Applied Probability and Statistics Laboratory Manual

For Kerala Technological university MCA course

Dr.Sunil Thomas Thonikuzhiyil

August 30, 2016

Dedicated to Our Supreme Lord DINKAN.


ചിങ്കൻ

Contents

Preface

I was a member of syllabus revision committee for Master of Computer Applications Course offered by Kerala Technological university during 2016. During the committee meeting Prof Jyothi John (Principal College of Engineering Chengannur) Suggested inclusion of applied statistics and probability theory as a core paper. It was also decided that a supporting lab course be also offered. I was entrusted with formulating the syllabus.

For theory classes there are several standard text books available. For laboratory classes there was a little bit of confusion. There are several options avoidable. Mayy commercial statistical analysis tools are available in the market. However, most of them are proprietary and costly. I suggested the use of R for the class. However several members felt that there was no good laboratory manuals teaching the basics. Students doing MCA come from a variety of backgrounds such as science and commerce.

I promised to write some introductory material so that students and faculty can use it for their lab practice. The material presented is an early draft and should be used with caution. It needs lot of editing and correction. This manual uses ideas and code taken from many sources. I will try to attribute credits as far as possible in the final version. Since the course has started in several affiliated colleges, there was some urgenty in releasing the material. Hence, several fine points may be missing. Any errors found in the draft may be sent to me via email at vu2swx at gmail dot com It is suggested that students use the R studio environment rather than vanilla R console. The R studio is user friendly and intuitive.

1

Visualizing Data

What we will learn

In this experiment we will try to learn the preliminaries of R along with methods of visualizing data in R

Tables, charts and plots. Visualizing Measures of Central Tendency, Variation, and Shape. Box plots, Pareto diagrams. How to find the mean median standard deviation and quantiles of a set of observations.

1.1 Introduction to R

To be written

1.1.1 Installing R

Platforms Window and linux

Installing R on a Windows PC

To install R on your Windows computer, follow these steps:

1. Go to http://ftp.heanet.ie/mirrors/cran.r-project.org.
2. Under "Download and Install R", click on the "Windows" link.
3. Under "Subdirectories", click on the "base" link.
4. On the next page, you should see a link saying something like "Download R 2.10.1 for Windows" (or R X.X.X, where X.X.X gives the version of R, eg. R 2.11.1). Click on this link.
5. You may be asked if you want to save or run a file "R-2.10.1-win32.exe". Choose "Save" and save the file on the Desktop. Then double-click on the icon for the file to run it.
6. You will be asked what language to install it in - choose English.

7. The R Setup Wizard will appear in a window. Click "Next" at the bottom of the R Setup wizard window.

8. The next page says "Information" at the top. Click "Next" again.

9. The next page says "Information" at the top. Click "Next" again.

10. The next page says "Select Destination Location" at the top. By default, it will suggest to install R in "C:
Program Files" on your computer.

11. Click "Next" at the bottom of the R Setup wizard window.

12. The next page says "Select components" at the top. Click "Next" again.

13. The next page says "Startup options" at the top. Click "Next" again.

14. The next page says "Select start menu folder" at the top. Click "Next" again.

15. The next page says "Select additional tasks" at the top. Click "Next" again.

16. R should now be installed. This will take about a minute. When R has finished, you will see "Completing the R for Windows Setup Wizard" appear. Click "Finish".

17. To start R, you can either follow step 18, or 19:

18. Check if there is an "R" icon on the desktop of the computer that you are using. If so, double-click on the "R" icon to start R. If you cannot find an "R" icon, try step 19 instead.

19. Click on the "Start" button at the bottom left of your computer screen, and then choose "All programs", and start R by selecting "R" (or R X.X.X, where X.X.X gives the version of R, eg. R 2.10.0) from the menu of programs.

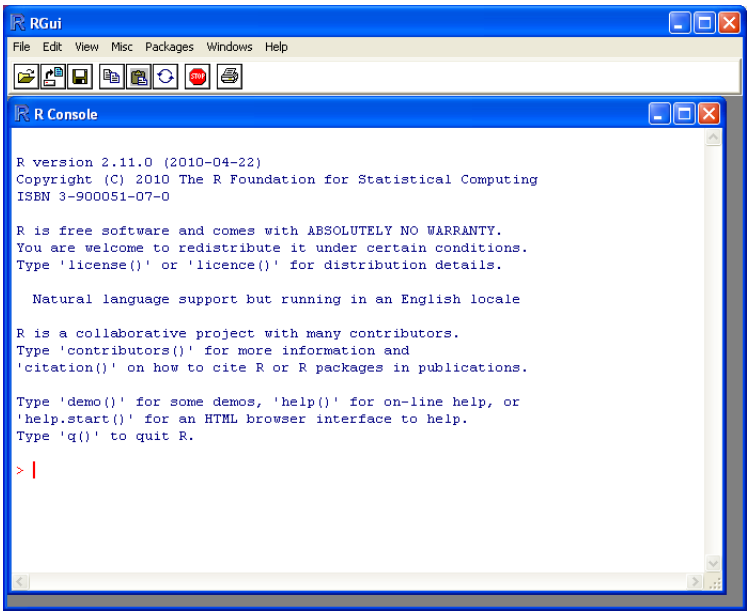
20. The R console (a rectangle) should pop up:

