STELLENBOSCH UNIVERSITY

R COURSE

19200129
THAMU MNYULWA

Contents

INTRODUCTION TO R: 2019

FINAL ASSIGNMENT: PARTS B and C

General

1. First read Section 6.1 carefully.
2. The tasks given below form the final assignment to be handed in (electronic format).
3. Answers as well as the R code to produce your answers must be handed in.
4. Note: Some tasks involve writing R code to produce answers that must be interpreted and conclusions given in the form of a short final report.

PART B

<u>Compulsory exercises for FRM students and all students registered for the full R course</u>

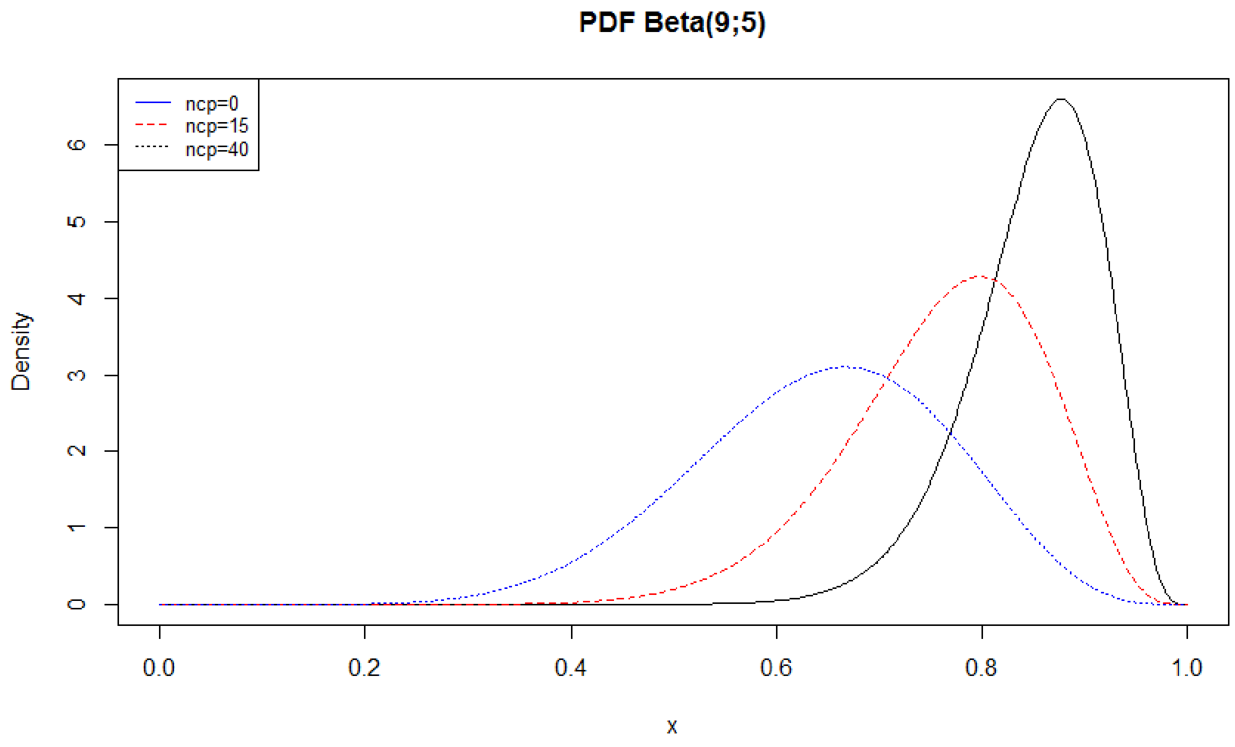
Exercise 4.5.2.

```
> fix(Ex.4.5.2)
```

```
#function to print the beta distribution on the same axis at different
#non central parameters (ncp)
#Function makes use of low level and high level plotting techniques in
#order to plot multiple plots on the same graph starting with the high
#level plot function and plotting other lines with the lines function.
# lty arg is changed as well as col in order to provide easier
#visualization ; start with ncp=40 , to allow fit
#function takes in no arguments
#Description of ncp and low level and high level plotting below

function()
{
# plot line ncp=40
plot(x=seq(0,1,length=500),y=dbeta(seq(0,1,length=500),shape1=9,shape2=5,
ncp=40)
,type='l',main="PDF Beta(9;5)", lty=1, xlab ="x", ylab = "Density")

# plot line ncp=15
lines(x=seq(0,1,length=500),y=dbeta(seq(0,1,length=500),shape1=9,shape2=5
,ncp=15),type='l',col="red", lty=2)

# plot line ncp=0
lines(x=seq(0,1,length=500),y=dbeta(seq(0,1,length=500),shape1=9,shape2=5
,ncp=0),type='l',col="blue", lty=3)

# Add a legend to plot
legend("topleft", legend = c("ncp=0 ", "ncp=15","ncp=40"), col=c("blue",
"red","black"), lty=1:3, cex=0.8)
}
```

```
> Ex.4.5.2()
```

Figure 1 : non centrality parameter



Interpretation:
The non centrality parameter (ncp) is a shape parameter to the beta distribution, from observation one could see how as the ncp rises from 0 to 40 the probability density function becomes more skew to the left and the highest point on the graph moves up from a density of three to a density of over 6 on the y-axis. This is consistent with the expectations of the higher non centrality parameter.

Densities on the same axis plot(), lines(), are necessary to change limits so there is enough space after high level data points
Note use of High level then low level plotting after.

Exercise 4.5.3.

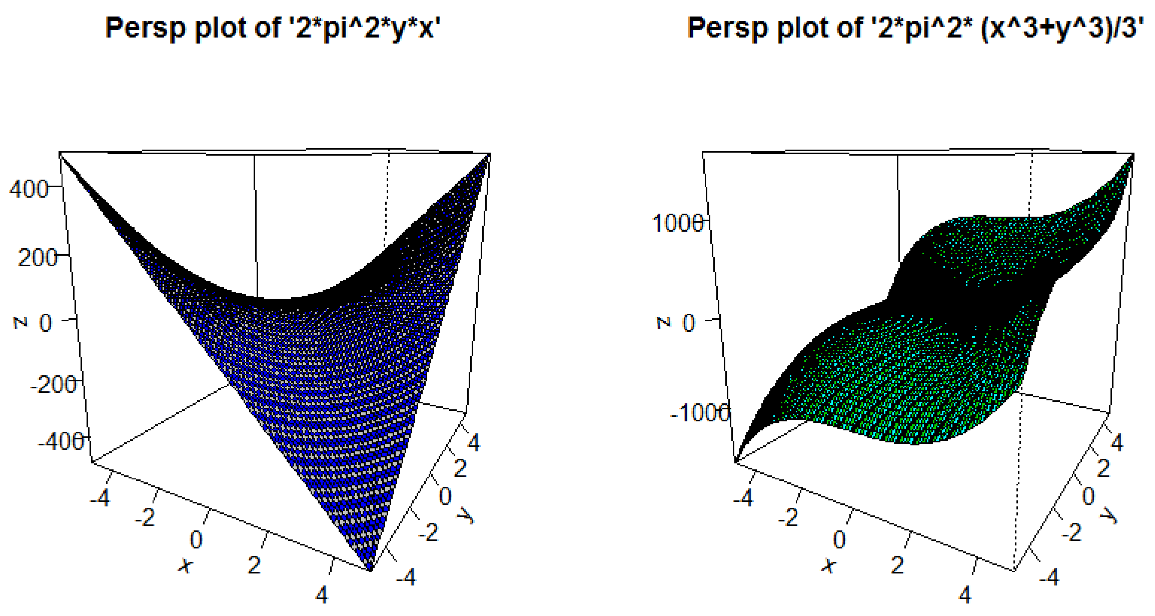
```
>fix(Ex.4.5.3)
```

```
function (begin =-5,end=5,length=100,theta=30,r=sqrt(3),
          phi=15, ticktype='detailed',
          fun="2*pi^2* sqrt((x^3+y^3)/3)",...)
{
   #predefined function "2*pi^2* sqrt((x^3+y^3)/3)" can be changed
   #User can change view direction theta and phi
   #User can change any argument in persp function.

# create a sequence
    pts <- seq(begin,end,len=length)

# paste combined the text then parse changes the string into a expression
# to be evaluated by eval

    fun2 <- parse(text = paste("function(x,y)",fun))

  #Explain carefully what parse is doing?
  #parse is changing the string into an expression
  #outcome of parse is not evaluated until run through eval to evaluate
  #you get the expression coming from parse and evaluate it with eval

# outer(X,Y, FUN = "*",...) :
# outer takes pts as x and y and uses eval to evaluate the fun2
# expression and uses it as a function to create z

    z <- outer(pts, pts, eval(fun2))

#Explain carefully what eval() is doing?
#eval takes the expression (output from parse in this case) and
#evaluates it


    persp(x=pts, y=pts, z ,theta=theta, phi=phi, r=r, ticktype=ticktype,
          ,xlab="x",ylab="y",zlab="z",...)

#Role of paste?
#paste combines all the strings and expressions using the ‘ ’ as a
#separator, note the “’” to add a parenthesis into the plot, these, need
#to be different for paste to recognize you are trying to add a
#parenthesis to the plot.

#put title on plot using title
    title(main=paste("Persp plot of '",fun, "'",sep=''))

}
```

OWN SPECIFIED BIVARIATE FUNCTION.

```
> par(mfrow=c(1,2))
> Ex.4.5.2(fun="2*pi^2*y*x",col=c(4,4,8,4,8))
> Ex.4.5.2(fun="2*pi^2* (x^3+y^3)/3",col=c(1,3,5,3,5))
```

Figure 2 : bivariate function



A bivariate function shows the relationship between two variables. X and Y in the case of our example. Our function above takes in the functions and returns a plot.

parse() *- takes the character text and converts it to an expression.*
eval() *- evaluates the converted expression.*
paste() *- converts its arguments to a character string and concatenates the result to the title of the plot.*

Section 5.2 (e) and (f). Carefully select your own input matrices to illustrate the usage of your functions.

Section 5.2 (e)

- Important Note : We know that a matrix is stored internally as a column vector that means that it is stored column wise.

One-index matrix test.mat[10]
Two-index matrix test.mat[2,3]

```
> fix(Ex.5.2.e)
```

```
function (mat,one.index)
{
#function turns a one-index reference to a to index reference for a
matrix
#mat is the single matrix that is to be re-indexed
#'one.index' is the one-index matrix reference that is given for 'mat'
#It begins by taking the row 'n.row' and column's of mat 'p.col'
#Restriction 1: one.index must be greater than 1
#Restriction 2: one.index must be less than product of row number and col
number
# Restriction 3: one.index must be integer
#Condition 1: Since matrix is stored internally in r as a column wise
vector, i.e.
#by row, If one.index is less than the number of rows it is in column 1 ,
row = one.index
#Condition 2: if matrix is not in last row of the mat, ie one.index
mod(%%) number of rows is not zero, rows = reminder.mod and col
=integer.portion +1
#Condition 3: if remainder.mod = 0, we are in the last row of the matrix,
row= number of rows (nrow) and col = integer portion (i.e one.index %/%
nrow)
n.row<-nrow(mat)
p.column<-ncol(mat)
#Restrictions
if( !one.index%%floor(one.index) == 0 ) stop( "Agument 'one.index' must
be an integer." )
if(one.index < 1) stop("Warning: One.index must be greater than 1")
if(one.index > p.column*n.row) stop("Warning: One.index must be less than
product of row number and col number")

if(one.index <= n.row) return( c(row=one.index , col=1) )
else
#"%/%" integer division, divideds then rounds up/down to nearest integer
#"%%" modulus, gives remainder after division of one number by another
#integer.portion, is column if in last row position of mat
#otherwise the column = integer.portion +1
#remainder.mod, is the row if not in last position
# otherwise the row is n.row as we are in last position
integer.portion <- one.index %/% n.row  # 5%/%2 is 2 , column /column-1
remainder.mod <- one.index %% n.row # 5%%2 is 1, row

if( remainder.mod != 0 ) return ( c(row=remainder.mod , col=
(integer.portion+1) ) ) #integer.portion “plus” 1,
else( return( c(row=n.row , col=integer.portion)) )

}
```

Test

```
> mat<-matrix(1:9,ncol = 3,byrow=TRUE)
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9

> Ex.5.2.e(mat,1)
row col
  1   1
> Ex.5.2.e(mat,2)
row col
  2   1
> Ex.5.2.e(mat,3)
row col
  3   1
> Ex.5.2.e(mat,4)
row col
  1   2
> Ex.5.2.e(mat,9)
row col
  3   3
> Ex.5.2.e(mat,6)
row col
  3   2
> Ex.5.2.e(mat,-1)
Error in Ex.5.2.e(mat, -1) : Warning: One.index must be greater than 1

> Ex.5.2.e(mat,0)
Error in Ex.5.2.e(mat, 0) : Warning: One.index must be greater than 1

> Ex.5.2.e(mat,3.455)
Error in Ex.5.2.e(mat, 3.455) : Agument 'one.index' must be an integer.
```

Section 5.2(f)

convert two-index matrix to a one-index matrix reference

```
> fix(Ex.5.2.e)
```

```
#function converts a two column index to a one column index
 #mat is the single matrix that is to be re-indexed
 #It begins by taking the user inputed row 'row' and column of mat 'col'
 #We then calculate mat's rows and columns, row.mat and col.mat
 # We then take the matrix rows and multiply them wih the inputed columns
and then add the row
 #'one.index' is the one-index matrix reference that we calculate for
'mat'
#Restriction 1: row must be greater than 1
#Restriction 2: column must be greater than 1
#Restriction 3: both row and column must be integers
function (mat,row,col)      {
#Restrictions
if(row < 1) stop("Warning: One.index must be greater than 1")
if(col < 1) stop("Warning: One.index must be greater than 1")
if( !row%%floor(row) == 0 ) stop( "Argument 'row' must be an integer." )
if( !col%%floor(col) == 0 ) stop( "Argument 'col' must be an integer." )
  row.mat <- nrow(mat)
  col.mat <- ncol(mat)

  one.index <- row.mat*(col-1) +row
  # Note order: same as "(row.mat*(col-1) ) + row"

return (cat("One Index from \n row = ",row,"col = ",col,"\n=",
c(index=one.index)))
}
```

```
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> Ex.5.2.e(mat,1,1)
One Index from
 row =  1 col =  1
= 1
> Ex.5.2.e(mat,2,2)
One Index from
 row =  2 col =  2
= 5
> Ex.5.2.e(mat,3,4)
One Index from
 row =  3 col =  4
= 12
> Ex.5.2.e(mat,3,3)
One Index from
 row =  3 col =  3
= 9
> Ex.5.2.e(mat,1.23,1)
Error in Ex.5.2.e(mat, 1.23, 1) : Argument 'row' must be an integer.
> Ex.5.2.e(mat,-1,1)
Error in Ex.5.2.e(mat, -1, 1) : Warning: ‘row’ must be greater than 1
```

Exercise 5.7.5. Pay special attention to (v).

Exercise 5.7.5 .i

```
> mean(LifeCycleSavings[LifeCycleSavings[,"pop15"]/LifeCycleSavings[,"pop
75"]>=10,"sr"])
[1] 8.723529
```

Exercise 5.7.5 .ii

```
> mean(LifeCycleSavings[LifeCycleSavings[,"pop15"]/LifeCycleSavings[,"pop
75"]<10,"sr"])
[1] 11.68437
```

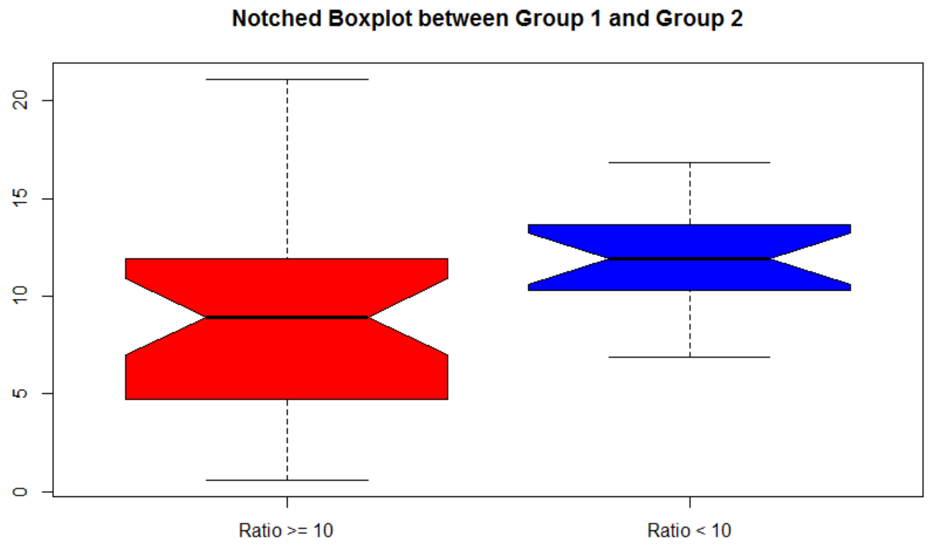
Exercise 5.7.5 .iii

```
># create 2 groups and perform Welch Two Sample t.test
># Interpretation below
>group1 <-
LifeCycleSavings[LifeCycleSavings[,"pop15"]/LifeCycleSavings[,"pop75"]>=1
0,"sr"]
> group2 <-
LifeCycleSavings[LifeCycleSavings[,"pop15"]/LifeCycleSavings[,"pop75"]<10
,"sr"]
> t.test(group1,group2)

       Welch Two Sample t-test

data:  group1 and group2
t = -2.7489, df = 46.198, p-value = 0.008504
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -5.1286999 -0.7929912
sample estimates:
mean of x mean of y
 8.723529 11.684375
```

Exercise 5.7.5 .iv

```
> # notched boxplot of Welch 2 Group Test
> boxplot(group1,group2,main="Notched Boxplot between Group 1 and Group 2
", notch = TRUE, names = c('Ratio >= 10','Ratio < 10'), col = c("red","bl
ue"))
```

Figure 3: Notched Boxplot



Exercise 5.7.5 .v
Welches t-test is a two-sample location test; it is for testing the hypothesis that two populations have equal means. Our Welches t-test rejected the hypothesis that the two samples have equal means at the p-value was below 0.05, we therefore concluded that the two samples means differ (shown below).

