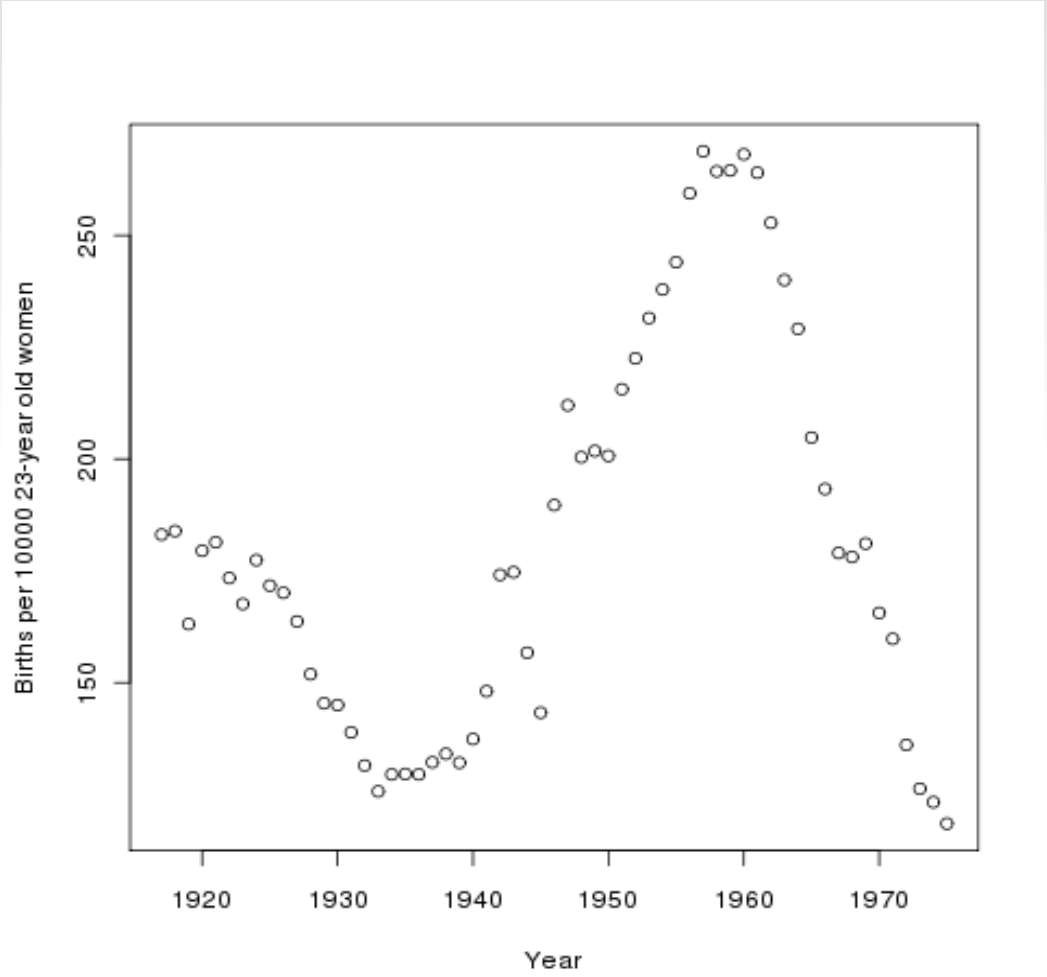
x-y (scatter) Plot

- In the EDA literature, the scatter plots are referred as *x-y* plot
- See Chapter 4 of Velleman and Hoaglin (1984)

Example 4. The births per 10000 23-year old women is available for the period 1917-1975

```
> birthrate=read.csv("/home/prabhanjan/Desktop/xy_data.csv",header=T)
> tiff("first_plot.tiff")
> plot(birthrate,xlab="Year",ylab="Births per 10000 23-year old women")
> dev.off()
null device
          1
```

x-y (scatter) Plot



The Stem-and-Leaf Plot

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.

The Stem-and-Leaf Plot

Example 5. 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.

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

The Stem-and-Leaf Plot

```
> 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
```

Quantitative Techniques in EDA
TRIMEAN

Trimean, as a measure of location, is the weighted average of median and two quartiles given by $$TM = \frac{Q_1 + 2Q_2 + Q_3}{4}$$

where $X_1, X_2, ..., X_n$ are the *i*-th quantiles. It can be equivalently written as the average of median and mid-hinge:

$$TM = \frac{1}{2}\left(Q_2 + \frac{Q_1 + Q_3}{2}\right)$$

Example 1. We consider the data of Sample 1 from Example 1 of subsection 3.2.2. The trimean is then computed below:

```
> qs=quantile(sample1,c(.25,.5,.75))
> (tm=as.numeric(qs[1]+2*qs[2]+qs[3])/4)
   -0.179375
```

Weisberg (1992) summarizes trimean as "a measure of the center (of a distribution) is that it combines the median's on center values with the midhinge'sattention to the extremes."

LETTER VALUES

As the emphasis is on the *data*, we are also interested to know more about each *datum.*
Some natural examples of useful datum are minimum, maximum, outlier, etc. Median is also a useful datum when the size of data is odd.

Depth. Suppose that the observations are arranged in an ascending order. For each datum, *depth* is the minimum of the position from either end of the batch.
Depth for a datum x is denoted by *d(x).* By definition, depth of median (M) is $d(M)=(n+1)/2$.

Hinges. Lets ask this question. Median is equally distanced between both the extremes, namely minimum and maximum. What is then equally distanced between the median and the minimum, and median and maximum respectively? The answer is given by hinges, denoted by *H*.
Again , by its definition, $d(H)=([d(M)]+1)/2$.

LETTER VALUES

We illustrate this concept with a superficial data values. Suppose that the data is make some preparations using R.

```
> x=c(13,17,11,15,12,7,24)
> min(x);max(x);median(x)
[1] 7
[1] 24
[1] 13
> tab=cbind(order(x),x[order(x)],c(1:7),c(7:1))
> colnames(tab)<-
c("x_label","x_order","Position_from_min","Position_from_max")
```

LETTER VALUES

```
> tab
     x_label    x_order Position_from_min    Position_from_max
[1,]       6          7                 1                    7
[2,]       3         11                 2                    6
[3,]       5         12                 3                    5
[4,]       1         13                 4                    4
[5,]       4         15                 5                    3
[6,]       2         17                 6                    2
[7,]       7         24                 7                    1
```

In the given data set, the position of the value 7 is 6, that is, so on. To calculate the depth of the datum 7, we take the minimum of its position from the minimum and the maximum. That is, depth (7) = min(1, 7) = 1, depth (11) = min(2,6) = 2, depth (12) = min(3,5) = 5, ..., depth (24) = min (7, 1) = 1.