1.1.2 Installing R studio

1. Go to RStudio IDE download page.

2. Click Download RStudio Desktop.

3. Then click for the download link recommended for your system.



4. Run the downloaded file (double click the file) to start the setup wizard.

5. Click “Next” until “Finish”

1.2 R fundamentals

This is a review of R fundamentals. No details are covered. It is assumed that the participant is familiar with programming.

1.2.1 Basic Math operations in R

Type the following on > prompt and see the results.

```
1 + 1
5 - 3
3 * 3
4 / 5
4 ^ 2  # Exponetiation
5 %% 2  # Modulus
5 %/% 2 # integer division
```

1.2.2 Variables and assignment in R

There is no need to declare variables in advance. Rules for variable names are almost similar to those in C programming language. There are two

assignment : + and <-. The preferred assignment operator is <- . Try the following.

```
x=20  # assign x=20
y<-3  # assign y =3
x     # Display x
y     # Display y
x=y
x
y<-x*10
```

A variable can be removed using rm() function.

```
rm()  # remove x
x     # try printing x after removal
```

1.2.3 R Data Types

R has a wide variety of data types including scalars, vectors (numerical, character, logical), matrices, data frames, and lists. Let us quickly review them.

```
# An integer can be assigned to a variable by suffixing L
x=5L
class(x)  # This prints the data type of x

# numeric
y=5.3
class(y)

# Character string
y=" hello"
class(y)

# date
date=as.Date("2016-10-16")
date
```

1.2.4 Logical operators

Similar to C > < <= >= == !=

```
5>6
```

1.2.5 Control statements

if-else

```
 x <- 5
if(x > 0){
   print("Positive number")
}
```

ifelse

```
 a = c(5,7,2,9) # a is a vector. The function c creates vector a
 a   #print a
 ifelse(a %% 2 == 0,"even","odd")
```

for loops

```
 x <- c(2,5,3,9,8,11,6) # x is a vactor
count <- 0
for (val in x) {
    if(val %% 2 == 0) count = count+1
}
print(count)
```

while loops

```
 i <- 1

while (i < 6) {
   print(i)
   i = i+1
}
```

1.3 R data structures

Here we will review and learn basic data structures commonly used in R.

1.3.1 Vector

A fundamental R data structure is the vector, which stores an ordered set of values called elements. A vector can contain any number of elements.

However, all the elements must be of the same type; for instance, a vector cannot contain both numbers and text.

You can build a vector as below using the c() function. c functions combines several entities of to a vector.

```
state<- c("Kerala","Tamil Nadu"," Maharashtra")
```

A vector can be printed by typing its name at the R prompt.

```
state
```

Let us create several vectors.

```
temperature <- c(98.1, 98.6, 101.4)
capital <- c("tvm","chennai","mumbai")
drought <- c(FALSE, FALSE, TRUE)
```

Let us print them.

```
state
temperature
capital
drought
```

You can access individual elements using square brackets.

```
temperature[2]
```

You can create a vector containing a sequence of numbers using :

```
x=10:50
x
x[3]
y=-2:5
z=10:1
y
z
```

Vector Operations

```
x*3

x/4
x+2
sqrt(x)
y=1/x
```

```
y
x=1:10
y=10:1
x+y
x-y
x*y
x/y
length(x)
m=c(1,2)
x
x+m
x<5
```

1.3.2 Factors

Factor is a data structure used for fields that takes only predefined, finite number of values (categorical data). For example, a data field such as marital status may contain only values from single, married, separated, divorced, or widowed. In such case, we know the possible values beforehand and these predefined, distinct values are called levels

```
status <- factor(c("single","married","married","single"));
status
```

1.3.3 Lists

Let us create and print a list.

```
n = c(2, 3, 5)
s = c("aa", "bb", "cc", "dd", "ee")
b = c(TRUE, FALSE, TRUE, FALSE, FALSE)
x = list(n, s, b, 3) # x contains copies of n, s, b

x
```

Display list elements.

```
x[2]
x[[2]]
x[[2]][1]
```

1.3.4 Data Frames

A data frame is used for storing data tables. It is a list of vectors of equal length. For example, the following variable df is a data frame containing three vectors n, s, b.

```
n = c(2, 3, 5)
s = c("aa", "bb", "cc")
b = c(TRUE, FALSE, TRUE)
df = data.frame(n, s, b)

# print the data frame df
df
```

Accessing rows and columns of data frame.

```
df[1]
df[[1]]
df$n
df[2,2]
```

Examining some built in data frames.