The $\text{p} - \text{value} = 0.008504 < \alpha = 0.05$ therefore reject the null hypothesis and therefore the two populations have a significant difference in the means. There is a significant difference in the means of the mean aggregate savings of the countries with the ratio greater or equal to 10 and the ratio smaller than 10. This is true given the variance is the same between the two groups.

IF notch is TRUE, a notch is drawn in each side of the boxes. If the notches of two plots do not overlap this is 'strong evidence' that the two medians differ.

In our example:
We see that the notches do not overlap. This is strong evidence that the two medians between our groups differ. The group with Ration>=10 has a lower median than that of the group with the ratio<10.

We also see that the group in red has a higher maximum and a lower medium than the second group this suggests a wider range in the data and the thinner range between the lower 25th percentile of the blue boxplot and the 75th percentile. This suggests that values in the second group are closer in distribution from the median amount in the data.

We conclude that the variance between the two groups is unequal.

Exercise 5.7.6.

Provide R code to give your final answer in a user friendly table.

```
>
> tapply(state.x77[,"Income"],state.region,function(x){names(x)[x==min(x)]})
     Northeast          South  North Central           West 
       "Maine"  "Mississippi" "South Dakota"   "New Mexico" 

> tapply(state.x77[,"Income"],state.region,function(x){x[x==min(x)]})
# Income of state minimum
    Northeast         South North Central          West 
         3694          3098          4167          3601 
# Better format using format function
> tapply(state.x77[,"Income"],state.region,function(x){format(names(x)[x==min(x)
])})
     Northeast          South  North Central           West 
       "Maine"  "Mississippi" "South Dakota"   "New Mexico" 
```

Exercise 6.2.1.

```
> summary(LifeCycleSavings[,'dpi'])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  88.94  288.21  695.66 1106.76 1795.62 4001.89 

> quantile(LifeCycleSavings$dpi,prob = seq(from = 0, to = 1,length=5))
       0%       25%       50%       75%      100% 
  88.9400  288.2075  695.6650 1795.6225 4001.8900 

> quantile(LifeCycleSavings$dpi,prob = c(0,0.25,0.5,0.75,1))
       0%       25%       50%       75%      100% 
  88.9400  288.2075  695.6650 1795.6225 4001.8900 
> fivenum(LifeCycleSavings$dpi)
[1]   88.940  287.770  695.665 1813.930 4001.890
```

Arguments of Quantile

```
quantile(x, probs = seq(0, 1, 0.25), na.rm = FALSE,
names = TRUE, type = 7, ...)
```

Function fivenum (the last one) differs slightly for the upper quartile and the lower quartile value but still has the same minimum and maximum. The function returns Tukey's five number summary (minimum, lower-hinge, median, upper-hinge, maximum) for the input data. Rather than the sample quantiles corresponding to the given probabilities as do quantile, summary and boxplot functions.

The difference between fivenum() and summary() lies in the lack of agreement on how the 1st and 3rd quartiles should be calculated. When the data sets consists of an odd number of observations it would be the same however our data contains even number of observations hence it is different.

Exercise 6.2.3.

Obtain PDF as well as CDF
In R PDF begins with "d" for density, whilst CDF begins with "q" for quantile.

Exercise 6.2.3.i

Note: PDF and CDF on F distribution, in R we know that
p is for "probability", the **cumulative distribution function (c. d. f.)** and
d for "density", the **density function (p. d. f.)(Probability Density Function)**

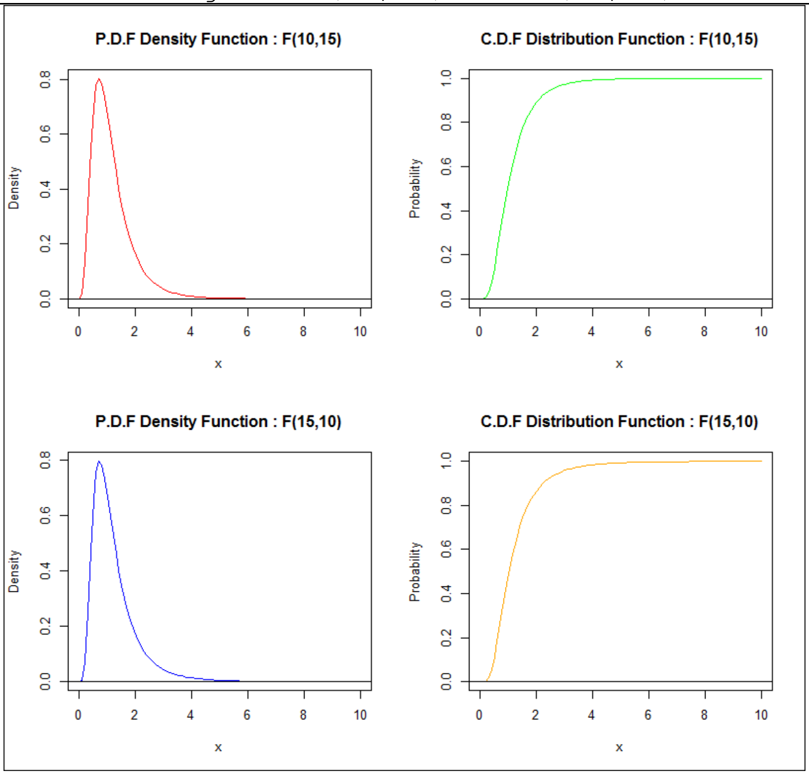
```
> args(pf)
function (q, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE)

> args(df)
function (x, df1, df2, ncp, log = FALSE)
```

```
>fix(Ex.6.2.3.i)
```

```
function ()
{
 par(mfrow=c(2,2))
 ####F(10,15)
 # plot Probability Density Function P.D.F
 pts<-seq(from=0, to=10,length = 100)
 df1<- df(pts,df1=10,df2=15)

 plot(x=pts,y=df1,type="l",main="P.D.F Density Function :
F(10,15)",xlab="x",ylab="Density",col="red")
 abline(h=0)

 # plot Cumulative Distribution Function C.D.F
 pts<-seq(from=0, to=10,length = 100)
 pf2<- pf(pts,df1=10,df2=15)

 plot(x=pts,y=pf2,type="l",main="C.D.F Distribution Function :
F(10,15)",xlab="x",ylab="Probability",col="green")
 abline(h=0)

  ####F(15,10)
  # plot Probability Density Function P.D.F
 pts<-seq(from=0, to=10,length = 100)
 df1<- df(pts,df1=15,df2=10)

 plot(x=pts,y=df1,type="l",main="P.D.F Density Function :
F(15,10)",xlab="x",ylab="Density",col="blue")
 abline(h=0)

 # plot Cumulative Distribution Function C.D.F
 pts<-seq(from=0, to=10,length = 100)
 pf2<- pf(pts,df1=15,df2=10)

 plot(x=pts,y=pf2,type="l",main="C.D.F Distribution Function :
F(15,10)",xlab="x",ylab="Probability",col="orange")
 abline(h=0)
 box("outer")  # add an outer box to see
}
```

```
>Ex.6.2.3.i()
```

Figure 4: Showing the F(15,10) and F(10,15) PDF and CDF



Exercise.6.2.3.ii

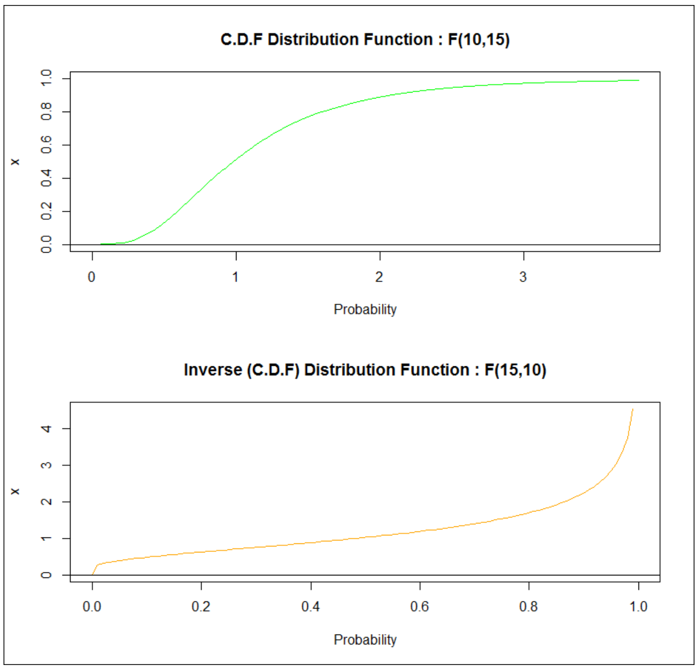
```
>fix(Ex.6.2.3.i)
```

```
function ()
{
par(mfrow=c(2,1))

###Note:For the inverse the pts must be shorter; change the sequence to
less

 ####F(10,15)
 # plot Cumulative Distribution Function C.D.F
 pts<-seq(from=0, to=1,length = 100)
 qf2<- qf(pts,df1=10,df2=15)
 plot(y=pts,x=qf2,type="l",main="C.D.F Distribution Function :
F(10,15)",xlab="Probability",ylab="x",col="green")
abline(h=0) # horizontal line at 0
 ####F(15,10)

 # plot Cumulative Distribution Function C.D.F
 pts<-seq(from=0, to=1,length = 100)
 qf3<- qf(pts,df1=15,df2=10)
 plot(x=pts,y=qf3,type="l",main="Inverse (C.D.F) Distribution Function :
F(15,10)",xlab="Probability",ylab="x",col="orange")
 abline(h=0)# horizontal line at 0
 box("outer")  # add an outer box to see
}
```

```
  [1] (90,100]   (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (100,110]  (100,110]  (100,110]
 [12] (90,100]   (110, Inf] (100,110]  (100,110]  (90,100]   (100,110]  (90,100]   (100,110]  (90,100]   (90,100]   (100,110]
 [23] (90,100]   (100,110]  (100,110]  (90,100]   (100,110]  (90,100]   (90,100]   (90,100]   (90,100]   (90,100]   (90,100]
 [34] (100,110]  (90,100]   (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (90,100]
 [45] (100,110]  (100,110]  (90,100]   (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (90,100]   (100,110]  (100,110]
 [56] (100,110]  (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (90,100]   (100,110]  (90,100]   (90,100]   (100,110]
 [67] (100,110]  (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (90,100]
 [78] (90,100]   (90,100]   (90,100]   (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (90,100]   (100,110]  (100,110]
 [89] (90,100]   (90,100]   (90,100]   (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (100,110]  (100,110]  (100,110]
[100] (100,110]  (100,110]  (90,100]   (100,110]  (100,110]  (90,100]   (100,110]  (90,100]   (90,100]   (100,110]  (75,90]
[111] (90,100]   (90,100]   (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (90,100]   (100,110]  (100,110]
[122] (100,110]  (100,110]  (100,110]  (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (100,110]  (110, Inf] (110, Inf]
[133] (90,100]   (90,100]   (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (90,100]   (100,110]  (100,110]
[144] (90,100]   (100,110]  (100,110]  (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (100,110]  (90,100]
[155] (90,100]   (100,110]  (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (90,100]
[166] (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (90,100]
[177] (100,110]  (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (100,110]
[188] (90,100]   (100,110]  (100,110]  (90,100]   (110, Inf] (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (90,100]
[199] (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (75,90]    (90,100]   (90,100]
[210] (100,110]  (110, Inf] (90,100]   (90,100]   (90,100]   (100,110]  (90,100]   (90,100]   (100,110]  (90,100]   (90,100]
[221] (100,110]  (90,100]   (90,100]   (90,100]   (100,110]  (90,100]   (100,110]  (90,100]   (90,100]   (100,110]  (100,110]
[232] (90,100]   (90,100]   (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (90,100]   (100,110]  (90,100]   (100,110]
[243] (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (90,100]
[254] (100,110]  (90,100]   (100,110]  (90,100]   (100,110]  (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (90,100]
[265] (100,110]  (100,110]  (90,100]   (100,110]  (100,110]  (90,100]   (100,110]  (90,100]   (90,100]   (90,100]   (100,110]
[276] (100,110]  (100,110]  (100,110]  (90,100]   (75,90]    (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (100,110]
[287] (110, Inf] (100,110]  (100,110]  (100,110]  (100,110]  (100,110]  (75,90]    (90,100]   (100,110]  (90,100]   (90,100]
[298] (100,110]  (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (100,110]  (90,100]   (100,110]  (90,100]   (90,100]
[309] (90,100]   (90,100]   (90,100]   (90,100]   (90,100]   (90,100]   (90,100]   (90,100]   (100,110]  (90,100]   (100,110]
[320] (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (100,110]  (90,100]   (90,100]
[331] (90,100]   (100,110]  (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (100,110]  (100,110]  (90,100]
[342] (100,110]  (90,100]   (90,100]   (90,100]   (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (100,110]  (100,110]
[353] (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (90,100]   (90,100]   (100,110]  (100,110]
[364] (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (90,100]   (100,110]  (90,100]
[375] (90,100]   (100,110]  (100,110]  (90,100]   (90,100]   (100,110]  (90,100]   (100,110]  (90,100]   (90,100]   (90,100]
[386] (90,100]   (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (100,110]  (100,110]  (100,110]  (90,100]
[397] (90,100]   (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (90,100]   (90,100]   (100,110]  (90,100]   (100,110]
[408] (100,110]  (90,100]   (90,100]   (100,110]  (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (100,110]  (90,100]
[419] (75,90]    (100,110]  (100,110]  (90,100]   (100,110]  (90,100]   (100,110]  (100,110]  (90,100]   (100,110]  (90,100]
[430] (100,110]  (100,110]  (100,110]  (90,100]   (100,110]  (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (90,100]
[441] (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (100,110]  (100,110]
[452] (90,100]   (90,100]   (90,100]   (90,100]   (100,110]  (100,110]  (100,110]  (90,100]   (100,110]  (100,110]  (90,100]
[463] (100,110]  (100,110]  (100,110]  (100,110]  (100,110]  (100,110]  (90,100]   (100,110]  (90,100]   (90,100]   (90,100]
[474] (100,110]  (90,100]   (90,100]   (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (100,110]  (100,110]  (90,100]
[485] (90,100]   (100,110]  (100,110]  (90,100]   (90,100]   (90,100]   (90,100]   (100,110]  (90,100]   (90,100]   (100,110]
[496] (90,100]   (100,110]  (100,110]  (90,100]   (100,110]
Levels: [-Inf,50] (50,75] (75,90] (90,100] (100,110] (110, Inf]
```

```
>Ex.6.2.3.i()
```

<u>Figure 5: Showing the F(15,10) and F(10,15) CDF</u>



Exercise 6.2.4.

*We show both args of cuts to illustrate the difference.

```
> set.seed(172389)
> sample <- rnorm(500, m=100, sd=sqrt(20))
> cut(sample,breaks = c(-Inf,50,75,90,100,110,Inf),include.lowest = TRUE)
```

```
> # add labels and create groups
> show<-cut(sample,breaks = c(-Inf,50,75,90,100,110,Inf),labels =
c("<50","50 to 75","75 to 90","90 to 100","100 to 110","110
<"),include.lowest = TRUE)
```

```
> # Use "show" to create a table counting frequency in each group
> table(show)

       <50   50 to 75   75 to 90  90 to 100 100 to 110      110 <
         0          0          5        231        258          6
># Create table with no predefined names to show in default form
> show<-cut(sample,breaks = c(-Inf,50,
75,90,100,110,Inf),include.lowest = TRUE)

># Use "show" to create a table counting frequency in each group
> table(show)
show
 [-Inf,50]    (50,75]    (75,90]   (90,100]  (100,110] (110, Inf]
         0          0          5        231        258          6
```

Exercise 6.2.7.

Make sure to give a complete answer.