The minimum and maximum values, also called as extremes, have a depth of 1, the second largest and second smallest have depth of 2, and so on. Thus, two observations, namely, the *i-th* and *(n + 1 – i)-th* ordered observations in ascending order have depth *i*.

Difference Between Quartiles and Hinges

- The depth of the hinges is calculated from the depth of the median
- This results in the fact that the hinges often lie slightly closer to the median than do the quartiles.
- This difference is quite small, and the arithmetic required to calculate the depth of the hinges is simpler.

LETTER VALUES

After finding the mid (median), the quarters (hinges), the next natural step is to find the **eights**, then the **sixteens**, and so on.

Naturally, we define

$$d(E) = ([d(H)]+1)/2$$

Illustration. If we have a data of size 100, the depth of median, hinges, eights, ... are approximately 50, 25, 23, ... This journey is to be continued till the depth reaches 1, in which case we have almost exhaustive information about the sample data set.

We will see more about this in the next section.

Letter Values

- The median, the hinges, and the eighths-are the start of the sequence of letter values, labeled with single letters-M, H, and E.
- D. C, B. A, Z, Y. X, W, ana so on

Formulas for the depth of the letters are then:

$$d(D) = ([d(E)]+1)/2$$
$$d(C) = ([d(D)]+1)/2$$
$$d(B) = ([d(A)]+1)/2$$
$$\ldots$$

LETTER VALUES

Example 6. Area of New Jersey Counties.

```
areanj=c(569, 234, 819, 221, 267, 500, 130, 329, 47, 423, 228, 312, 476,
468, 642, 192, 365, 307, 527, 103, 362)
counties= c("Atlantic", "Bergen", "Burlington", "Camden", "Cape",
"Cumberland", "Essex", "Gloucester", "Hudson", "Hunterdon", "Mercer",
"Middlesex", "Monmouth", "Morris", "Ocean", "Passaic", "Salem",
"Somerset", "Sussex", "Union", "Warren")
library(LearnEDA)
lval(areanj)
  depth  lo  hi  mids spreads
1  11.0 329 329 329.0       0
2   6.0 228 476 352.0     248
3   3.5 161 548 354.5     387
4   2.0 103 642 372.5     539
5   1.0  47 819 433.0     772
```

LETTER VALUES

Exhibit 2–3 Letter-Value Display for the Area of New Jersey Counties (in square miles) Shown in Exhibit 2–1

$n = 21$

		Lower	*Upper*	*Mid*	*Spread*
M	11	329		329	
H	6	228	476	352	248
E	3.5	161	548	354.5	387
D	2	103	642	372.5	539
	1	47	819	433	772

RESISTANT LINE

Consider the problem of fitting a line on a simple paired data:

$$y = a + bx,$$

where x is the regressor, y is the regressand, and a and b are intercept and slope terms respectively.

The statistical problem here is inference related to a and b.

Galton first solved this problem. Lets have brief peek into his data and solution.

```
library(UsingR)
data(galton)
names(galton)
attach(galton)
galtonlm=lm(child~parent)
summary(galtomlm)
anova(galtonlm)
plot(galtonlm)
```

RESISTANT LINE

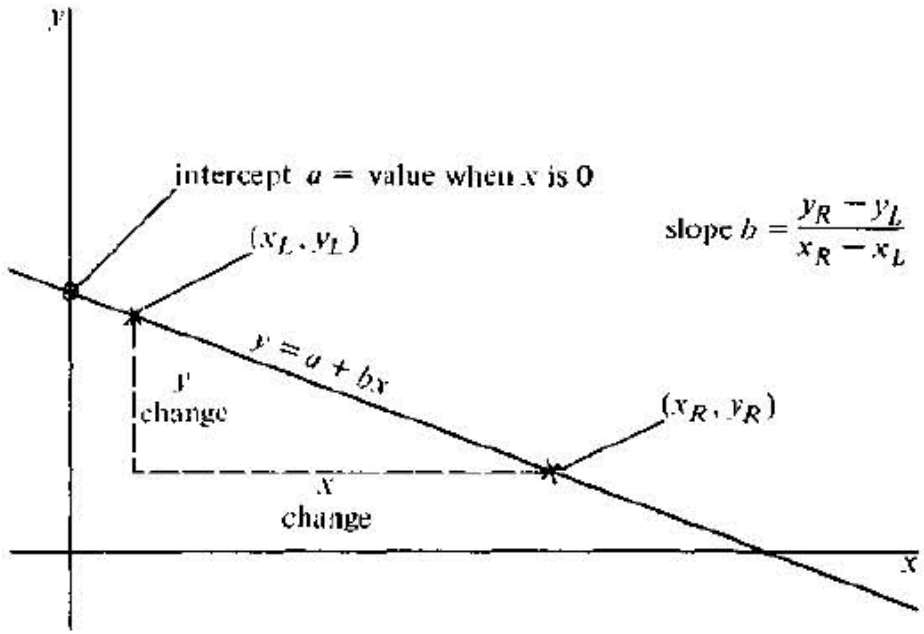
```
rp=read.csv("rocket_propellant.csv",header=T)
attach(rp)
tiff("rocket.tiff")
plot(Age_of_Propellant, Shear_Strength)
dev.off()

rplm=lm(Shear_Strength ~ Age_of_Propellant)
summary(rplm)
anova(rplm)

Age_of_Propellant[19]=20000000
rplm1=lm(Shear_Strength~ Age_of_Propellant)
summary(rplm1)
anova(rplm1)
```