There are several built in data frames you can use. mtcars is a simple frame about car parameters.

```
mtcars
help(mtcars) # read the documentation.
```

Let us experiment with this data set.

```
mtcars[1, 2]

mtcars["Mazda RX4", "cyl"]
mtcars["Mazda RX4",]
nrow(mtcars)

ncol(mtcars)
head(mtcars)
mtcars[[9]]

mtcars[["am"]]
mtcars$am
mtcars[1]

mtcars["mpg"]
mtcars[c("mpg", "hp")]
mtcars[24,]

mtcars[c(3, 24),]
mtcars["Camaro Z28",]
mtcars[c("Datsun 710", "Camaro Z28"),]
```

1.3.5 Exercises

Examine iris data set.

1.3.6 Importing Data

Download the iris data set from . Read the background information from wikipedia.

Data can be stored in a comma separated value format Common extension is .csv. You can read a csv file as below. It is assumed that the file is in your current directory. If you get an error here most probably you are in wrong directory or the data file is missing. Please download the data file and place it in your current directory.

```
mydata = read.csv("iris.csv") # read csv file. Make sure that you
    have the csv file
```

If you get an error, please try out the following commands.

```
getwd() # This command prints your current directory

setwd("/home//sunil") # use this commnad to set the direcotry.


mydata = read.csv("iris.csv") # read csv file

mydata
str(mydata ) # display the structure of mydata variable
```

1.3.7 Find out statistics of your data.

```
 mean(mtcars$mpg)
median(mtcars$mpg)
sd(mtcars$mpg)
summary(mtcars)



range(mtcars$mpg)
diff(range(mtcars$mpg))


IQR(mtcars$mpg) # I am not explaining this. Find out from
    documentation.:D
```

Repeat the above for iris data set.

1.4 Plotting in R

1.4.1 Box Plots

Boxplots are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them.

```
boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders",
   ylab = "Miles Per Gallon", main = "Mileage Data")
```

1.4.2 Histograms

```
 # Create data for the graph.
v <- c(9,13,21,8,36,22,12,41,31,33,19)
hist(v,xlab = "Weight",col = "yellow",border = "blue")


hist(mtcars$mpg, main = "Histogram of miles per gallon",
xlab = "mpg")
```

1.4.3 Line graph

```
 v <- c(7,12,28,3,41)
plot(v,type = "o")



# multiple lines
v <- c(7,12,28,3,41)
t <- c(14,7,6,19,3)
plot(v,type = "o",col = "red", xlab = "Month", ylab = "Rain fall",
   main = "Rain fall chart")
lines(t, type = "o", col = "blue")


\end{lstlisting
\subsection{Scatter plots}
```

```
\begin{lstlisting}[language=R]
 plot(x = mtcars$mpg,y = mtcars$cyl)
 help (plot )
 plot(x = mtcars$mpg, y = mtcars$hp,
main = "Scatterplot of mpg vs. hp",
xlab = "mpg",
ylab = "hp")


pairs(~wt+mpg+disp+cyl,data = mtcars,
   main = "Scatterplot Matrix")
```

2

Probability Distributions

2.1 Background Information

To be completed. Describe Basic ideas about probability, distribution function, density, Expectation and variance.

2.2 Simulating Coin Toss

The probability of getting a Heads or a Tails on a coin toss is both 0.5. We can use R to simulate an experiment of flipping a coin a number of times and compare our results with the theoretical probability.

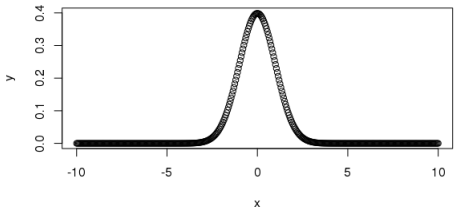
0 = Tails and 1 = Heads

```
sample(0:1,15,rep=T) #flip 15 times

# read the manual for sample function and try to under stand the
    options.
help(sample)
```

Let us write a small function so that we can easily repeat coin toss experiment and calculate probability of getting 1.

```
flip= function(n) sample(0:1,n,rep=T) # look at the function
    definition and the argument passing.
exp1<-flip (100)  # let us flip 100 times
prob= sum(exp1==1)/100

exp2= flip(10000); # a  semi colon suppresses long outputs
prob= sum(exp2==1)/10000
```

Rolling a dice with six faces 20 times can be simulated in R using sample(1:6,20,rep=T). Write a function to simulate roll of a dice and roll it



10000 time. Compute the probability of getting 4 any number less than 4.