Exercise 6.2.7 (i)

```
## takes "company.10var" in as a list, then assign it as data.frame
> company.10var <- source("clipboard")
> company.10var <- as.data.frame(company.10var[[1]])
> head(company.10var,2)
  Successful   X1   X2   X3   X4   X5   X6   X7   X8   X9  X10
1          1 0.09 0.13 0.03 0.25 0.25 0.04 0.03 1.61 0.74 1.38
2          1 0.04 0.07 0.02 0.06 0.04 0.02 0.01 1.39 0.85 1.98

> tail(company.10var,2)
   Successful   X1   X2   X3   X4   X5   X6   X7   X8   X9    X10
78          2 0.15 0.27 0.16 0.22 0.37 0.58 0.27 2.84 1.75 112.38
79          2 0.13 0.27 0.14 0.30 0.55 0.28 0.26 1.67 0.73   4.46
```

Exercise 6.2.7 (ii)

```
#Notice that column Successful has 1 for successful, 2 for unsuccessful
#subscript "Successful" column into "company.10var.faliure"
#&"company.10var.success"
# We hard code the commands to illustrate them and then compile a function to
# allow the process to be carried out on multiple datasets.
```

```
> company.10var.success <- company.10var[company.10var[,'Successful']==1, ]
> head(company.10var.success,2)
  Successful   X1   X2   X3   X4   X5   X6   X7   X8   X9  X10
1          1 0.09 0.13 0.03 0.25 0.25 0.04 0.03 1.61 0.74 1.38
2          1 0.04 0.07 0.02 0.06 0.04 0.02 0.01 1.39 0.85 1.98

> company.10var.faliure <- company.10var[company.10var[,'Successful']==2, ]
> head(company.10var.faliure,2)
   Successful   X1   X2   X3   X4   X5   X6   X7   X8   X9  X10
25          2 0.17 0.18 0.11 0.24 0.34 0.19 0.15 2.36 1.26 5.20
26          2 0.13 0.18 0.09 0.19 0.29 0.18 0.14 1.85 1.13 8.73
```

```
# remove “Successful” column and make covariance matrices for subscripts
# changed data.frame into matrix for future use
# rounded for publication purposes
> cov.success<-var(as.matrix(company.10var.success[,-1]))
> cov.faliure <-var(as.matrix(company.10var.faliure[,-1]))
```

```
> round(cov.success,3)
       X1    X2    X3     X4     X5    X6    X7     X8     X9     X10
X1  0.018 0.009 0.009  0.115  0.116 0.012 0.010  0.022  0.035   0.495
X2  0.009 0.011 0.009  0.081  0.082 0.012 0.010  0.018  0.014   0.522
X3  0.009 0.009 0.009  0.105  0.106 0.011 0.010  0.022  0.020   0.340
X4  0.115 0.081 0.105 13.509 13.451 0.105 0.108 -0.208 -0.018   1.413
X5  0.116 0.082 0.106 13.451 13.396 0.106 0.110 -0.205 -0.016   2.008
X6  0.012 0.012 0.011  0.105  0.106 0.015 0.012  0.023  0.023   0.713
X7  0.010 0.010 0.010  0.108  0.110 0.012 0.010  0.023  0.021   0.488
X8  0.022 0.018 0.022 -0.208 -0.205 0.023 0.023  0.232  0.199   4.939
X9  0.035 0.014 0.020 -0.018 -0.016 0.023 0.021  0.199  0.216   5.750
X10 0.495 0.522 0.340  1.413  2.008 0.713 0.488  4.939  5.750 694.850
> round(cov.faliure,3)
        X1     X2     X3     X4     X5    X6     X7     X8     X9      X10
X1   0.004  0.003  0.002  0.003  0.005 0.003  0.003 -0.008  0.002    0.298
X2   0.003  0.005  0.003  0.005  0.009 0.005  0.005 -0.014 -0.005    0.841
X3   0.002  0.003  0.002  0.003  0.004 0.004  0.003 -0.005 -0.002    0.824
X4   0.003  0.005  0.003  0.008  0.013 0.002  0.005 -0.037 -0.021    0.053
X5   0.005  0.009  0.004  0.013  0.025 0.001  0.009 -0.057 -0.029    0.176
X6   0.003  0.005  0.004  0.002  0.001 0.026  0.007  0.117  0.055    6.275
X7   0.003  0.005  0.003  0.005  0.009 0.007  0.006 -0.008 -0.002    1.439
X8  -0.008 -0.014 -0.005 -0.037 -0.057 0.117 -0.008  1.462  0.661   30.686
X9   0.002 -0.005 -0.002 -0.021 -0.029 0.055 -0.002  0.661  0.380   16.472
X10  0.298  0.841  0.824  0.053  0.176 6.275  1.439 30.686 16.472 3919.422
```

```
> company.10var.faliure<-company.10var.faliure[,-1]

> company.10var.success<-company.10var.success[,-1]
```

```
> mean.vec.success <-apply(company.10var.success,2,mean)

> mean.vec.faliure <-apply(company.10var.faliure,2,mean)
> round(mean.vec.success,4)
     X1      X2      X3      X4      X5      X6      X7      X8      X9     X10 
 0.0258  0.0042 -0.0521 -1.0142 -1.0037 -0.0600 -0.0492  1.3754  0.7892 12.6071 
> round(mean.vec.faliure,4)
     X1      X2      X3      X4      X5      X6      X7      X8      X9     X10 
 0.1204  0.1915  0.1033  0.2313  0.3751  0.2298  0.1656  1.9411  1.0162 40.3665 
```

Exercise 6.2.7 (iv)

LDA

#Linear function of two variables that optimally separate 2 groups

```
>fix(Ex.6.2.7.a)
```

```
> Ex.6.2.7.a()
```

```
function (company.10var=company.10var)
{
# function takes in company.10 var and splits it
#into success = company.10var.success
#into faliure = company.10var.faliure
x <- c(1:10)

#Subscripting to split company.10var, remove col_1
company.10var.success<-company.10var[company.10var[,1]==2 ,c(-1) ]
company.10var.faliure<-company.10var[company.10var[,1]==1 ,c(-1) ]

#Get mean vectors of two groups and difference
mean.s1 <- apply(company.10var.success,2,mean,na.rm=TRUE)
mean.f2 <- ap
ply(company.10var.faliure,2,mean,na.rm=TRUE)

#Get number of rows of two groups
n.s1 <-nrow(company.10var.success) # Can we use nrow / length
n.f2 <-nrow(company.10var.faliure)

#Get covariance matrixes two groups
S.1.s <-cov(company.10var.success)
S.2.f <-cov(company.10var.faliure)

##Group Variance
num <- ((n.s1-1)*S.1.s)+ ((n.f2-1)*S.2.f)
denom  <- (n.s1-n.f2-2)
S <- num/denom

#Group Mean
x.bar<- apply(company.10var[,1:10], 2, mean)

 # Classiffication Func & LDF   hard code
LDF<-(t(mean.s1-mean.f2))%*% solve(S)*x.bar #Double CHECK
Function <- ((t(mean.s1-mean.f2))%*%solve(S)*x.bar) -
(0.5*(t(mean.s1-mean.f2))%*%solve(S)*(mean.s1-mean.f2))

#Coefficients of linear discriminant function (LDF)
 LDF.coeff <- t(mean.s1-mean.f2)%*% solve(S)

 #LDF 2nd
 LDF <- LDF.coeff%*%x
 #CDF 2nd
 CDF <- LDF-0.5*LDF.coeff%*%(mean.s1+mean.f2)
list("Linear discriminant
coefficients"=LDF.coeff,LDF=LDF,CDF=CDF,2LDF.Ex.6.2.6.iv.a=LDF,2LD
F.coeff=LDF.coeff)
}
```

```
> Ex.6.2.7.a()
```

```
$`Linear discriminant coefficients`
            X1        X2       X3        X4       X5       X6        X7
X8         X9        X10
[1,] -6.179782 -11.46436 37.33781 -18.75429 18.84393 6.805857 -16.07244 -0.12139
45 0.02672031 -0.00662445

$LDF
         [,1]
[1,] 29.63856

$CDF
         [,1]
[1,] 29.54862

$LDF.Ex.6.2.6.iv.a
         [,1]
[1,] 29.63856

$LDF.coeff
            X1        X2       X3        X4       X5       X6        X7
X8         X9        X10
[1,] -6.179782 -11.46436 37.33781 -18.75429 18.84393 6.805857 -16.07244 -0.12139
45 0.02672031 -0.00662445
```

```
#Double Printed Answer to try and verify
```

Exercise 6.2.9.

A :n*m
# V(A)vector space of A, containing vectors of the same size.

Singular Value Decomposition
A rectangular matrix A (m x n) with real coefficients, admits a decomposition form:

$$A = U\Sigma V^T$$

Here U is the orthogonal (m x m) matrix
V is the orthogonal (n x n) matrix
Σ is a rectangular (m x n) matrix

U and V are called the singular vectors of matrix A.

```
function (mat)
# Ex. 6.2.9
#Function takes in A a rectangular matrix
# V(A)vector space of A, containing vectors of the same size
# rectangular matrix A (m x n) with real coefficients,
# admits a decomposition form A=USV^T
# Restriction 1: A is a matrix (rectangular)
# Function takes dimensions of A, dim(A)
# Restriction 2: A is rectangular
# Add column of zeros
# Do SVD decomposition
{
if(!is.matrix(mat)) print("Error: 'mat' is not a martix.")
if(nrow(mat)<=ncol(mat)) print("Error: 'mat' is not rectangular.")

dim <-dim(mat)

if (dim[1] > dim[2])

mat <-matrix(c(mat,rep(0, (dim[1] -dim[2]) * dim[1])), nrow=dim[1])

u <-svd(mat)$u
v <-svd(mat)$v
d <-svd(mat)$d

r <-sum(d > 1e-05) # rank of mat

orthogonalbasis.VA <- u[, 1:r]
orthogonalbasis.perp.VA <- u[ ,(1 + r):dim[1]]

list(rank_mat = r,
Orthogonal_Basis_Vec_A = orthogonalbasis.VA,
Orthogonal_Basis_perp_Vec_A = orthogonalbasis.perp.VA)

}
```

```
> Ex.6.2.9(A)
$`rank_mat`
[1] 2

$Orthogonal_Basis_Vec_A
           [,1]       [,2]
[1,] -0.2161115 -0.1238314
[2,] -0.4322230 -0.2476628
[3,] -0.6282210 -0.5378085
[4,]  0.2965658 -0.5414257
[5,] -0.5327909  0.5839086

$Orthogonal_Basis_perp_Vec_A
           [,1]       [,2]       [,3]
[1,]  0.6684681 -0.5248853 -0.4643354
[2,] -0.2844382  0.4685199 -0.6718857
[3,] -0.1245259 -0.2180578  0.5030353
[4,]  0.5396605  0.5487365  0.1629777
[5,]  0.4068232  0.3953777  0.2309892

> Ex.6.2.9(2)
[1] "Error: 'mat' is not a martix."
Error in if (nrow(mat) <= ncol(mat)) print("Error: 'mat' is not
rectangular.") :
  argument is of length zero
```

Exercise 6.2.12.

Game

Consider the following game. You are given a computer screen containing a rectangle filled at random with evenly spaced letters. Repetitions of the same letter are allowed. The challenge is to select sequentially the first n letters of the alphabet as quickly as possible. You must read each line from left to right and from top to bottom. Going backwards is not allowed. The time to complete the task is taken as well as if the rules have been obeyed. Program an R version of this game.

Try this one as best as you can. If you have attempted it but could not get it to work properly, hand in your attempt.

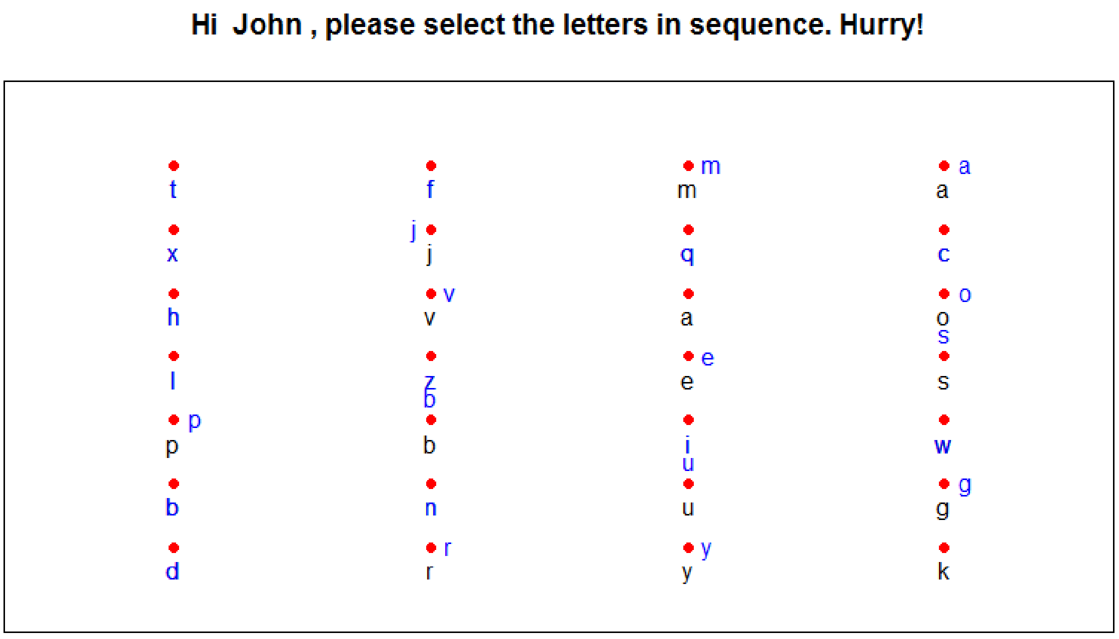
fix(Ex.6.2.12)

```
function ()
{
    ## Game instructions given to user in console
    print("The goal of this game is to pick the letters on the console in
sequence. You will be timed and the aim is to complete is as soon as
possible. Your time begins after inputting your name.Let the games
begin!!!")

    # Prompt user to enter name and the game begins
    name <- readline(prompt="Enter your name: ")
    print("Your time starts now")
    start.time <- proc.time()

    # Create the empty canvas with the letters and points below, note
that the letters are
    #In a different random order everytime one restarts the game
    plot(c(0, 200), c(0, 400), type= "n", xlab = "", ylab =
"",yaxt="n",xaxt="n", main= paste("Hi ",name,", please select the letters
in sequence. Hurry!"))

    # Create the points to plot and sample with replacement to make the
points random in how they are presented to the user
    # reorder sampling of x values
    # The game uses sampling with replacement to create a new order to
plot the letters which gives the user
    #The illusion of the game creating a new random canvas
    pts <- seq(25,175,50)
    pts.replace <- sample(pts,4,replace = FALSE)
    pts.fin <- rep(pts.replace,7)
    #reorder sampling of y values
    y.val <- seq(50,350,50)
    y.sample <- sample(y.val,7,replace = FALSE)
    y.fin <- rep(y.sample,4)

    # Plot the letters for the user
    points(pts.fin, y.fin,col="red",pch=16)

    text(x=pts.fin,y=y.fin,labels=letters,pos=1)

    temp1 <- identify(x=pts.fin,y=y.fin,
                      labels=letters,n=3,col="lightblue")
#Possible sequences that would result in a Success message once game is
complete
    vec1 <- c(1:26)
    vec2 <- c(27,2:26)
    vec3 <- c(1,28,3:26)
    vec4 <- c(27,28,3:26)


    #Check whether the vector that we get as an answer agrees with the
sucess vectors
    #Print a message to the user congradulating him on sucess, then give
the user his elapsed time
    #Give the user the sequence of his results
    if(temp1==vec1 || temp1==vec2 || temp1==vec3 || temp1==vec4)
        result <- c(paste("Congradulations,",name," completed the game
successfully"))
    else
        result <- c("Unfortunately your selected sequence was incorrect,
please try again")

    end.time <- proc.time()

    time.taken <- end.time[3]-start.time[3]

    list(result = result,time_taken = time.taken,Your_Sequence_Was =
temp1)
}
```

We use the name John and choose the wrong sequence

```
> Ex.6.2.12 ()
[1] "The goal of this game is to pick the letters on the console in sequence. Y
ou will be timed and the aim is to complete is as soon as possible. Your time b
egins after inputting your name. Let the games begin!!!"
Enter your name: John
[1] "Your time starts now"
```

Figure 6: Game Failure

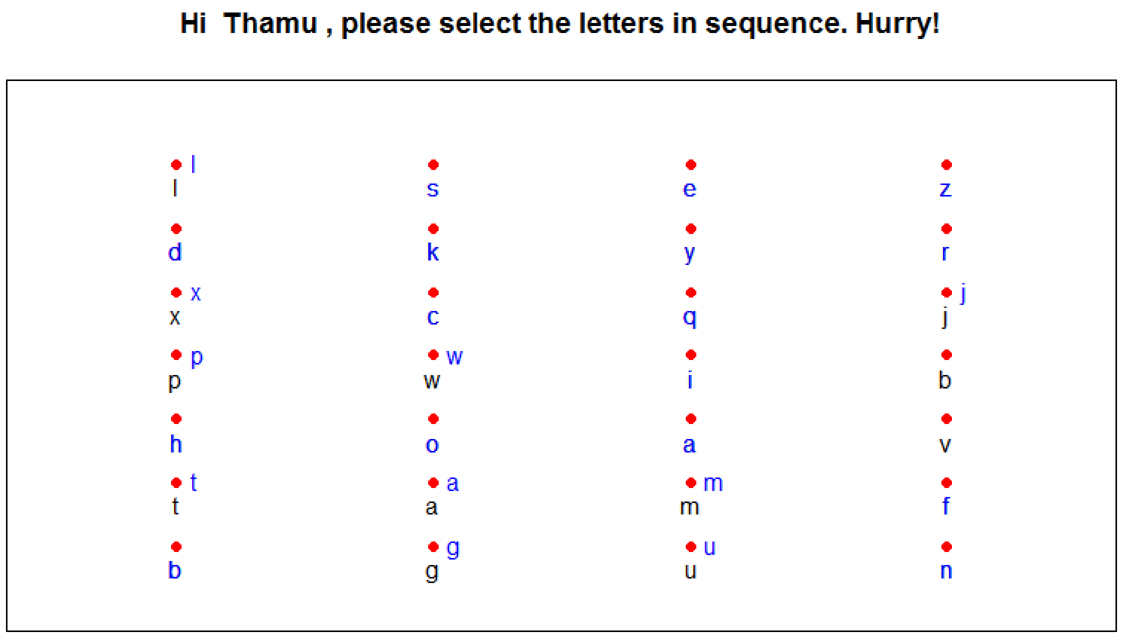


```
$`result`
[1] "Unfortunately your selected sequence was incorrect, please try again"

$time_taken
elapsed
 163.24

$Your_Sequence_Was
 [1]  2  3  4  5  6  7  8  9 10 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
28
```

We use the name Thamu and choose the correct sequence

```
> Ex.6.2.12 ()
[1] "The goal of this game is to pick the letters on the console in sequence. Y
ou will be timed and the aim is to complete is as soon as possible. Your time b
egins after inputting your name. Let the games begin!!!"

Enter your name: Thamu
[1] "Your time starts now"

$`result`
[1] "Congradulations, Thamu  completed the game successfully"

$time_taken
elapsed
 106.76

$Your_Sequence_Was
 [1]  1  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 23 24 25 26 27
28
```

Figure 7: Game Success



Further Description

The identify function allows us to find the closest observation to add to the sequence, we use this to allow the user to specify which letter he would like to add to the sequence.
We make use of the inbuilt letters vector in R and place each letter into a vector and then check whether our inputted vector agrees with one of the options that agree with a successful game. Take note because of the number of letters in the sequence and the number of spaces some letters must be repeated because of the recycling principle.