RESISTANT LINE

$$b = \frac{y \text{ change}}{x \text{ change}} = \frac{y_R - y_L}{x_R - x_L}.$$

$$a = y_L - bx_L.$$

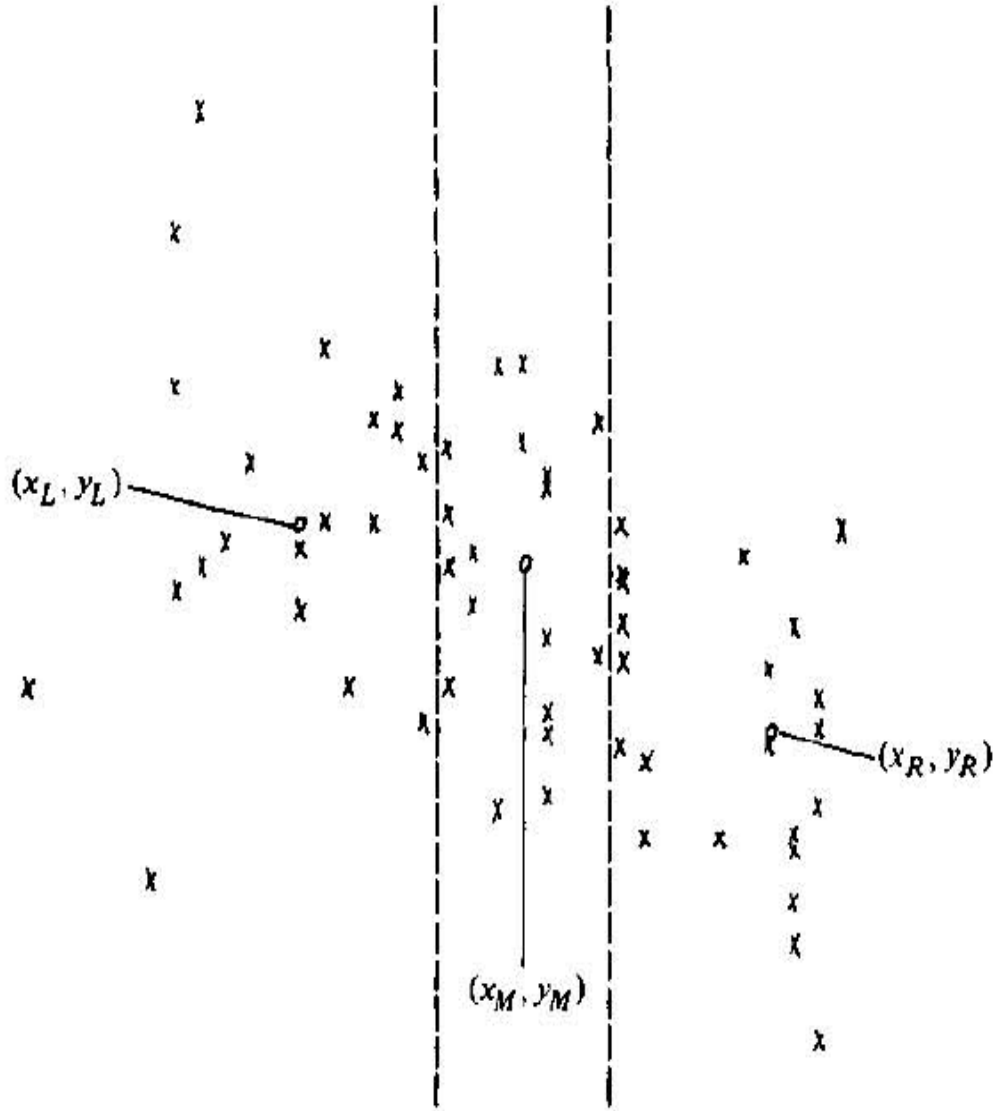
Exhibit 5–1 Finding the Slope and Intercept of the Line $y = a + bx$



Note: In this example y_R is smaller than y_L so $y_R - y_L$ is negative and the slope, b, is also negative.

RESISTANT LINE

Exhibit 5-2 Dividing a Plot into Thirds and Finding Summary Points



The Theory of Resistant Lines

$$(x_L, y_L)$$
$$(x_M, y_M)$$
$$(x_R, y_R).$$

$$b = \frac{y_R - y_L}{x_R - x_L} \qquad a - y_M - bx_M$$

$$a_L = y_L - bx_L$$
$$a_M = y_M - bx_M$$
$$a_R = y_R - bx_R$$

$$a = (\tfrac{1}{3})(a_L + a_M + a_R) = (\tfrac{1}{3})[(y_L + y_M + y_R) - b(x_L + x_M + x_R)]$$

Residuals for Resistant Line

residual = data − fit.

$$r_i = y_i - (a + bx_i).$$

RESISTANT LINE

```
> rline(Shear_Strength, Age_of_Propellant)
$a
[1] 12.55275
$b
[1] -0.02286165
$xC
[1] 2182.85
$half.slope.ratio
[1] 1.093622
$residual
 [1]  2.395145e+00 -3.410196e-01 -1.508718e+00  1.668421e+00 -6.489206e+00
 [6] -4.401741e+00  2.344889e+00 -1.087551e+00 -1.050815e+00  1.355867e-01
[11]  4.374589e-02 -3.848627e+00  3.232867e+00  7.156615e-01 -9.862699e-02
[16]  2.490100e+00 -1.259132e+00  3.507621e-01  2.000000e+07 -8.638221e-01
```