2.3 Exploring Probability distributions

In this section we will try to explore various probability distributions. R supports many of them. For each distribution for functions are provided pdf cdf inverse cdf and a random number generator. The density functions start with d. Eg dnorm() dbinom are densty functions for normal and binomial distributions. The cdf starts with p inverse with q and random number generator with r. Eg rbinom() will give a sample from a binomial distribution.

2.4 Normal distribution

Gaussian probability distribution is the most important probability distribution we will study.

$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$ There are two parameters μ, the mean and σ ,the standard deviation. Read the documentation of normal distribution.

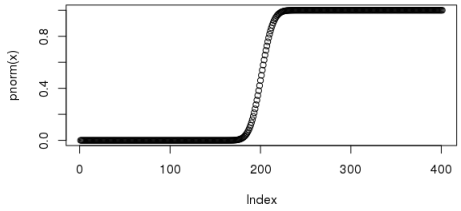
```
help(Normal)
```

There are four function. dnorm pnorm qnorm and rnorm. Let us experiment with each of them. Let us examine dnorm. It give the probability density at a point. For example dnorm(0) will return 0.3989423.

```
x=seq(-10, 10, by = 0.05)   # create a vector of numbers between
    -10 ad 10
y=dnorm(x)                  # find probability density at each
    point in the above vector
plot(x,y)                   #plot
```

Your out put will look like figure 1

The above code assumes that you are using a normal distribution of mean=1 and sd=1. You can change both parameters.




```
dnorm(0,mean=4,sd=10) # density at 0 from a distribution with mena
    4 and sd=10
```

Try plotting the distribution for different values of mean and standard deviation. Observer how the curve looks like.

Pnorm function give the cumulative distribution.

```
pnorm(0)  #cdf at 0

#plot cdf from -10 to 10
 x <- seq(-20,20,by=.1)
 plot(pnorm(x))
```

Here also you can change mean and standard deviation. If you wish to find the probability that a number is larger than the given number you can use the lower.tail option. eg. pnorm(0,mean=2,lower.tail=FALSE)

The next function we look at is qnorm which is the inverse of pnorm. The idea behind qnorm is that you give it a probability, and it returns the number whose cumulative distribution matches the probability.

```
x <- seq(0,1,by=.05)
y <- qnorm(x)
 plot(x,y
```

The last function rnorm returns a random number whose probability distribution is normal.

```
rnorm(4) # generate 4 random nos. from normal distribution.
rnorm(4,mean=3,sd=3) # generate 4 random nos. from normal
    distribution with mean 3 and sd 3
```

Let us examine the histogram of a series of random numbers picked from a normal distribution.

```
y=rnorm(2000) # generate 2000 random nos. from normal
    distribution.
hist(y)
```

Try increasing the size of y above and observe how the histogram tends to a normal distribution.

2.5 Binomial Distribution

Read the documentation of binomial distribution.

```
help(Binomial)
```

The binomial distribution requires two extra parameters, the number of trials and the probability of success for a single trial. The commands follow the same kind of naming convention, and the names of the commands are dbinom, pbinom, qbinom, and rbinom.

```
x <- seq(0,50,by=1)
y <- dbinom(x,50,0.2) # 50 trials with probability of success 0.2
plot(x,y)
y <- dbinom(x,50,0.6)
plot(x,y)
x <- seq(0,100,by=1)
y <- dbinom(x,100,0.6)
plot(x,y)
```

2.6 Poisson distribution

Read the documentation of Poisson distribution

```
help("Poisson")
```

```
dpois()  to be completed as above
```

3

Random Samples

3.1 Introduction

We will try to under stand random sampling and the importance of central limit theorem.

3.2 Random number generators in R

R provide several functions to generate random numbers drawn from standard distributions. For using them, you only need know the parameters that are given to the functions such as a mean, or a rate. Several examples are given below. For each, a histogram is given for a random sample of size 100, and density.

Uniform distribution

The random number generator using uniform distribution tries to draw a number from a range $[a, b]$ using uniform probability. Let us first get familiar with uniform distribution functions.