To refer to these repeated letters we create multiple vectors that allow the user to successfully complete the game whilst selecting a different letter that leads to the same correct sequence.

We also chose to return the sequence to the player in numeric form to allow him to easily spot his mistakes, hoping to allow the player to complete the game successfully on his next try or at least finish faster.

Section 7.9(b).

Discussion:
How do operators differ from ordinary R functions ?
First difference lies in how it is created.
To write an operator ensure that function is in "fix("%E%")‟.Where as a function can be created as fix(E).
The second difference lies in how it can be used. We illustrate that below.

```
> 4%E%2
[1] 2
```

The operator can be used in to perform mathematical operations such as '+' or '-' . Above it is used to calculate the Euclidean distance between 4 and 2.

```
> fix('%E%')
```

```
function (x,y)
{
#%E% : Operator for Euclidean Distance
# to write a operator ensure that function is in "" fix ("%E%")
# x and y must be vectors of the same length
# Function returns the Euclid distance between the two vectors
# Restrictions

if(!is.vector(x)|!is.vector(x)) stop("Arguments must be vectors \n")
if(!is.numeric(x)|!is.numeric(x)) stop("Arguments must be numeric \n")


if(abs(length(x)) != abs(length(y))) stop("The vectors are different lengths \n
")

ans <- sqrt(sum((x-y)^2))
return(ans)
}
```

```
> c(1,2,5,3)%E%c(1,2,3,4)
[1] 2.236068

> A<-rbind(c(1,2,3,4),c(1,2,5,3))
> dist(A)    # Double Check Answer with
         1
2 2.236068

> 4%E%2
[1] 2
```

Section 7.10.

What is a replacement function? Write your own replacement function and illustrate its usage.

Discussion:
What is a replacement function?
According to the textbook, ”A step-by-step R tutorial”, “functions ending in ‘<-‘are replacement functions. A replacement function appears on the left hand-side of the assignment symbol using the name without the ‘<-‘to replace contents of the objects appearing in its argument list by the contents of the object appearing at the right hand side of the assignment symbol.”

From my understanding the main differentiating factors that make a function a replacement, function is that:

- firstly it ends in ‘<-‘ (appears on the left hand side of the assignment symbol)
- Secondly, it allows one to replace objects in its argument list by contents of the object appearing at the right side of the symbol. i.e “return complete object with suitable changes made.”
- The last argument of “ the function corresponding to the replacement data on the right-hand side of the assignment, must be named value.”
- It is important to note that it is possible if not probable that “a companion function exists with the same name” as the replacement function “without the ‘<-‘.” E.g. ‘diag’ and ‘diag<-‘

Example 1 : pg 128
Replacement functions act like they modify their arguments in place, and have the special name xxx<-. They typically have two arguments (x and value), although they can have more, and they must return the modified object. For example, the following function allows you to modify the second element of a vector:

```
`fun<-` <- function(x, value) {
  x[2] <- value
  x
}
```

```
x <- 1:10
fun(x) <- 5L
x
[1]  1  5  3  4  5  6  7  8  9 10
```

Exercise 8.7 (d).

Note that you are asked to perform a **simulation *study*.** This task includes a summary of the results of your simulation study in a well-chosen graph together with a short report including conclusions and recommendations.
As a hint on how to output your results: read Section10.4.

Exercise 8.7 (d).i

Direct element Wise Calculation

```
>fix(Ex.8.7.d.i)

# remember moving column wise is moving down the rows

#R[j,i] <- S[j,i] %/% (sqrt(S[i,i]*S[j,j])) is wrong,
```

```
########## Ex.8.7.d.i
function (p)
{
start.time <- proc.time()
# create a matrix
mat <- matrix(rnorm(100*p), nrow=100)
cov.mat <- cov(mat)

S <-cov(data)
R <- matrix(rep(c(0)),nrow=nrow(S),ncol=ncol(S))

for(i in 1:nrow(S)){
      for(j in 1:ncol(S)){

      R[j,i] <- S[j,i] / (sqrt(S[i,i]*S[j,j]))
      }
}
end.time <- proc.time()
return((begin.time - end.time)[])
}
```

```
> Ex.8.7.d.i(n=100)
 user.self   sys.self    elapsed user.child  sys.child
         0          0          0         NA         NA
> Ex.8.7.d.i(n=1000)
 user.self   sys.self    elapsed user.child  sys.child
      0.24       0.02       0.25         NA         NA
> Ex.8.7.d.i(n=10000)
 user.self   sys.self    elapsed user.child  sys.child
     24.33       0.56      24.97         NA         NA
```

Exercise 8.7 (d).ii Two applications of sweep()

```
>fix(Ex.8.7.d.ii)

function (data)
{
#start.time <- proc.time()
 #if(missing(data)){ data<-matrix(rnorm(9),ncol=3,) }
       S <- cov(data)
 var.vec <- diag(S)
A <- sweep(S,2,sqrt(diag(S)),"/")
B <- sweep(A ,1,sqrt(diag(S )) , "/")
 # B is the coefficient of correlation 'R'
#proc.time()

}
```

Exercise 8.7 (d).iii

```
>fix(Ex.8.7.d.ii)

function (n)
{
#S is the cov mat
#R is the cor mat
#Function uses outer
#This takes in the X and Y arguments in the first two positions to release a
# third z argument after applying it into a bivariate FUN function.
#Returns elapsed time between last and first observation

begin.time <- proc.time()
mat <- matrix(rnorm( 100 * n), nrow= 100)
S <- cov(mat)

b <- sqrt(diag(S))
c <- outer(b,b,"*")

R <- S/c

end.time <- proc.time()
return((begin.time - end.time)[3])

}
```

Final Function

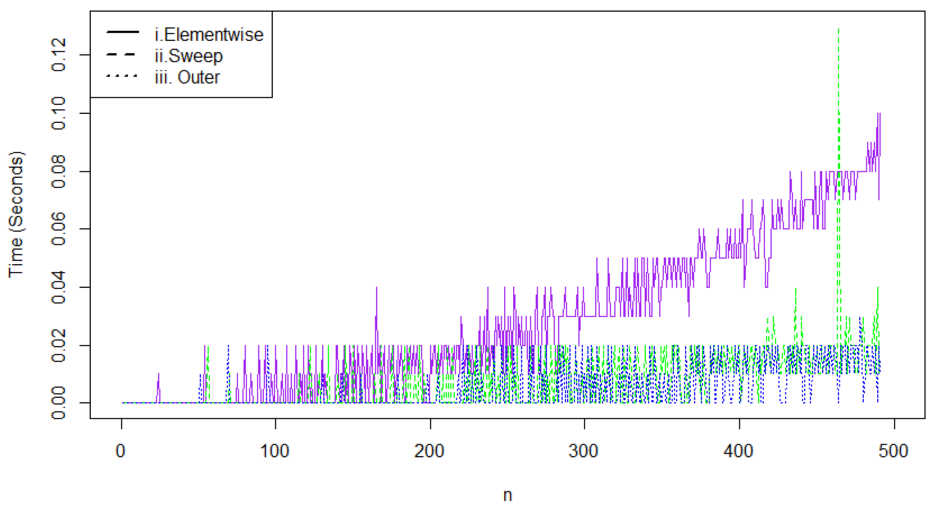
```
>fix(Ex.8.7.d)
```

```
function(x1=500){
# n: number of observations within sample
# p: number of samples

###########
A <- function (n)  {
data <- matrix(rnorm(100*n), nrow=100)
begin.time <- proc.time()
# create a matrix
S <-cov(data)
R <- matrix(rep(c(0)),nrow=nrow(S),ncol=ncol(S))

for(i in 1:nrow(S)){
        for(j in 1:ncol(S)){
        R[j,i] <- S[j,i] / (sqrt(S[i,i]*S[j,j]))
        }
}
end.time <- proc.time()
return((end.time - begin.time)[3])
}
###########
B <- function (n)
{
data <- matrix(rnorm(100*n), nrow=100)
begin.time <- proc.time()
       S <- cov(data)
 var.vec <- diag(S)

A <- sweep(S,2,sqrt(diag(S)),"/")
B <- sweep(A ,1,sqrt(diag(S)) , "/")
 # B is the coefficient of correlation 'R'
#proc.time()
end.time <- proc.time()
return((end.time - begin.time)[3])
}
###########
C <- function (n) {
mat <- matrix(rnorm(100*n), nrow=100)
begin.time <- proc.time()
#S is the cov mat
#R is the cor mat
#Function uses outer
#This takes in the X and Y arguments in the first two positions to release a
# third z argument after applying it into a bivariate FUN function.
#Returns elapsed time that the function takes to run
b <- sqrt(diag(S))
c <- outer(b,b,"*")
R <- S/c
end.time <- proc.time()
return((end.time - begin.time)[3])
}
###########
plot.FUN <- function(x=500){

n <- seq(10:x)

a <- sapply(n, A)
b <- sapply(n, B)
c <- sapply(n, C)
plot(1,xlim=c(0,x), ylim=c(0, max(c(a,b,c))), type="n",xlab="n", ylab="Time (Sec
onds)", main="Plot of Elapsed Times")
points(a,type="l",lty=1,lwd=1,col="purple")
points(b,type="l",lty=2,lwd=1,col="green")
points(c,type="l",lty=3,lwd=1,col="blue")
```

```
legend("topleft", legend=c("i.Elementwise", "ii.Sweep", "iii. Outer"), lwd=c(2,2
,2), lty=c(1:3))
}
plot.FUN(x=x1)
########### 1
}
```

```
> Ex.8.7.d()
```

Plot of Elapsed Times



Interpretation
Our function tests the time elapsed from start of each alternative method as the sample size increases from 10 to 500. This is user defined in this case.
From inspection we can see that the methods are somewhat equal at very small sample sizes but as the sample size increases, the calculation method has significantly higher elapsed time than both the second and last method.
We conclude that the elementwise calculation is more computationally demanding followed by the second sweep method and lastly followed by the third method. The third method is most efficient method on average.

Section 8.10.i

Code functions **full2resp** and **resp2full** and illustrate the use of your functions on questdata.

Fix(Ex.8.10.c.i)

```
function(){
# Ex.8.10.c.i
# This function creates questdata and answers Ex.8.10.c.i on pg 144
#Function returns all as a list as to extract questdata
```

```
#and to explain functions dublicate() and unique
#Assume was part of exercise
```

```
Q1 <- c('b','d','a','a','b','a','b','d','c','b')
Q2 <- c('c','d','d','d','c','d','c','d','b','c')
Q3 <- c('a','c','c','c','a','c','a','c','a','a')
Q4 <- c('d','a','e','e','d','e','d','a','e','d')
dfram <-matrix(c(Q1,Q2,Q3,Q4),ncol=4,nrow=10) #looses the col names
#print(dfram)

#dfram: n x p dataframe

if(!is.data.frame(dfram)) dfram <- as.data.frame(dfram) # make a
dataframe
colnames(dfram)<-c("Q1","Q2","Q3","Q4")
questdata <- dfram
# Ex.8.10.i
temp1 <-unique(questdata[,1])
temp2 <-duplicated(questdata)
temp3 <-duplicated(questdata, MARGIN=1)
temp4 <-duplicated(questdata, MARGIN=2)
temp5 <-unique(questdata)
temp6 <-unique(questdata,MARGIN=1)
temp7 <-unique(questdata,MARGIN=2)

list(questdata=questdata, Ex.8.10.i1 = temp1,Ex.8.10.i2 = temp2
,Ex.8.10.i3 = temp3 ,
Ex.8.10.i4 = temp4,Ex.8.10.i5 = temp5 ,
Ex.8.10.i6 = temp6 ,Ex.8.10.i7 = temp7)
}
```

```
> Ex.8.10()
$`questdata`
   Q1 Q2 Q3 Q4
1   b  c  a  d
2   d  d  c  a
3   a  d  c  e
4   a  d  c  e
5   b  c  a  d
6   a  d  c  e
7   b  c  a  d
8   d  d  c  a
9   c  b  a  e
10  b  c  a  d
```

```
$Ex.8.10.i1
[1] b d a c
Levels: a b c d

$Ex.8.10.i2
 [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE

$Ex.8.10.i3
 [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE

$Ex.8.10.i4
 [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE

$Ex.8.10.i5
  Q1 Q2 Q3 Q4
1  b  c  a  d
2  d  d  c  a
3  a  d  c  e
9  c  b  a  e

$Ex.8.10.i6
  Q1 Q2 Q3 Q4
1  b  c  a  d
2  d  d  c  a
3  a  d  c  e
9  c  b  a  e

$Ex.8.10.i7
  Q1 Q2 Q3 Q4
1  b  c  a  d
2  d  d  c  a
3  a  d  c  e
9  c  b  a  e
```

duplicated(),
This function takes in a sequence of variables and returns a sequence of whether each observation in the sequence is duplicated.

```
> duplicated(c(1,2,3,2))
[1] FALSE FALSE FALSE  TRUE

> !duplicated(c(1,2,3,2))
[1]  TRUE  TRUE  TRUE FALSE
```

!duplicated(x) gives a boolean vector which will be true the first time a value appears. The results will be in the order that the values are encountered in the vector input.

Unique( ),

To get just the unique values in a sequence, the unique function can be used. The results will be in the order that the values are encountered in the vector being studied.