RESISTANT LINE

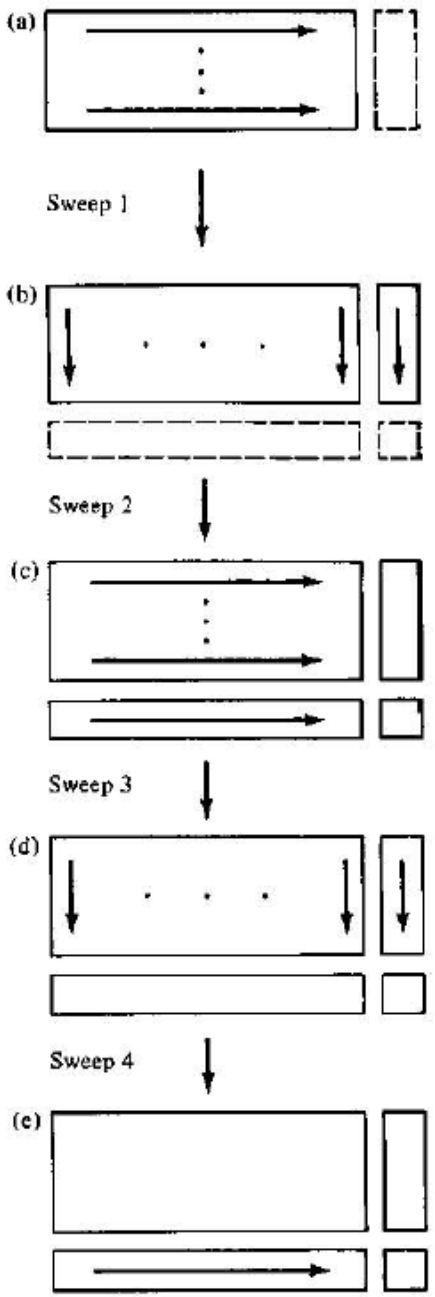
```
> Age_of_Propellant[19]=20000000
>
> rline(Shear_Strength, Age_of_Propellant)
$a
[1] 12.55275
$b
[1] -0.02286165
$xC
[1] 2182.85
$half.slope.ratio
[1] 1.093622
$residual
 [1]  2.395145e+00 -3.410196e-01 -1.508718e+00  1.668421e+00 -6.489206e+00
 [6] -4.401741e+00  2.344889e+00 -1.087551e+00 -1.050815e+00  1.355867e-01
[11]  4.374589e-02 -3.848627e+00  3.232867e+00  7.156615e-01 -9.862699e-02
[16]  2.490100e+00 -1.259132e+00  3.507621e-01  2.000000e+07 -8.638221e-01
```

Median Polish*

- Median polish is to ANOVA like Resistant Line is to a simple linear regression model
- Finds an additively-fit model for data in a two-way layout table
- The thought process here is this:

$$output = row\ effect + column\ effect + overall\ median$$

Exhibit 8–4 Median Polish as a Sequence of Four Sweeping Operations, Starting with the Rows of the Data



Median Polish

```
> library(LearnEDA)
> smoker=read.csv("smokers_mp.csv",header=T)
> smoker=smoker[-c(1,8),]
> medpolish(smoker[,2:4])
1 : 4.19
2 : 4.12
Final: 4.12
Median Polish Results (Dataset: "smoker[, 2:4]")
Overall: 0.47
Row Effects:
          2           3           4           5           6
0.000000e+00 -3.800000e-01 -1.100000e-01  5.551115e-17 -2.100000e-01
          7           9          10          11          12
2.500000e-01 -3.100000e-01 -1.800000e-01  8.000000e-02  4.130000e+00
         13          14          15          16          17
1.790000e+00  1.470000e+00 -3.300000e-01 -2.000000e-02  1.010000e+00
```

Column Effects:

None	Grams_1_14	Grams_14_25
-0.03	0.00	0.00

Residuals:

	None	Grams_1_14	Grams_14_25
2	-0.37	0.00	0.39
3	-0.06	0.04	0.00
4	0.08	0.00	-0.26
5	0.00	0.07	-0.10
6	0.32	0.00	-0.04
7	-0.05	0.00	0.04
9	-0.13	0.00	0.02
10	-0.14	0.00	0.10
11	0.17	0.00	-0.01
12	-0.35	0.04	0.00
13	0.00	-0.11	0.21
14	0.10	0.00	-0.08
15	-0.11	0.00	0.02
16	0.00	0.37	0.00
17	0.00	0.33	-0.01

PROBABILITY

- The Classical Birthday Problem
- Conditional Probability and Intel Fiasco
- Bayes formula with Illustration
- p, d, q of Binomial Distribution
- The Uniform Distribution

Probability Surprises

The Classical Birthday Problem

“What is the probability that two students of a batch have birthday on the same day?”

Event E = “at least two employees have same beedi”

As with many scenarios, its easier to work with the complementary event

E' = “all the employees have different birthday”

The number of ways that all have different birthdays is the number of ways of choosing *k* different days from 365 days, that is,

$$^{365}P_k$$

and the total number of ways of arranging *k* birthdays is

$$365^k$$

Hence, the probability of *k* different birthdays is

$$^{365}P_k / 365^k$$

and the complement of which gives the probability of at least 2 birthday are same

$$1 - {}^{365}P_k / 365^k$$

R Program for Birthday Problem

```
> # probability of different birthdays k = 2
> k=2
> prod(365:(365-k+1))/365^k
[1] 0.9972603
> # probability of at least two birthdays being same k = 2
> 1- prod(365:(365-k+1))/365^k
[1] 0.002739726
```