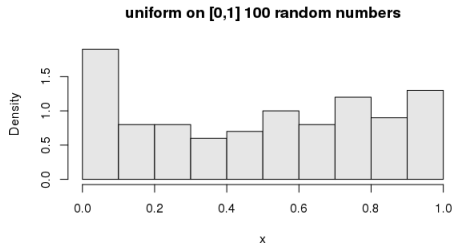
```
help("Uniform")
```

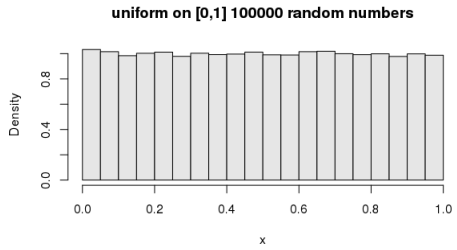
The random number generation function is runif. Let us generate some random numbers.

```
runif(1,0,10) # generate one random number between 0 and 10.
runif(10,0,10) # generate 10 random numbers between 0 and 10.
runif(5)      # this is generate 5 random numbers between 0 and 1
```

Let us plot the histogram of 100 random numbers generated from a uniform distribution.

```
x=runif(100)
```






```
hist(x,probability=TRUE,col=gray(.9),main="uniform on [0,1] 100
    random numbers"
# Read the documentation of hist function to understand the
    arguments
```

We got some the above plot as in fig hist . It is rather a crude approximation to uniform. Let us increase the number of random samples to 100000.

```
x=runif(100000)
hist(x,probability=TRUE,col=gray(.9),main="uniform on [0,1] 100000
    random numbers"
# Read the documentation of hist function to understand the
    arguments
```

The plot is now looking like the one in fig hist10000

It may be noted that the plots you get may look slightly different as we are generating random numbers. If you want to plot density function, use dunif to along with curve function.

```
x=runif(100)
 curve(dunif(x,0,1)) # read documentation of curve.
```

3.3 Central limit theorem

The central limit theorem is probably the most important theorem in statistics. let $x_1, x_2 \cdots x_n$ be random samples from any distribution with mean μ and variance σ^2 , central limit theorem states that for large sample size n the sampling distribution of sample mean $\hat{x} = \frac{1}{n}\sum i = 1^n X_i$ is approximately normal with mean σ and variance σ^2/n

3.4 Central limit theorem in R

The binomial distribution with parameters n and p is the discrete probability distribution of the number of successes in a sequence of n independent yes/no experiments, each of which yields success with probability p. Consider a binomial distribution with N=20 p=0.25. We do 25 trial. Each trial has a probability of success 0.25. We can generate random numbers distributed using this binomial distribution using rbinom(1,n.p). This function returns the number of successes in n trials. The first parameter is the number of random numbers to be generated. eg. rbinom(10,n,p) will generate 10 such random numbers. The mean of binomial distribution is np and the variance is np(1-p).

Let us draw 10 samples from the above distribution and find out the average. We demote it as one experiment.

```
n=25
p=0.25;

X=rbinom(10,n,p)
```

Let us compute the average of these samples

```
Y=sum(X)/10
```

The average of the distribution is 6.25 (np). Compare it Y. It will be some where near 6.25. Now let us do the above operation 10 times and compute average in each case. Here we are sampling 10 numbers from a binomial distribution 10 times and finding out how the averages behave. We are doing 10 experiments.

```
Y=matrix( rbinom(100,n,p ), 10, 10) ) # Y is a 10 by 10 matrix.
    Each row contains 10 numbers sampled from the binomial
    distribution.
Y  # Examine Y

Z= rowSums(Y) /10 # Z contains 10 averages

hist(Z)   # Find out how Z is distributed.
```

The central limit theorem says that the histogram of the above averages will approach normal distribution as we do more and more experiments. Let us do 100 such experiments.

```
Y=matrix( rbinom(1000,n,p ), 100, 10) ) # Y is a 100 by 10 matrix.
    Each row contains 10 numbers sampled from the binomial
    distribution.
Y  # Examine Y

Z= rowSums(Y) /10 # Z contains 100 averages

hist(Z)   # Find out how Z is distributed.
```

Repeat it for 1000 experiments

```
Y=matrix( rbinom(10000,n,p ), 1000, 10) ) # Y is a 1000 by 10
    matrix. Each row contains 10 numbers sampled from the binomial
    distribution.
Y  # Examine Y

Z= rowSums(Y) /10 # Z contains 1000 averages

hist(Z)   # Find out how Z is distributed.
```

You can see that the histogram approaches normal

4

Binomial Distribution and Central Limit theorem

5

Confidence Intervals

6

Correlation

7

Regression