Section 8.10.iii

full2resp

Obtain the response pattern representation of questionnaire data like those given above. Test your function on quest data.

```
> questdata<-Ex.8.10()$`questdata`
> # take quest data as a data frame from the previous question
> Ex.8.10()$`questdata`
   Q1 Q2 Q3 Q4
1   b  c  a  d
2   d  d  c  a
3   a  d  c  e
4   a  d  c  e
5   b  c  a  d
6   a  d  c  e
7   b  c  a  d
8   d  d  c  a
9   c  b  a  e
10  b  c  a  d
> fix(full2resp)
```

```
function (data=questdata)
{
# resp to full
# note the use of aggregate & rep
# used freq as a counter and changed into a list to count

data <-data.frame(data)

freq <-rep(1,nrow(questdata))
temp1<-aggregate(freq, by = as.list(data), FUN=sum)

colnames(temp1) <-c ("Q1","Q2","Q3","Q4","Freq")
return(temp1)
}
```

```
> full2resp()

  Q1 Q2 Q3 Q4 Freq
1  d  d  c  a    2
2  b  c  a  d    4
3  c  b  a  e    1
4  a  d  c  e    3
```

Section 8.10.vi

resp2full

Obtain the full data set given its response pattern representation.

```
function(resp.data = full2resp() , freq = full2resp()[,5] )
{
# for freq the line 5 does not exist, its a counter

data<-NULL
for (i in 1:nrow(resp.data)){
     for(j in 1:freq[i]){
          data<-rbind(data, resp.data[i, ])
      }
}

rownames(data) <- 1:nrow(data)
return(data)
}
```

```
> resp2full() # illustrates how function works with ‘Freq’

   Q1 Q2 Q3 Q4 Freq
1   d  d  c  a    2
2   d  d  c  a    2
3   b  c  a  d    4
4   b  c  a  d    4
5   b  c  a  d    4
6   b  c  a  d    4
7   c  b  a  e    1
8   a  d  c  e    3
9   a  d  c  e    3
10  a  d  c  e    3
```

```
> output<-full2resp()
> resp2full(output[,1:4],output[,5])
   Q1 Q2 Q3 Q4
1   d  d  c  a
2   d  d  c  a
3   b  c  a  d
4   b  c  a  d
5   b  c  a  d
6   b  c  a  d
7   c  b  a  e
8   a  d  c  e
9   a  d  c  e
10  a  d  c  e
```

Section 8.11(c)

```
>fix(factorial)
```

```
function (x)
{
# This Function Calculates the factorial of x
# Recursion and function Recall to Calculate x!.

if( !is.numeric(x) ) return(c("Entry Must be numeric"))

if(!((x-(x%/%1))==0)) return("Error: Not a whole number")

if( x < 0 )
     {return("Error: Please enter a positive number")
}
else
     {
if( x == 0 ) 1
else {x*Recall(x-1)}
}

}
```

```
> factorial(5)
[1] 120
> Ex.8.11.c(5)
[1] 120
> factorial(2)
[1] 2
> Ex.8.11.c(2)
[1] 2
```

Section **10.4.9.**

Figure 1: Bar Plot with different colours
Figure 2: Bar Plot using lines
Figure 3: Histogram with Density superimposed on the graph
Figure 4: Line

Title for Grid “Ex.10.4.9” which is to create ‘Figure 10.4.4 ’

Barplot Empty to do Fig 10.4.4 pg 171

```
>fix(Ex.10.4.9)
```

```
function ()
{
require(MASS)
on.exit(par(mfrow=c(1,1)))
par(mfrow=c(2,2))
par(mar=c(4,4,6,4) )
```

```
## Figure 1: Barplot with different colours and labels
# Note the 6 levels, the y-lab is the median MPG in the city per
category(type car)

levels <- levels(Cars93[,"Type"])

# take the 3 first letters of level as name1, for x-axis labels
a <- levels(Cars93[,"Type"])
name3 <- substr(a,start=1,stop=3) # vec a is vector of names
name2 <- substr(a,start=1,stop=2) # 2 char

ba <- barplot( tapply(Cars93[,"MPG.city"],Cars93[,"Type"],median),
col=c(1:length(levels)),
ylim=c(0,35),xlab="Type of Car",
ylab="Median MPG in City", names.arg = name3,
main="Bar plot of MPG in City")

# axis below with same
axis(1, at=ba , tick=TRUE, labels=F , pos=0)
text(paste("n=",table(Cars93[,"Type"])),x=ba,y=(tapply(Cars93[,"MPG.city"
],Cars93[,"Type"],median)),pos=3)
```

```
# Figure 2: Barplot using lines of MPG City
# Use lines and draw lines from zero to the highest point
# pre set the table & plot empty table with labels, low level for high to
# come
temp1 <- tapply(Cars93[,"MPG.city"],Cars93[,"Type"],median)

# empty plot, with labels and placeholders
plot(0:7, c(rep(0,7), max(temp1) ) , ylim=c(0,40), type="n",
xlab="", ylab="Meadian MPG in City",
main=" Bar plot of MPG in City", xaxt="n")

# plot the lines
for (i in 1:length(temp1)) lines(c(i,i), c(0, temp1[i]), lwd=3 )

title(xlab="Type of Car ", mgp = c(1.65,0.5,0))
axis(side=1, at=1:6, labels=name2, mgp=c(1.65,0.5,0))
```

```
# add labels higher than temp1, with" = "as sep
text(1:6, temp1+3, paste("n", table(Cars93[,'Type']), sep=" ="))
abline(h=0)  # add horizontal line on axis
```

```
# Figure 3
hist(Cars93[,"Weight"] , freq=FALSE,xlab="Weight",
ylim = c(0,(6.5*10^-4)), col="grey",
main = "Histogram with Normal PDF of Weight" )

# plot normal PDF
# length(Cars93[,'Weight']) [1] 93
pts <- seq(from=1500,to=5000,length=93)
dnorm1 <-
dnorm(pts,mean=mean(Cars93[,"Weight"]),sd=sqrt(var(Cars93[,"Weight"])))
lines(x=pts,y=dnorm1,col='red')
```

```
# Figure 4

# Create Graph4 to hold data, City(x-axis), Hwy(x-axis)and Type

 Graph4<-Cars93[,c("MPG.city","MPG.highway","Type")]
 # subscript and only take when Type is 'Compact','Large'or 'Small'
 Graph4<-
Graph4[Cars93[,"Type"]=="Compact"|Cars93[,"Type"]=="Large"|Cars93[,"Type"
]=="Small",]

Graph4[,3]<-factor(Graph4[,3])
Hwy<-tapply(Graph4[,2],Graph4[,3],mean)
Cit<-tapply(Graph4[,1],Graph4[,3],mean)

G4<-barplot(rbind(Cit,Hwy),beside=T,space=c(1,1.5,3,1.5,3,1.5),
names.arg=rep(c("Cit","Hwy"),3)
,ylim=c(0,35),
ylab="Miles per Gallon", col=c(3,4)
,main="Mean Mpg")

G4<- matrix(G4,ncol=3)
axis(side=1,at = apply(G4,2,mean),line=3 ,
labels=c("Compact","Large","Small"),tick=FALSE,pos=-5)

abline(h=0)
```

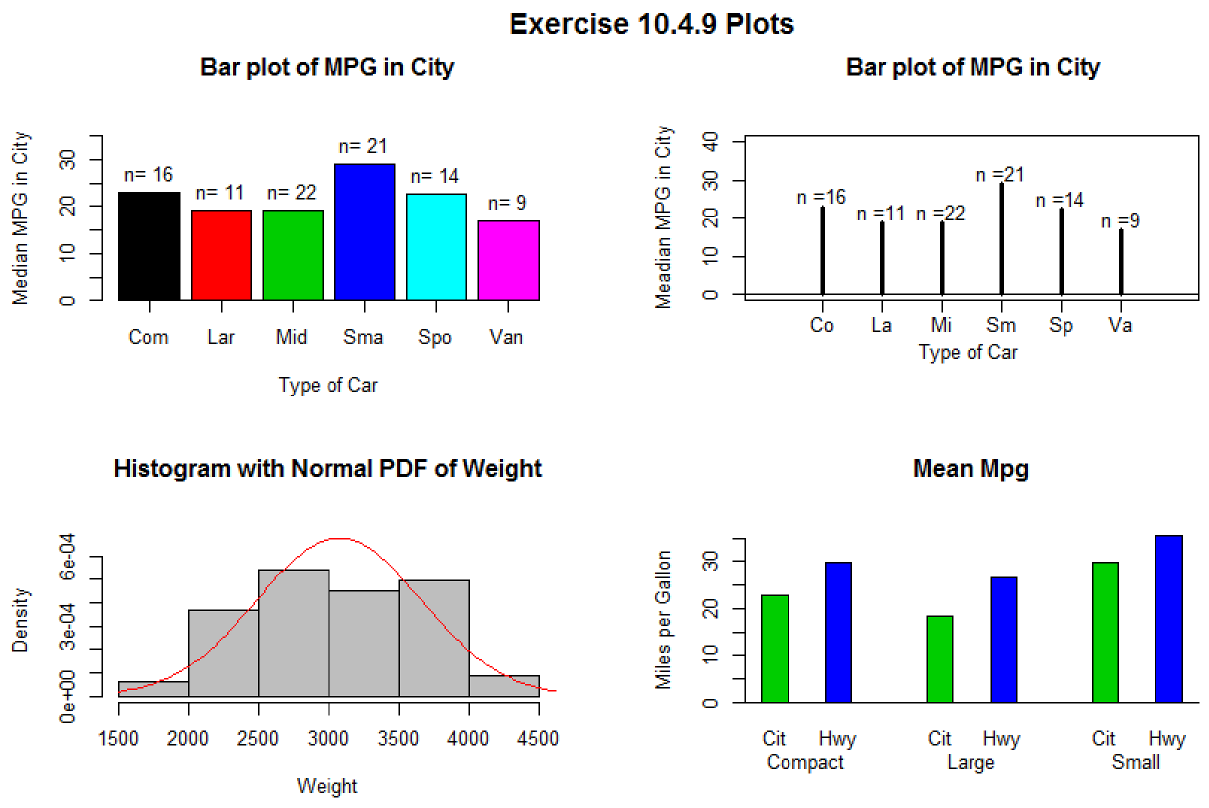
```
# Add main title
par(new=T)
par(mfrow=c(1,1))
       par(mar=c(3,3,2,3))
       plot(1:10,1:10,type = "n",xlab="",ylab = "",axes = F)
       title(main=paste(" Exercise 10.4.9 Plots"))

}
```

Figure 10.4.4

Figure 9 : Figure 10.4.4 in text book

Section 10.9.

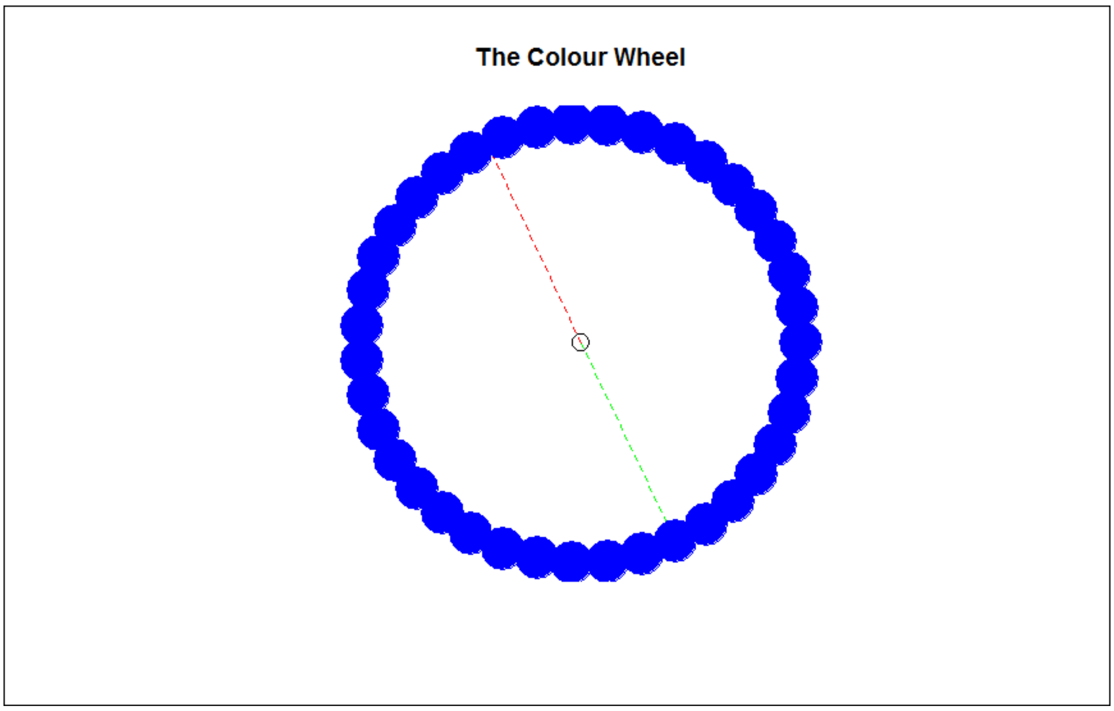
```
>fix(Ex.10.9)
```



```
function (sleep=0.1)
{
# function that uses plots and lines to plot a moving circle
# moving lines rotate around the center point
# Press ESC to end animation
# uses nested for loops, points and lines
# first for loop draws lines and centre point
# second for loop creates the points on the outside of the circle
# circle is in fact points
# centre's cordinates are [0,0]
# Sleep allows user to control Sys.sleep which dictates how slow the
# plot seems to reprint at different coordinates from visual inspection

n <- 40
t <- seq(0,2*pi,length=n)
x <- cos(t)
y <-sin(t)

for (i in 1:n)
{ plot.new()
plot.window(c(-1,1), c(-1,1),asp=1)
points(x[i],y[i],pch=16,cex=2)
lines(x=c(0,x[i]), y=c(0,y[i]), lty=2,col="green") # line 1
lines(x=c(0,-x[i]), y=c(0,-y[i]), lty=2,col="red") # second line
points(x=0,y=0,cex = 2)
#nested for loop that keeps points and draws outer circle
for(j in 1:100){
points(x[j],y[j],pch=20,cex=7,col="blue")
}

box(which = "outer") # Add a box to outside
title("The Colour Wheel") # Give a nice title to the animation

Sys.sleep(sleep)
#Sys.sleep() suspends execution of R expressions
#for a given number of seconds
}
Recall(sleep)
}
```

```
>Ex.10.9 (sleep=0.1)
```

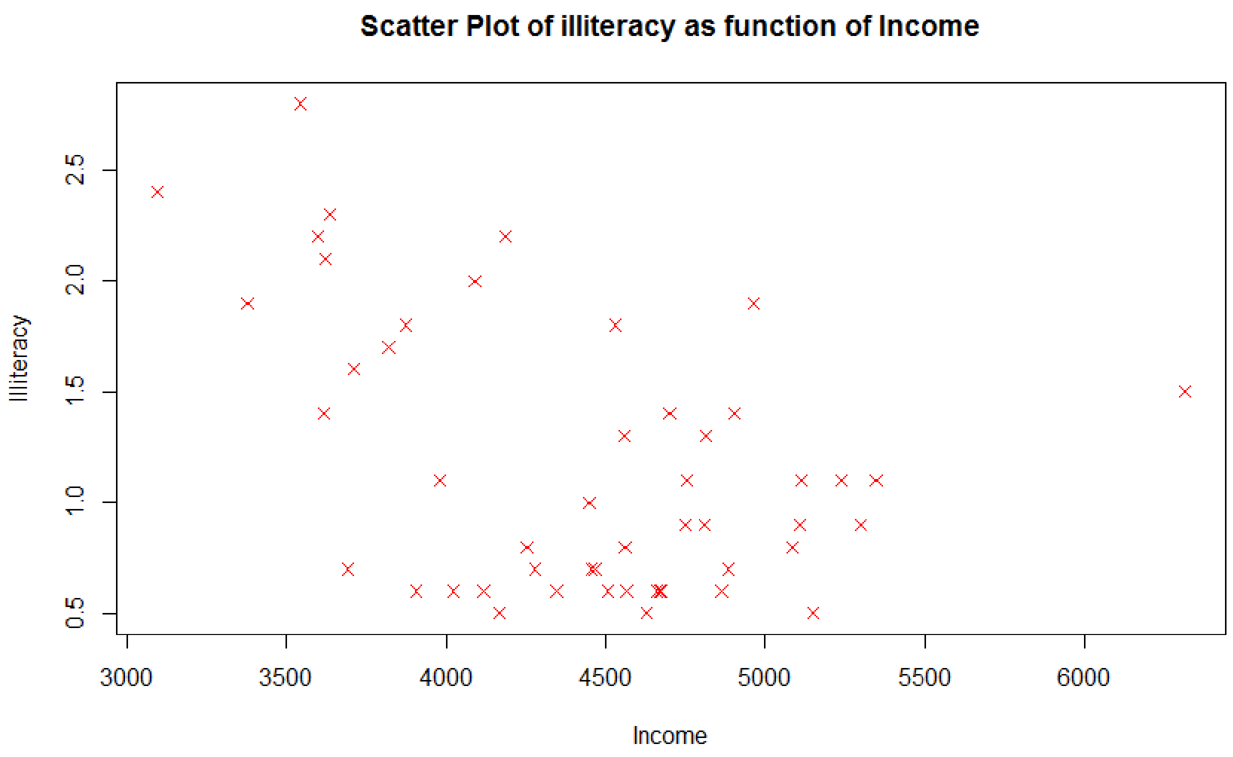
Figure 10: Moving Colour Wheel



Exercise 10.10.3.

```
>#  Plot a scatter plot of illiteracy as function of Income
>plot(x=state.x77[,2] ,y=state.x77[,3],main = "Scatter Plot of illiteracy as fun
ction of Income",col=10,pch=4,xlab = "Income",ylab = "Illiteracy")
```