... Put k = 5, 10, 20, 30, and 40, and then

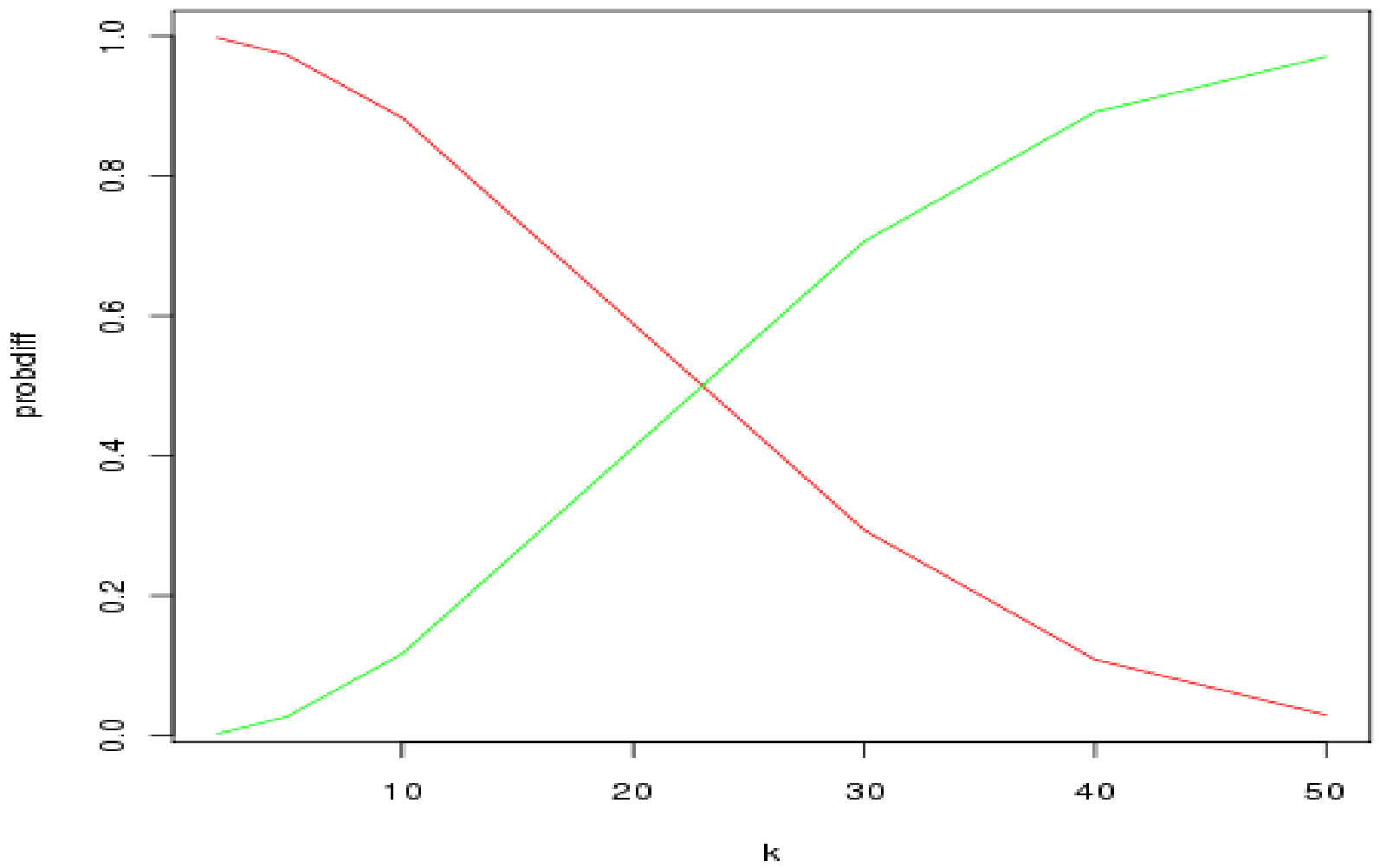
```
> # probability of different birthdays k = 50
> k=50
> prod(365:(365-k+1))/365^k
[1] 0.02962642
> # probability of at least two birthdays being same k = 50
> 1- prod(365:(365-k+1))/365^k
[1] 0.9703736
```

Using Loop

```
k=c(2, 5, 10, 20, 30, 40, 50)
probdiff=c(); probat2same=c()
for(i in 1:length(k)){
                        kk=k[i]
                        probdiff[i]=prod(365:(365-kk+1))/(365^kk)
                        probat2same[i]= 1- prod(365:(365-kk+1))/(365^kk)
                            }
plot(k, probdiff, col="red", "l")
lines(k, probatsame, col="green","l")
```

Size *k*	Probability of different birthdays	Probability of at least two same birthdays
2	0.99726027	0.00273973
5	0.97286443	0.02713557
10	0.88305182	0.11694818
20	0.58856162	0.41143838
30	0.29368376	0.70631624
40	0.10876819	0.89123181
50	0.02962642	0.97037358

Plot of the Birthday Probabilities



Conditional Probability

Given two events, the probability of an event A given that event B has occurred is given by

$$P(A/B) = P\frac{(A \cap B)}{P(B)}.$$

	Compiles on First Run	Does not Compile on First Run	
C++	72	48	120
Java	64	16	80
	136	64	200

Conditional Probability

Given two events, the probability of an event A given that event B has occured is given by

$$P(A/B)=P\frac{(A\cap B)}{P(B)}.$$

P(C/C++)=0.60

P(DC/C++)=0.40

	Compiles on First Run	Does not Compile on First Run	
C++	72	48	120
Java	64	16	80
	136	64	200

P(C/Java)=0.80

P(DC/Java)=0.20

Independent Events and the Intel Fiasco

An event E_1 is said to be independent of event E_2 if

$$P(E_1/E_2)=P(E_1)$$

In simple words, E_2 does not influence E_1.

The Intel Fiasco. 1994.

- A chip results in incorrect division 1 in 9 billion opportunities
- Intel transformed that as an error occuring once in 27000 years

E: occurence of error

$$P(E)=\frac{1}{9{,}000{,}000{,}000}$$

$$P(no\ error)=\left(1-\frac{1}{9{,}000{,}000{,}000}\right)$$

Now, probability of no error in two divides is

$$P(\text{no error in two divides})=P(\text{no error in first divide})\times P(\text{no error in second divide})$$

$$=\left(1-\frac{1}{9{,}000{,}000{,}000}\right)^2$$

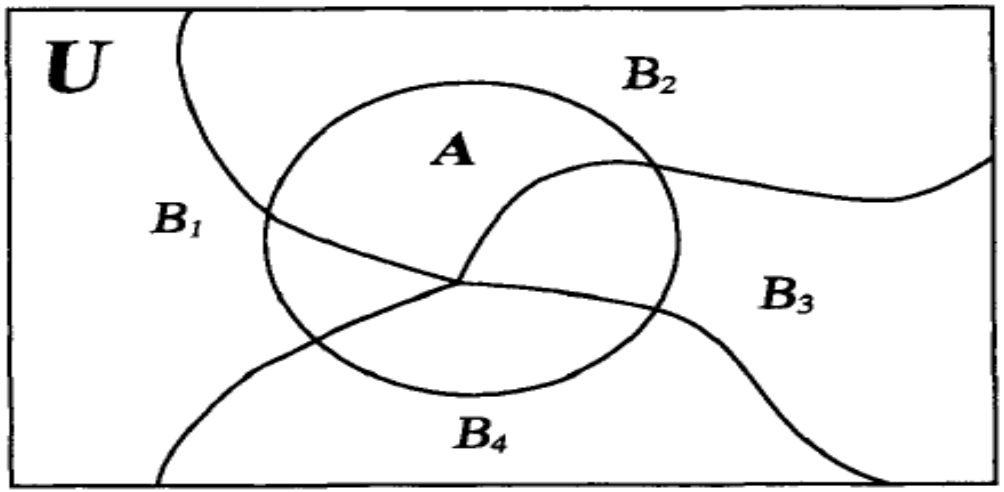
Now, lets calculate the probability that a billion divides contain no error:

```
> proberror= 1 - 1/9000000000
> (noerrorbill=proberror^1000000000)
[1] 0.8948393
```

The Bayes Formula

$B_1, B_2, \ldots, B_m$

$B_i \cap B_j = \phi, \quad , \cup_{i=1}^{m} B_i = \Omega$



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

$$P(B_i/A) = P\frac{(B_i \cap A)}{P(A)}$$

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

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

Example of Bayes Formula

50% of the documents in the company are written in Word, 30% in Latex, and 20% in Html format

$$P(A_W)=0.5, P(A_L)=0.3, P(A_H)=0.2.$$

In each of these formats exceeding 10 pages for Word, Latex, and Html are respectively 40%, 20% and 20%

$$P(E/A_W)=0.4, P(E/A_L)=0.2, P(E/A_H)=0.2.$$

Thus, the probability of a randomly selected document exceeding 10 pages is

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

Example of Bayes Formula

```
> 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
```

Standard Probability Distributions

Binomial Distribution.

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

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

The below codes helps us in understanding the binomial distribution:

```
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()
```

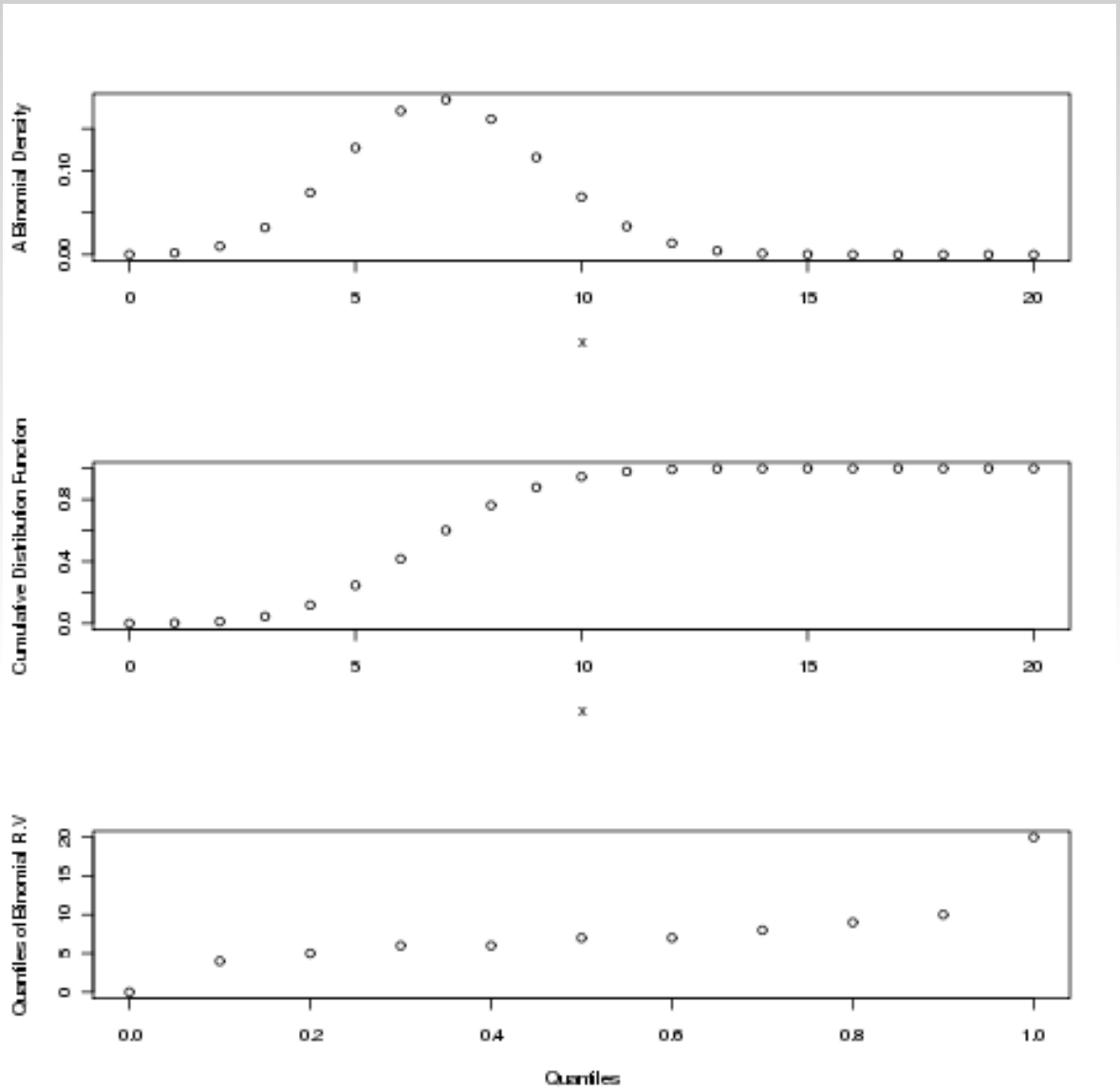


Figure. The Density, Cumulative Distribution, and Quantile Plot for a Binomial Random Variables

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. Suppose X ~ 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
```

Some Estimation Problems

Estimation of Shape Parameter of a Gamma Distribution

$$f(x;k,\theta) = x^{k-1}\frac{e^{-x/\theta}}{\theta^k\,\Gamma(k)} \text{ for } x > 0 \text{ and } k,\theta > 0.$$

Suppose that θ is known to be 30, and we are interested in estimation of the shape parameter *k*.

31.37	0.52	1.79	7.91	5.59
4.24	0	23.65	7.03	4.23
0.83	0.21	0.07	26.32	3.44
24.01	25.33	19.24	35.75	0.06

Shape Parameter of a Gamma Distribution

The log-likelihood function is then given by

$$\text{log-lik}(k|\boldsymbol{x}) = -\sum_{i=1}^{20} x_i/30 - (k-1)\sum_{i=1}^{20} \log(x_i) - 20\text{k} \times \log(30) - 20 \times \log(\Gamma(k))$$

On entering the values in the above equation, we get

$$\text{log-lik}(k|\boldsymbol{x}) = 7.38 - (k-1)20.79 - 68.02k - 20 \times \log(\Gamma(k))$$

The above equation is rewritten as

$$\text{log-lik}(k|\boldsymbol{x}) = 28.17 - 88.21k - 20\log(\Gamma(k))$$

Shape Parameter of a Gamma Distribution

```
> log_lik= function(k) {28.17-88.21*k-20*gamma(k)}

> optimise(log_lik,c(0,1),maximum=T)
  $maximum
  [1] 0.4493373
  $objective
  [1] -50.88713
```

Of course, if your strength is American english, then

```
> optimize(log_lik,c(0,1),maximum=T)
$maximum
[1] 0.4493373

$objective
[1] -50.88713
```

Fitting of Probability Distributions

Lets consider the rates of lung cancer in four Danish cities. This example was first considered by Erling Andersen in 1977, see Dalgaard (2008).