Figure 11: Illiteracy Plot



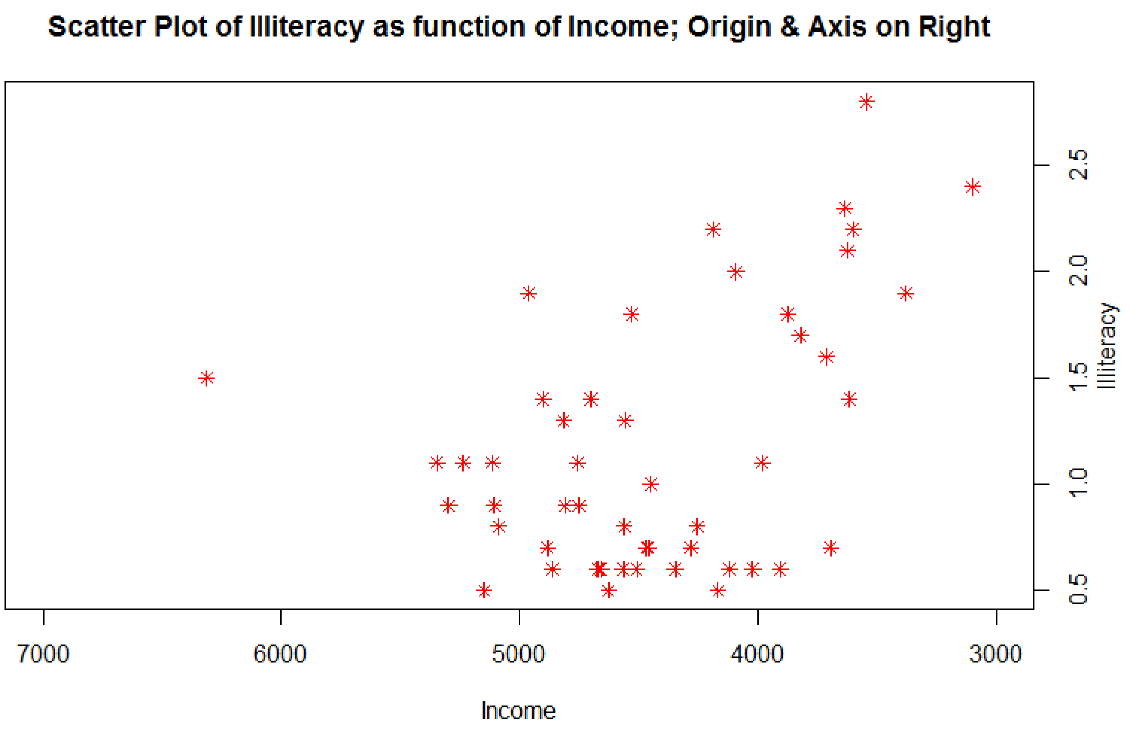
Interpretation
We see a somewhat negative relationship between Illiteracy and Income. As illiteracy increases income decreases.

```
>fix(invert)
```

```
function (x=state.x77[,2],y=state.x77[,3],...)  {
#Function plots scatterplot with axis on the right side
#allows user to change data accordingly and alterations to plot

# Note Important to use rev on both x and y
par(mar=c(6,3,4,6))

x1 <- rev(x)
y1 <- rev(y)

text1 <-c("Scatter Plot of Illiteracy as function of Income; Origin & Axis on
Right")

# ylab="",yaxt="n" in plot to remove the old axis
# changed the xlim =c(7000,3000), to reverse x

plot(x=x1 ,y=y1, main = text1,xlim =c(7000,3000) ,col=10,pch=8,xlab =
"Income",ylab="",yaxt="n" ,...)

# Move axis to right side
axis(side=4,)
mtext(side=4,text="Illiteracy",line=2)
}
```

```
> invert()
```

Figure 12: Illiteracy Plot, Axis on Right



Interpretation
Above n the inverted graph we still see a somewhat negative relationship between Illitracy and Income. As illiteracy increases income decreases.

Note that the origin has moved to the right but the values on the y still have the same positions, we has to invert use rev function on both the x and the y to allow this plot to be made.

Exercise 11.7.

Linear Regression and using Anova

Exercise 11.17.a

```
> Ex.11.7.a()
$`one.way.anova`
Analysis of Variance Table

Response: MPG.highway
          Df Sum Sq Mean Sq F value Pr(>F)
ManFact    3  39.77  13.256  0.5671 0.6442
Residuals 17 397.38  23.375

$two.way.anova
Analysis of Variance Table

Response: MPG.highway
                  Df  Sum Sq Mean Sq F value   Pr(>F)
ManFact            3  39.768  13.256  0.8224 0.504507
WeightGrp          1 181.894 181.894 11.2852 0.005128 **
ManFact:WeightGrp  3   5.947   1.982  0.1230 0.944876
Residuals         13 209.533  16.118
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretation:
The null hypothesis of the interaction term is that there is no significant interaction.

ManFact:WeightGrp , p-value = 0.944876 , therefore we cannot reject the null hypothesis. The interaction term has no significant interaction.
The significance is so high that we cannot reject the null hypothesis.

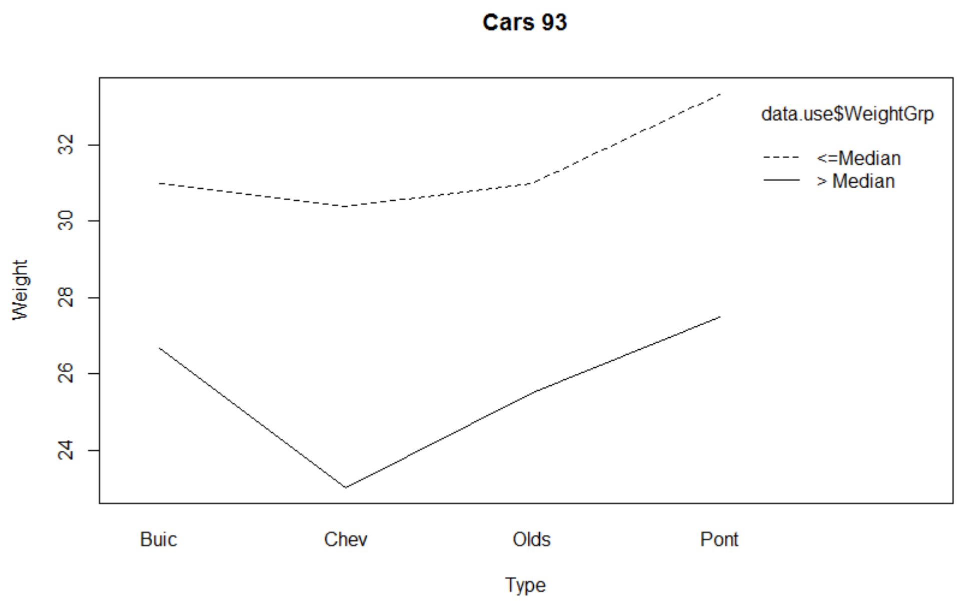
We then move on the main effects.
For ManFact p-value=0.504507, therefore we cannot reject the null hypothesis. The term ManFact has no significant interaction.

For WeightGrp p-value=0.005128, we reject the null hypothesis. The term WeightGrp has a significant interaction with MPG.highway.

Theory Discussed in class:
The degrees of the main effects is the number of levels of the main effects minus one. For ManFact (i.e 4-1=3) it is 3 and for WeightGrp (2-1=1) it is 1, the ineteraction is the product of those two for example in ManFact:WeightGrp it is 3 (i.e 3 x 1 =3) .

You then have the residuals; in this case, you have 21 objects therefore 21 minus the levels 1 minus one, minus levels 2 minus one. If you add those values then you should have 21. (3+1+3+13=20) the extra one that is missing comes from your constant term.

Figure 13: Cars 93 Interaction Plot



Interpretation
We see from the interaction plot that there is an interaction as the two lines are parallel.

We conclude that there is some indication of interaction in the sense that the manufacturer does slightly influence the behavior of the miles per gallon. Because if the manufacturer is Chevrolet it seems like the drop is much more severe in miles per gallon. The heavy Chevrolet cars go down more than the others do.

Theory: Behind Graph

Note: Interaction plot has three arguments, first argument is the first group, second argument is the second group (does not matter which comes first), the third argument is a dependent variable. First two variables are grouping variables and the third is dependent variable.

On the verticle axis you have the local dependent variable which is the third argument of the interaction function. You then have the levels of the first argument ("Buic", "Chev", "Olds", "Pont") If you switch the terms in interaction plot then you will have 4 lines each representing the cars and light, heavy and medium cars on the x axis. But meaning will be the same.

If there was no interaction the solid line and dotted lines should be parallel as what happens in the one does not influence the other.

Interaction means if the two factors interact the presence of one factor will influence the other this is seen as deviation from parallel line.

```
> Ex.11.7.a() # In Book
function ()
{       require(MASS)

data.use <- Cars93[grep("Buick|Chevrolet|Oldsmobile|Pontiac", Cars93[,"Manufacturer"]),
]

short <- c("Buic", "Chev", "Olds", "Pont") # Levels of 1st argument

data.use$ManFact <- factor(short[match(substring(data.use[,"Manufacturer"],1,4), short)
])

# One way Anova
one.way.anova <- aov(MPG.highway ~ ManFact, data = data.use)

group.mns <- tapply(data.use[,"MPG.highway"], data.use$ManFact,mean)

#Two-way anova

data.use$WeightGrp <- factor (cut(data.use$Weight, c(0, quantile(data.use$Weight,probs
=c(0.5,1))),labels = c("<=Median", "> Median")))


two.way.anova <- aov(MPG.highway ~ ManFact + WeightGrp + ManFact:WeightGrp, data=data.u
se)

# ManFact:WeightGrp, gives us the interaction between the two

two.way.anova.lm <- lm(MPG.highway ~ ManFact + WeightGrp + ManFact:WeightGrp, data=data
.use)

par(mar=c(5,4,5,4))
interaction.plot(data.use$ManFact, data.use$WeightGrp, data.use$MPG.highway,xlab="Type"
,ylab="Weight",main="Cars 93")

list(one.way.anova=anova(one.way.anova), two.way.anova = anova(two.way.anova),
 two.way.anova.lm = anova(two.way.anova.lm))
}
```

Exercise 11.17

```
#Hints
#Is there a interaction between the two? We see that the two lines are
#not parallel. We see that there is an interaction, we run an ANOVA and
#look at the interaction first.
#We see a very small probability so the null hypothesis that there is no
#interaction is rejected.
```

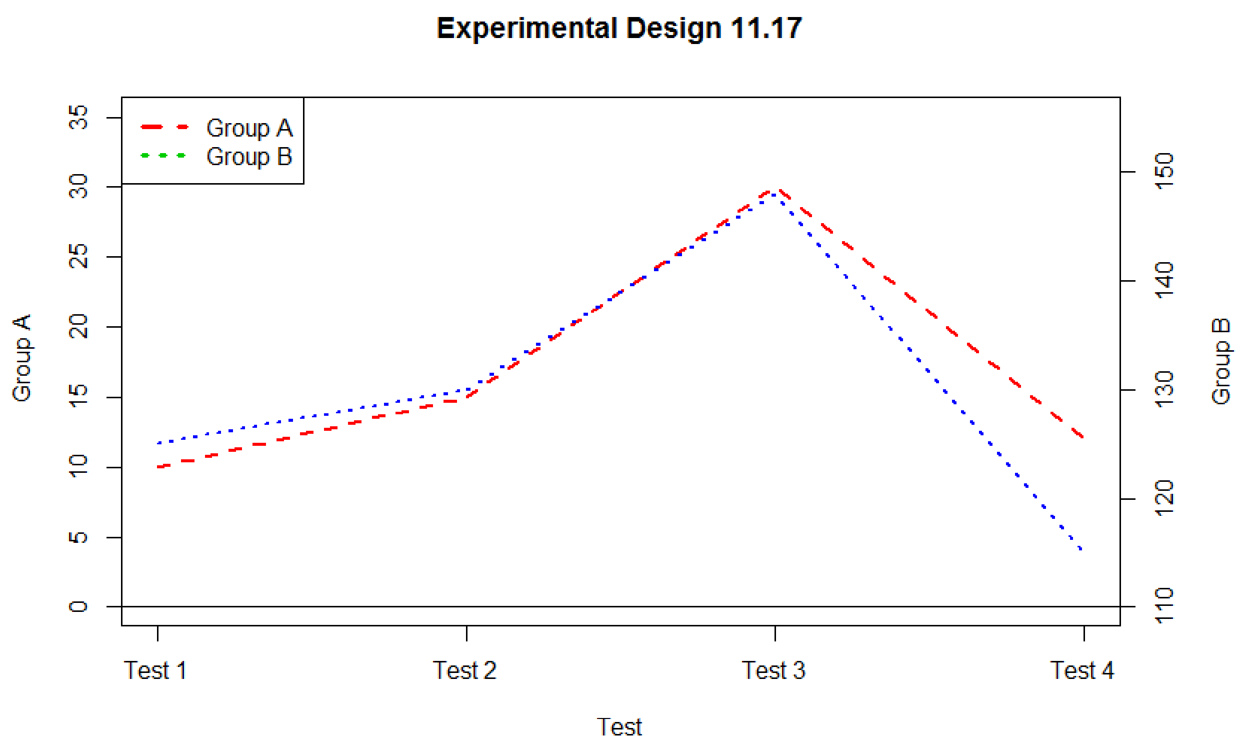
```
>Ex.11.17()
```

```
function (data)
{
# Function allows user to define data
# if data is not supplied function creates
# data, data goes through and creates plot of lines with axies on left
and right

# Firststly the function checks wherther necessayry data is supplied
if(missing(data)){
# Create the data set "data" with values .
# format rownames and column names
Test1 <- c(10, 125)
Test2 <- c(15, 130)
Test3 <- c(30, 148)
Test4 <- c(12, 115)
data<- data.frame(Test1, Test2, Test3, Test4)
rownames(data) = c("Group A", "Group B")
print(data)
}
# set margins and begin plotting
par(mar=c(5,5,5,5))

# Plot first GROUP and add labels to left axis
plot(x = c(1:4) , y = data[1, ], lwd=2, type = "l", col="red", lty = 2,
xlab= "Test" , ylab = " Group A",
 xaxt = "n",ylim=c(0,35), main="Experimental Design 11.7")
abline(h=0)  # horizontal line at zero

# Plot second group onto graph
par(new = TRUE)
plot(x = 1:4, y = data[2, ], lwd=2, type = "l", lty = 3, col ="blue",
xlab = "", ylab = "", xaxt = "n", yaxt = "n", ylim=c(110,155))

# Add tick marks to right side
axis(side = 4)
# Add axis labels to teh bottom of the graph with necessary labels
axis(side = 1, at = 1:4, labels = c("Test 1", "Test 2", "Test 3", "Test
4"))
# Add axis onto right side of graph with corresponding tickmarks
mtext(text = "Group B", side = 4, line = 3)
# Add legend to the graph top left with necessary labels
legend("topleft", legend = c("Group A","Group B"),lwd=3, col=c(2:3) ,lty
= 2:3)
}
```

```
>Ex.11.17()
```

```
         Test1 Test2 Test3 Test4
Group A     10    15    30    12
Group B    125   130   148   115
```

Figure 14: Experiment Design



PART C

Compulsory for all students registered for the full R course (FRB students are EXCLUDED)

Section 12.2 (a) and (b)

```
Poultry2012 <- Poultry.data[Poultry.data[ ,1]==2012, ]
```

```
> head(Poultry2012,1)
   Year  ProdUnit NetWt FeedMix  BrutWt
81 2012 CookleDoo 4.644     AB1 18.1684
```

Section 12.2 (a)

Repeat the above analysis using the other numeric variables as the response variable. (Only one numeric variable BrutWt).

Questions instructions.

>fix(Ex.12.2)

function ()

```
{

main.1=c("Interaction Plot: Poultry 2012, BrutWt on Response")
col.1 <- c("blue","orange","green")
# Below we fit an ineraction plot on Poultry2012 data
# The numeric response is BrutWt
# xpd = FALSE as not to clip the corners off

with(Poultry2012,interaction.plot(FeedMix,ProdUnit,response =
BrutWt,xpd=FALSE, main=main.1, col=col.1))

# Below we fit a two way ANOVA on Poultry2012 data
# if there is significant interaction we must move to look at the main
effects in another
# Numeric Response variable is BrutWt

two.way.anova <-lm(BrutWt~ProdUnit+FeedMix+ProdUnit:FeedMix,
data=Poultry2012)

list(Two_Way_Anova = anova(two.way.anova))
}
```

```
> Ex.12.2()
```

```
$`Two_Way_Anova`
Analysis of Variance Table

Response: BrutWt
                  Df Sum Sq Mean Sq F value    Pr(>F)    
ProdUnit           2 664.11  332.06 411.482 < 2.2e-16 ***
FeedMix            2 870.81  435.40 539.550 < 2.2e-16 ***
ProdUnit:FeedMix   4 157.65   39.41  48.841 < 2.2e-16 ***
Residuals        349 281.63    0.81                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

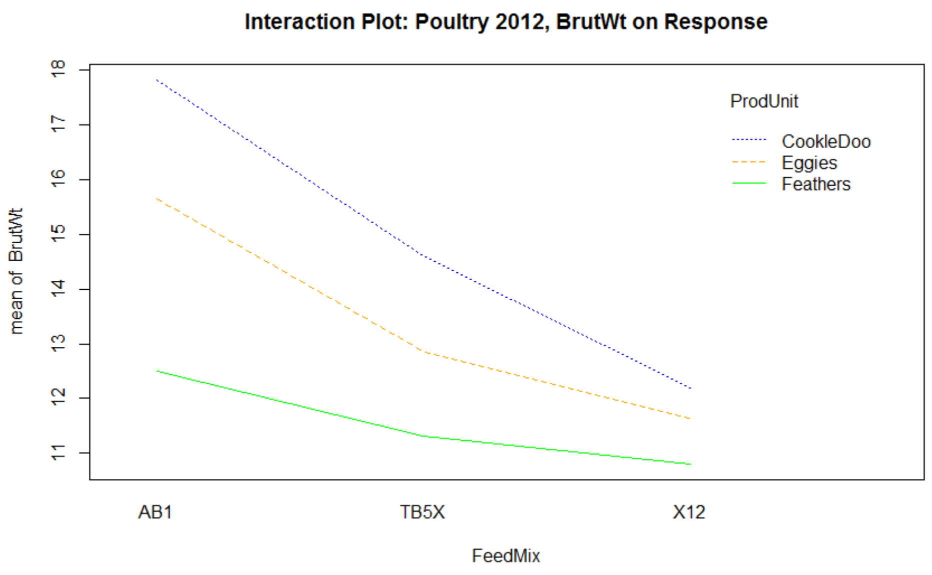
$H_0: (\alpha\beta)_{11} = (\alpha\beta)_{12} = \ldots\ldots = (\alpha\beta)_{33}$
H_A: *There exists an Interaction*

Interpretation:

We reject the null hypothesis there exists an interaction.
The interaction term ProdUnit:FeedMix is significant p = 2.2e-16 $< \alpha = 0.05$.
There exists an interaction when using ProdUnit and FeedMix to predict response BrutWt

If an intection term is significant the main effects must be kept irregardless of if they are significant or not. This is called the hirarchy principle and a principle component in dealling with interaction terms.