```
> library(ISwR)
> library(MASS)
> library(vcd)
> eba1977
        city   age  pop cases
1  Fredericia 40-54 3059    11
2     Horsens 40-54 2879    13
3     Kolding 40-54 3142     4
...
23    Kolding   75+  659    12
24      Vejle   75+  619     7
> fitdistr(eba1977[,4],"Poisson")
    lambda
  9.3333333
 (0.6236096)
```

```
> goodfit(eba1977[,4],"poisson")
Observed and fitted values for poisson
distribution
with parameters estimated by `ML'
 count observed     fitted
     2        1 0.09243568
     4        1 0.67101456
     5        1 1.25256051
     6        1 1.94842746
     7        3 2.59790327
     8        2 3.03088715
     9        1 3.14314223
    10        4 2.93359942
    11        5 2.48911466
    12        2 1.93597807
    13        1 1.38993297
    14        1 0.92662198
    15        1 0.57656479
```

Fitting of Poisson Distribution

Yes, thats not really good enough. You ask something of this type!

And also, futher you demand

$$\chi^2 = \sum \frac{(O-E)^2}{\sigma^2}$$

$$\chi^2_{\text{red}} = \frac{\chi^2}{\nu} = \frac{1}{\nu}\sum \frac{(O-E)^2}{\sigma^2}$$

So, here we go

```
> gfpois=goodfit(eba1977[,4],"poisson")
> chival=sum((gfpois$observed-gfpois$fitted)^2/gfpois$fitted)
```

Unfortunately, I forgot this

$$Pr\left(\chi^2 > \chi^2_{obs}\right)$$

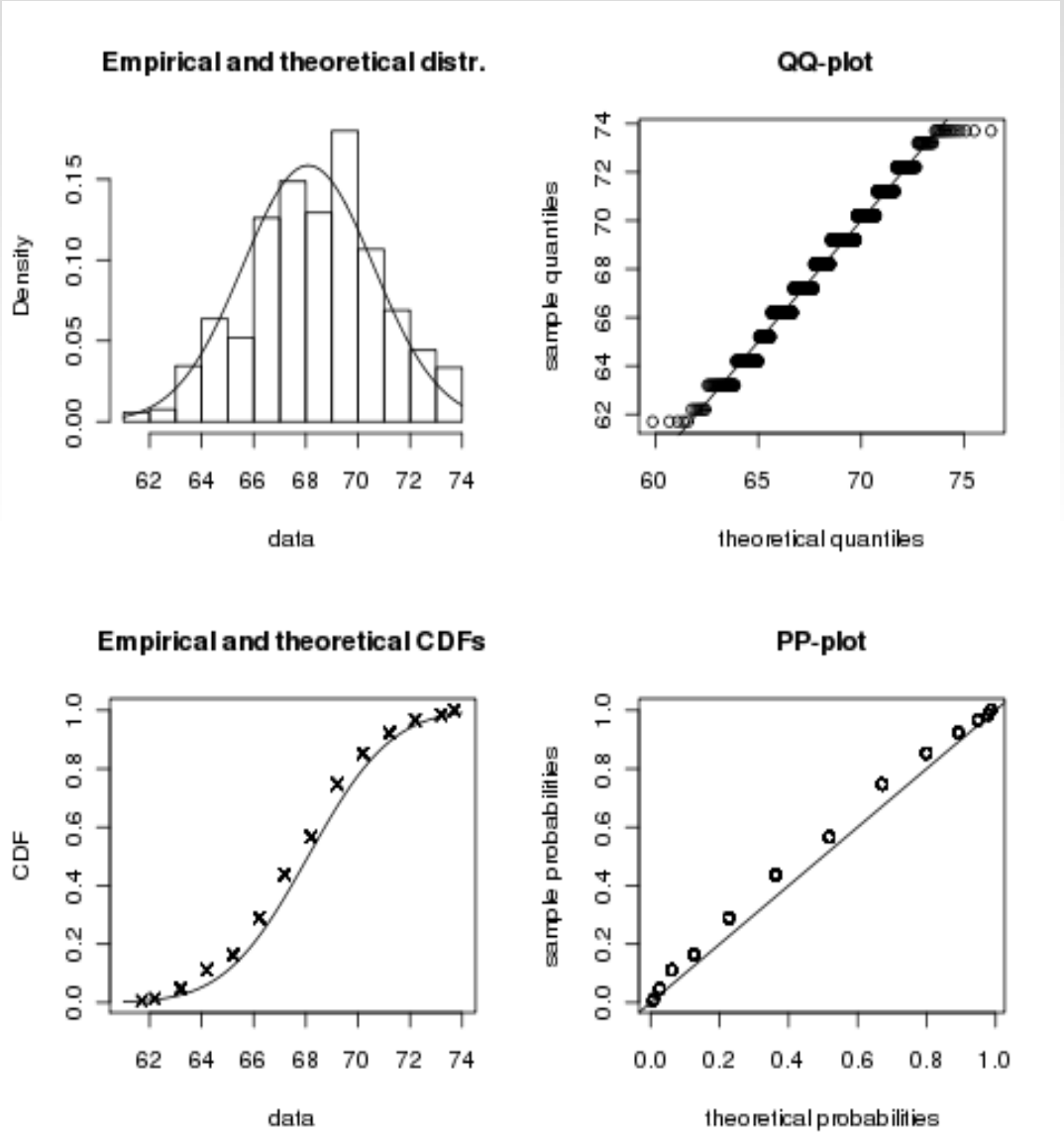
```
> 1-pchisq(chival,gfpois$df)
[1] 0.1914753
```

Fitting a Normal Distribution

- Consider the data on the height of son from the Galton data

```
> library(UsingR)
> library(fitdistrplus)
> hs=galton[,1]
> fitdistr(hs,"normal")
> fhs=fitdist(hs[,1], “norm”)
> tiff("Fit_normal.tiff")
> plot(fhs)
> dev.off()
null device
          1
```

Fitting a Normal Distribution



Classical Tests in R

The Z-Test

Suppose that $x_1, x_2, \ldots, x_n$ is a random sample from a normal distribution.

Assuming the variance is known, we are interested in testing for the mean $H_0: \mu = \mu_0$

- If the variance is known, use the Z-test.
- If the variance is unknown and the sample size is large, at least n =30, we use the Z test.

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

  The Z-test is then given by

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

The Z-Test

Example. (Freund and Wilson, 2003).
The mean weight of peanuts put in jars is required 8 oz.
The variance of the weights is known to be 0.03
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 .
We can test this in R as follows:

```
> library(PASWR)
> 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.

One-sample *t*-Test

The *t*-test is given by

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

The t-test 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 forand 95% confidence interval as an output;

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

Two sample *t*-Test

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

- This data set is exactly 105 years old
- Let us 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!

```
> 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
```

One-sample *t*-Test

```
> t.test(sleep[1:10,1],mu=2.3)

        One Sample t-test
data:  sleep[1:10, 1]
t = -2.7398, df = 9, p-value = 0.02286
alternative hypothesis: true mean is not equal to 2.3
95 percent confidence interval:
 -0.5297804  2.0297804
sample estimates:
mean of x
     0.75
```

We thus reject the null hypothesis.

Two-sample *t*-Test

We explain the two-sample test in some detail.

- $x_1, x_2, \ldots, x_{n_1}$ a random sample from $N(\mu_x, \sigma_x^2)$
- $y_1, y_2, \ldots, y_{n_2}$ *a random sample from* $N(\mu_y, \sigma_y^2)$

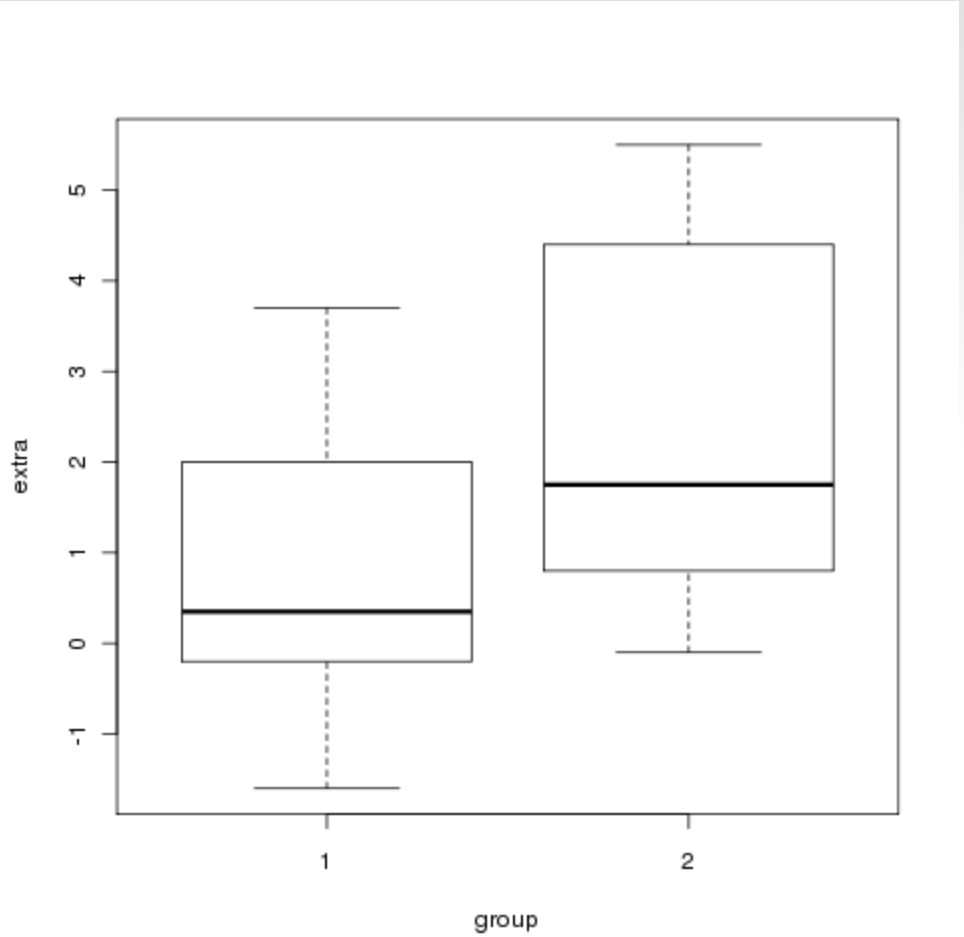
The two-sample t-test is then given by $H_0 : \mu_x = \mu_y$

which has t-distribution with degrees of freedom, with s_{xy} being the pooled standard deviation.

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

Two-sample *t*-Test

```
> 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
```



Testing for Variances

```
> var.test(extra~group,data=sleep)

        F test to compare two variances

data:  extra by group
F = 0.7983, num df = 9, denom df = 9, p-value = 0.7427
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.198297 3.214123
sample estimates:
ratio of variances
         0.7983426
```

Testing Proportions

Testing for Single Proportion.

$$X_1, X_2, \ldots, X_n \sim B(1, p)$$

The hypothesis of interest:

$$H_0 : p = p_0$$

The z-test is then given by

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

Example 1. Testing for single proportion

- 432 heads turn up on a throw of 1000 times of a coin
- 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*.

- India has won 23 out of 62 test matches against Australia
- Pakistan has 16 test matches in 58 matches
- 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
```

Nonparametric Tests

The Kolmogorov-Smirnov Test

$X_1, X_2, \ldots, X_n$: a random sample from a probability distribution F,

$Y_1, Y_2, \ldots, Y_m$: a random sample from probability distribution G.

Do the two samples arise from a same population?

Test the hypothesis $H_0: F = G = F'(say)$.
without making any assumptions for the distribution of F and G.
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
```

THANK YOU