Figure 14: Interaction Plot Poultry



Interpretation
This interaction plot tells us that **there is an interaction between FeedMix and ProdUnit.**
We see this as we know that an interaction effect tells us that the effect of one factor depends on the movement of the other factor and it's shown by the lines in our plot running parallel.

Interaction effect, ProdUnit:FeedMix (Two Way ANOVA)

p-value = 2.2e-16 < $\alpha =$ 0.05 therefore we fail to reject the null hypothesis . There is some significant interaction between ProdUnit and FeedMix when predicting response BrutWt.

The correct procedure would be to then rerun an ANOVA with each main effect to conclude with certainty which main effect is statistically significant.

Section 12.2 (b)

```
>fix(Ex.12.2.b)
```

```
function ()
{
Table<-with(Poultry2012,table(FeedMix,ProdUnit))
main=c("Interaction Plot: Poultry 2012")
col.1 <- c("blue","red","purple")

#with(data ,
interaction.plot(independent_var1,independent_var2,dependent_variable) )
# xpd= FALSE ensures plot fits
with(Poultry2012,interaction.plot(FeedMix,ProdUnit,response=NetWt,xpd=FALSE,m
ain=main,col=col.1 ))
two.way.anova <-lm(NetWt~ProdUnit+FeedMix+ProdUnit:FeedMix,data=Poultry2012)

list(Table = Table , Two_Way_ANOVA = anova(two.way.anova))
}
```

```
>Ex.12.2.b ()
```

```
$`Table`
       ProdUnit
FeedMix CookleDoo Eggies Feathers
   AB1         40     39       40
   TB5X        40     40       40
   X12         40     40       39

$Two_Way_ANOVA
Analysis of Variance Table

Response: NetWt
                  Df  Sum Sq Mean Sq F value    Pr(>F)
ProdUnit           2  59.077  29.538 238.389 < 2.2e-16 ***
FeedMix            2 187.266  93.633 755.662 < 2.2e-16 ***
ProdUnit:FeedMix   4  16.315   4.079  32.918 < 2.2e-16 ***
Residuals        349  43.244   0.124
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

$H_0: (\alpha\beta)_{11} = (\alpha\beta)_{12} = \ldots\ldots = (\alpha\beta)_{33}$
$H_A: There\ exists\ an\ Interaction$

Interpretation:

We reject the null hypothesis there exists an interaction.
The interaction term ProdUnit:FeedMix is significant p = 2.2e-16 $<\alpha = 0.05$.
There exists an interaction when using ProdUnit and FeedMix to predict response NetWt

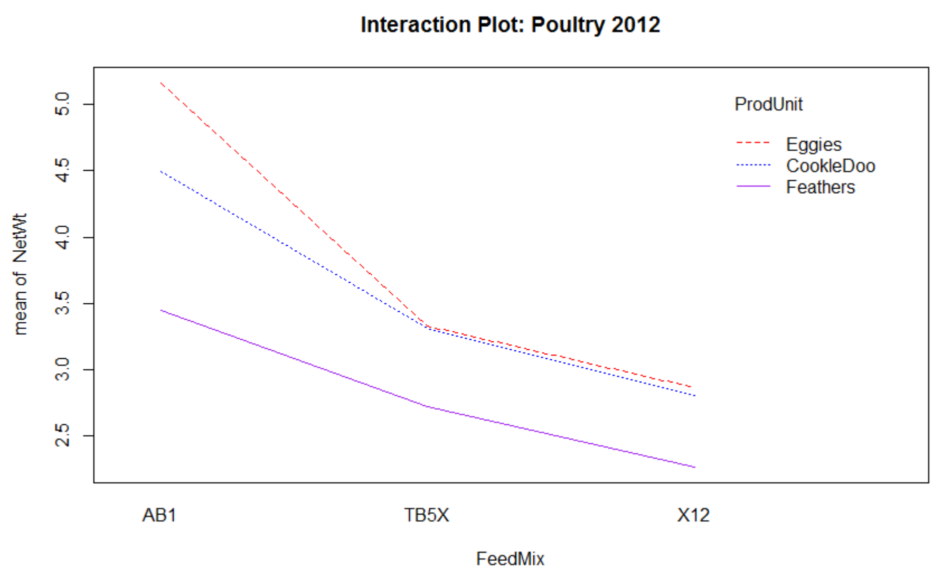
Interaction effect, ProdUnit:FeedMix (Two Way ANOVA)

p-value = 2.2e-16 < $\alpha =$ 0.05 therefore we fail to reject the null hypothesis . There is some significant interaction between ProdUnit and FeedMix when trying to predict the response NetWt.

Since the interaction effect is statistically significant the correct step would be to rerun the ANOVA with main effects and then determine whether each main effect is statistically significant. We cannot isolate only the interaction of ProdUnit or FeedMix alone. This is explained by the hierarchy principle, which states that if a interaction term is significant in a model then the main effects must also be included in the model.

From the table we can see that the data is a balanced design.

Figure 15 Interaction Plot Poultry 2012



Interpretation
We see this as we know that an interaction effect tells us that the effect of one factor depends on the movement of the other factor and it's shown by the lines in our plot not running parallel.
The effect of Feedmix interacts with ProdUnit when predicting NetWt.

The null hypothesis of the interaction term is that there is no significant interaction. The above plot supports our decision to reject the null hypothesis.

$H_0: (\alpha\beta)_{11} = (\alpha\beta)_{12} = \ldots\ldots = (\alpha\beta)_{33}$
H_A: *There exists an Interaction*

Interpretation:

We reject the null hypothesis there exists an interaction.
The interaction term ProdUnit:FeedMix is significant p = 2.2e-16 $< \alpha = 0.05$.
There exists an interaction when using ProdUnit and FeedMix to predict response NetWt

Interaction effect, ProdUnit:FeedMix (Two Way ANOVA)

p-value = 2.2e-16 < $\alpha =$ 0.05 therefore we fail to reject the null hypothesis . There is some significant interaction between ProdUnit and FeedMix when trying to predict the response NetWt.

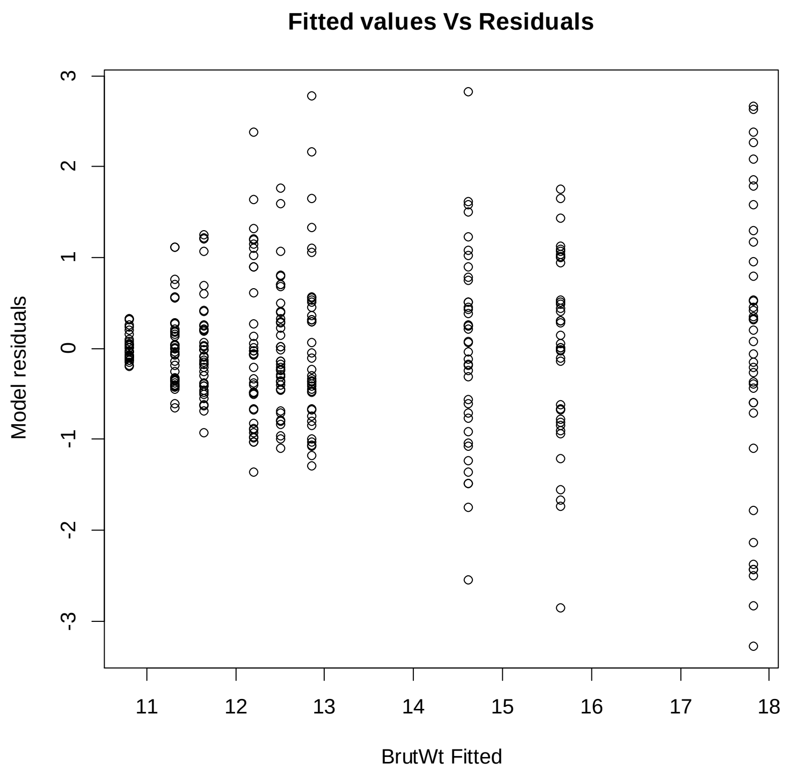
Since the interaction effect is statistically significant the correct step would be to rerun the ANOVA with main effects and then determine whether each main effect is statistically significant. We cannot isolate only the interaction of ProdUnit or FeedMix alone. This is explained by the hierarchy principle, which states that if a interaction term is significant in a model then the main effects must also be included in the model.

From the table we can see that the data is a balanced design.

Residuals Vs Fitted Values

```
> plot(x=BrutWt.fit$fitted  ,y=BrutWt.fit$residuals, xlab = 'BrutWt
Fitted', ylab = 'Model residuals', main = "Fitted values Vs Residuals")
```

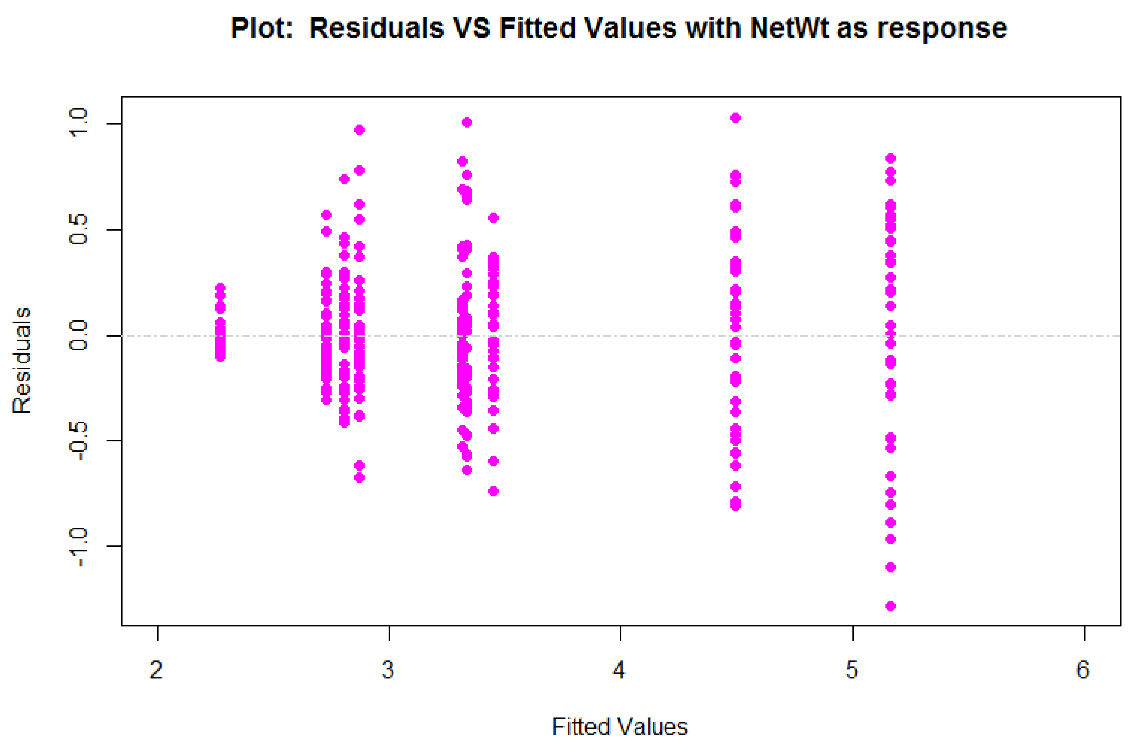
Figure 16: BrutWt response



Interpretation
We see a bit of a funnel shape indicating heteroscedasticity in residuals vs fitted values of BrutWt.

```
function ()

{
# Function plots residuals on Y axis and fitted valeus on x axis of
linear model
# in the Poultry2012 data
# Function takes in not arguments explicitly

#Creating a linear model with NetWt as response
two.way.anova <- lm(NetWt ~ ProdUnit + FeedMix + ProdUnit:FeedMix, data =
Poultry2012)

#We get the residuals and the fitted values
fittedvalues <- two.way.anova$fitted
resvalues <- two.way.anova$residuals

#Plot residuals on Y axis and fitted valeus on x axis
plot(y=resvalues,x=fittedvalues,xlab="Fitted Values",ylab="Residuals",
xlim=c(2,6),
main="Plot:  Residuals VS Fitted Values with NetWt as
response",pch=16,col=6)
# Add line at y=0 to show where 0 lies
abline(h=0,col="light grey",lty=4)
}
```

Figure 17: NettWt response



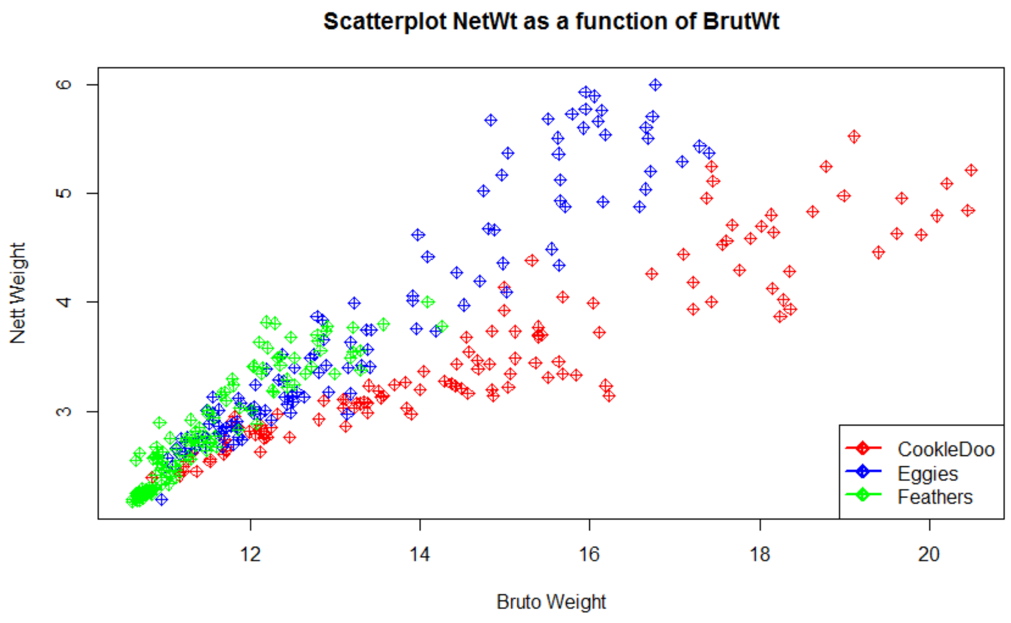
Interpretation
We see a funnel shape which is a sign of **heteroscedasticity** (Unequal variances).

Section 12.3

Section 12.3.a

```
function ()
{
attach(Poultry.data)
on.exit(detach(Poultry.data))

with(Poultry.data,plot(x=BrutWt, y=NetWt))

dev.new()

with(Poultry2012,plot(x=BrutWt, y=NetWt))

#########################################################
graphics.off()

cols <- c("red","blue","green")
cnt <- 0
with(Poultry2012,plot(x=BrutWt, y=NetWt, ylim=range(NetWt), type="n",
xlab="Bruto Weight", ylab="Nett Weight"))
for(i in levels(Poultry2012$ProdUnit))

{cnt <- cnt + 1

points(x=Poultry2012$BrutWt[Poultry2012$ProdUnit==i],
y=Poultry2012$NetWt[Poultry2012$ProdUnit==i],pch=9,col=cols[cnt])
}

legend("bottomright", legend=levels(Poultry2012$ProdUnit),pch=9, col
=cols,lwd=2,lty=rep(1,3))
title( main ="Scatterplot NetWt as a function of BrutWt")
}
```

Figure 19 : Scatterplot NetWt as a function of BrutWt



```
We see a positive relationship between Nett Weight and Brut Weight for
each class of ProdUnit. (Note Prod Unit is a categorical variable with
three levels namely Feathers, CookleDoo and Eggies. As each categories
Nett Weight rises, its Brute Weight rises.

We also see that one category mostly has low values of both Nett Weight
rises, Brute Weight. This is in contrast to the other two which have a
much larger range in values.

Overall we find that eggies has a differenct intraction (slope) between
the NettWeight and Brut Weight in comparison to CookieDoo.
```

Section 12.3. b

The Code is in the function in 12.3.c

```
> Net.one.way = lm(NetWt ~ ProdUnit)
> anova(Net.one.way)
Analysis of Variance Table
```

```
Response: NetWt
           Df Sum Sq Mean Sq F value    Pr(>F)
ProdUnit    2  78.42  39.210  79.481 < 2.2e-16 ***
Residuals 594 293.04   0.493
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretation
Production Unit is significant p-value = 2.2e-16 < $\alpha = 0.05$.

```
$`Ex.12.3.1.c`

Call:
lm(formula = NetWt ~ ProdUnit + BrutWt, data = Poultry2012)

Residuals:
     Min       1Q   Median       3Q      Max
-0.96806 -0.26650 -0.03603  0.25686  1.32985

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      -2.23248    0.15789 -14.139   <2e-16 ***
ProdUnitEggies    0.82473    0.05098  16.178   <2e-16 ***
ProdUnitFeathers  0.57325    0.05953   9.629   <2e-16 ***
BrutWt            0.38781    0.01036  37.430   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.375 on 354 degrees of freedom
Multiple R-squared:  0.8372,  Adjusted R-squared:  0.8359
F-statistic:   607 on 3 and 354 DF,  p-value: < 2.2e-16
```

Interpretation
All three production units are significant with p-values < 2e-16 .
Ovarall the moodel is significant with p-value = 2.2e-16< 0.05.

Section 12.3. c

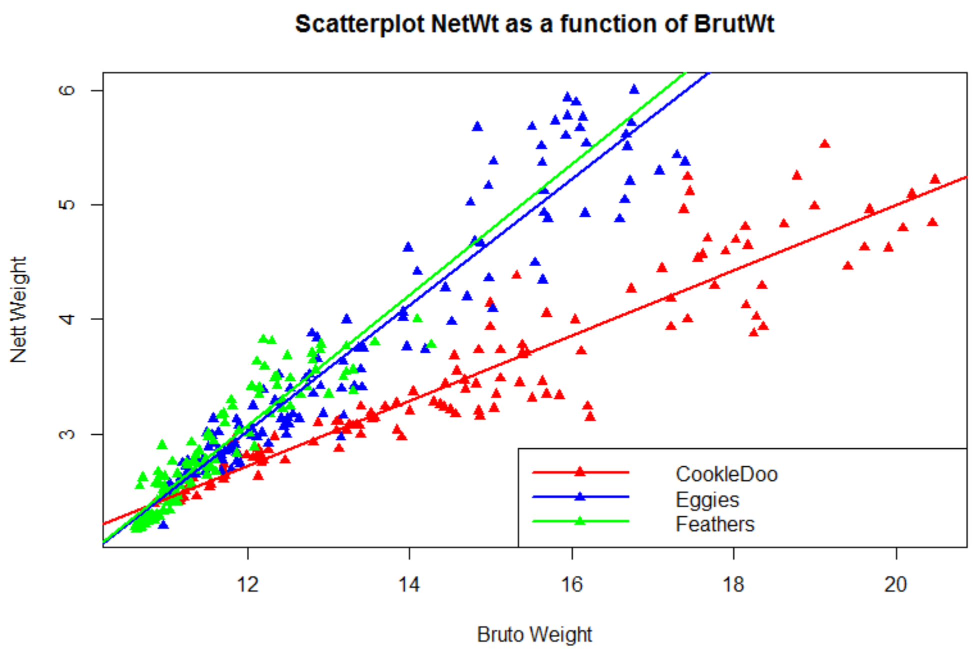
```
> fix(Ex.12.3.b_c)
```

```
function ()
{
with(Poultry.data,plot(x=BrutWt, y=NetWt))

dev.new()
#####################################################
graphics.off()
cols <- c("red","blue","green")
cnt <- 0
with(Poultry2012,plot(x=BrutWt, y=NetWt, ylim=range(NetWt), type="n",
xlab="Bruto Weight", ylab="Nett Weight"))

for(i in levels(Poultry2012$ProdUnit))
{
cnt <- cnt + 1

points(x=Poultry2012$BrutWt[Poultry2012$ProdUnit==i],
y=Poultry2012$NetWt[Poultry2012$ProdUnit==i],pch=17,col=cols[cnt])
}

legend("bottomright", legend=levels(Poultry2012$ProdUnit),pch=17, col
=cols,lwd=2,lty=rep(1,3))
title( main ="Scatterplot NetWt as a function of BrutWt")
#######################################################
# One way Ancova with interaction effect
one.way.ancova.I <- lm(NetWt ~ ProdUnit + BrutWt + BrutWt:ProdUnit,
data=Poultry2012)
temp1 <- anova(one.way.ancova.I)
temp2 <- coefficients(one.way.ancova.I)

# One Way Anova only showing the main effects ProdUnit and BrutWt
one.way <- lm(NetWt ~ ProdUnit + BrutWt , data=Poultry2012)
temp3 <- summary(one.way)
###Seperate regression lines for plot
one.way.ancova.sep.reglns <- lm(NetWt ~ -1 + ProdUnit/BrutWt,
data=Poultry2012)
coeffs <- coefficients(one.way.ancova.sep.reglns)
abline(a=coeffs[1],b=coeffs[4],col="red",lwd=2)
abline(a=coeffs[2],b=coeffs[5],col="blue",lwd=2)
abline(a=coeffs[3],b=coeffs[6],col="green",lwd=2)
# list answers
list(  Ex.12.3.1.c = temp3,  Ex.12.3.1.d._one_way_ancova = temp1,
Ex.12.3.1.d.coefficients=temp2)
}
```

```
> Ex.12.3.b_c()
```

12.13.1 i)

Figure 20 : Scatterplot NetWt Vs BrutWt



Interpretation

```
$`Ex.12.3.1.c`

Call:
lm(formula = NetWt ~ ProdUnit + BrutWt, data = Poultry2012)

Residuals:
     Min       1Q   Median       3Q      Max
-0.96806 -0.26650 -0.03603  0.25686  1.32985

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      -2.23248    0.15789 -14.139   <2e-16 ***
ProdUnitEggies    0.82473    0.05098  16.178   <2e-16 ***
ProdUnitFeathers  0.57325    0.05953   9.629   <2e-16 ***
BrutWt            0.38781    0.01036  37.430   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.375 on 354 degrees of freedom
Multiple R-squared:  0.8372,	Adjusted R-squared:  0.8359
F-statistic:   607 on 3 and 354 DF,  p-value: < 2.2e-16


$Ex.12.3.1.d._one_way_ancova
Analysis of Variance Table

Response: NetWt
                 Df  Sum Sq Mean Sq F value    Pr(>F)
ProdUnit          2  59.077  29.538  382.94 < 2.2e-16 ***
BrutWt            1 197.038 197.038 2554.39 < 2.2e-16 ***
ProdUnit:BrutWt   2  22.636  11.318  146.72 < 2.2e-16 ***
```

```
Residuals       352  27.152   0.077
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Interpretation (Ex.12.3.1.d._one_way_ancova) :

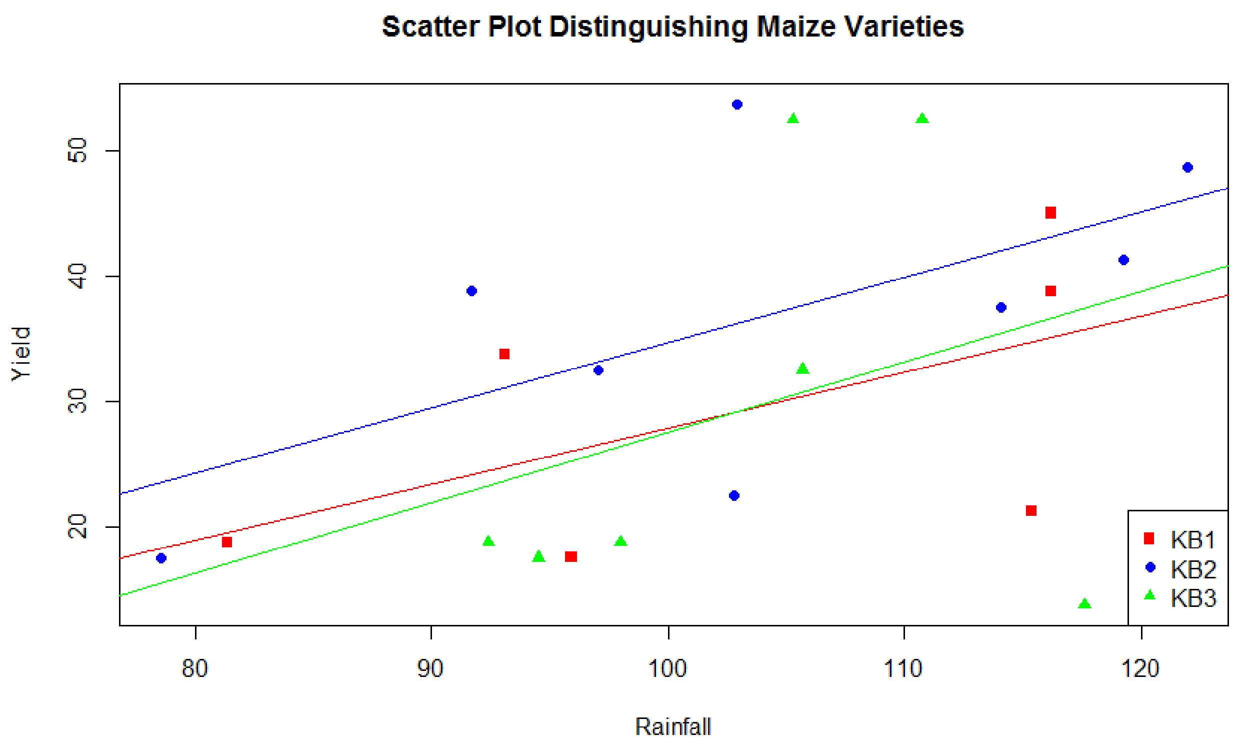
We see a significant interaction between Production Unit and BrutWt when predicting response NetWt.

Section 12.4

```
> Table.12.4.1
Variety Yield Rainfall
1      KB1 17.50    95.90
2      KB1 45.00   116.20
3      KB1 38.75   116.20
4      KB1 33.75    93.10
5      KB1 18.75    81.34
6      KB1 21.25   115.36
7      KB1    NA       NA
8      KB1    NA       NA
9      KB2 37.50   114.10
10     KB2 41.25   119.28
11     KB2 48.75   121.94
12     KB2 32.50    97.02
13     KB2 53.75   102.90
14     KB2 38.75    91.70
15     KB2 22.50   102.76
16     KB2 17.50    78.54
17     KB3 18.75    98.00
18     KB3 13.75   117.60
19     KB3 52.50   105.28
20     KB3 17.50    94.50
21     KB3 18.75    92.40
22     KB3 52.50   110.74
23     KB3 32.50   105.70
24     KB3    NA       NA
```

```
function (data=Table.12.4.1)
{
attach(Table.12.4.1)
on.exit(detach(Table.12.4.1))

with(data, plot(x=Rainfall, y= Yield, ty= "n", main = "Scatter Plot
Distinguishing Maize Varieties"))

with(data, points(data[data[,1]=="KB1" ,3:2],pch=15, col ="red"))

with(data, abline(lm(Yield ~ Rainfall, data =
data[data[,1]=="KB1",]),col="red"))

with(data, points(data[data[,1]=="KB2",3:2],pch=16, col ="blue"))

with(data, abline(lm(Yield ~ Rainfall, data
=data[data[,1]=="KB2",]),col="blue"))

with(data, points(data[data[,1]=="KB3",3:2],pch=17, col ="green"))

with(data, abline(lm(Yield ~ Rainfall, data
=data[data[,1]=="KB3",]),col="green"))

out1 <- lm(Yield ~ Variety + Rainfall + Rainfall:Variety, data=data)

out2 <- lm(Yield ~ Rainfall + Variety, data=data)

list(summary(out1), anova(out1), summary(out2), anova(out2))

legend("bottomright" ,pch=c(15,16,17),legend=c("KB1","KB2","KB3"),col =
c("red", "blue","green"))}
```

Figure 21 : Scatter Plot Maize Varieties



```
> fix(Ex.12.4)


> Ex.12.4(Table.12.4.1)
[[1]]
Call:
lm(formula = Yield ~ Variety + Rainfall + Rainfall:Variety, data = data)

Residuals:
     Min       1Q   Median       3Q      Max 
-23.6889  -6.9111  -0.7152   8.3521  22.0093 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         -16.93880   41.05965  -0.413    0.686
VarietyKB2           -0.44784   54.31783  -0.008    0.994
VarietyKB3          -11.94558   74.14820  -0.161    0.874
Rainfall              0.44755    0.39512   1.133    0.275
VarietyKB2:Rainfall   0.07354    0.52159   0.141    0.890
VarietyKB3:Rainfall   0.11642    0.71409   0.163    0.873

Residual standard error: 13.22 on 15 degrees of freedom
  (3 observations deleted due to missingness)
Multiple R-squared:  0.2861,	Adjusted R-squared:  0.04808 
F-statistic: 1.202 on 5 and 15 DF,  p-value: 0.3549

[[2]]
Analysis of Variance Table

Response: Yield
                 Df  Sum Sq Mean Sq F value  Pr(>F)
```

```
Variety           2  259.56  129.78  0.7430 0.49240
Rainfall          1  784.45  784.45  4.4911 0.05117 .
Variety:Rainfall  2    5.76    2.88  0.0165 0.98366
Residuals        15 2620.01  174.67
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

[[3]]

Call:
lm(formula = Yield ~ Rainfall + Variety, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-22.8055  -7.4708  -0.7977   8.1203  22.1230

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -22.49647   23.47933  -0.958   0.3514
Rainfall      0.50150    0.22253   2.254   0.0377 *
VarietyKB2    7.13840    6.71291   1.063   0.3025
VarietyKB3    0.07529    6.91505   0.011   0.9914
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.43 on 17 degrees of freedom
  (3 observations deleted due to missingness)
Multiple R-squared:  0.2845,      Adjusted R-squared:  0.1582
F-statistic: 2.253 on 3 and 17 DF,  p-value: 0.1192
[[4]]
Analysis of Variance Table

Response: Yield
          Df  Sum Sq Mean Sq F value  Pr(>F)
Rainfall   1  794.53  794.53  5.1440 0.03665 *
Variety    2  249.49  124.74  0.8076 0.46231
Residuals 17 2625.78  154.46
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Theory:
First hypothesis test is

$$H_0: \beta_1 = \beta_2 = \beta_3$$

If we do not reject the Null hypothesis and conclude that all slopes are the same then the following hypothesis is tested.

$$H_0: \beta = 0$$

This is to test if the common slope, β, is equal to zero.

If this H_0 is rejected and the true slope of the common β is not equal to zero then a new model will be fit with Rainfall and Variety as predictors but excluding the interaction effect.

If we fail to reject H_0 and conclude that β is equal to zero then anova will be performed on model in the second step.

```
maize.data <- read.table('clipboard', header = T)

> ancova1  <- lm(Yield ~ Rainfall + Variety + Rainfall:Variety)
> anova(ancova1)
Analysis of Variance Table

Response: Yield
                 Df  Sum Sq Mean Sq F value  Pr(>F)
Rainfall          1  794.53  794.53  4.5488 0.04987 *
Variety           2  249.49  124.74  0.7142 0.50551
Rainfall:Variety  2    5.76    2.88  0.0165 0.98366
Residuals        15 2620.01  174.67
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

$$H_0: \beta_1 = \beta_2 = \beta_3$$
$$H_A: At\ least\ two\ \beta_j's\ are\ different$$

Rainfall:Variety p-value = **0.98366 > $\alpha = 0.05$. We do not reject the null Hypothesis the slopes must be the same.**

Coefficients for the model

$$Y_{ij} = \mu + \alpha_i + \beta_{ij}x_{ij} + \epsilon \quad i = 1,2, \ldots k; j = 1,2, \ldots, n_i$$

```
> ancova.regions <- lm(Yield ~ -1 + Rainfall/Variety)
> coefficients(ancova.regions)
          Rainfall Rainfall:VarietyKB1 Rainfall:VarietyKB2 Rainfall:VarietyKB3
      0.2866166015       -0.0006516299        0.0694415287                  NA
```

Variable	Estimated coefficient
Rainfall	0.2866166015
Rainfall: Variety KB1	-0.0006516299
Rainfall: Variety KB2	0.0694415287
Rainfall: Variety KB3	(This is the baseline)

Estimated Coefficients are low suggesting low interaction.

```
> lm.3 <- lm(Yield ~ Rainfall + Variety)
> anova(lm.3)
Analysis of Variance Table

Response: Yield
          Df  Sum Sq Mean Sq F value  Pr(>F)
Rainfall   1  794.53  794.53  5.1440 0.03665 *
Variety    2  249.49  124.74  0.8076 0.46231
Residuals 17 2625.78  154.46
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

$$H_0: \beta = 0$$
$$H_0: \beta \neq 0$$

When predicting Yield the variety seems to be insignificant according to the above, output. It seems that rainfall is significant rather.

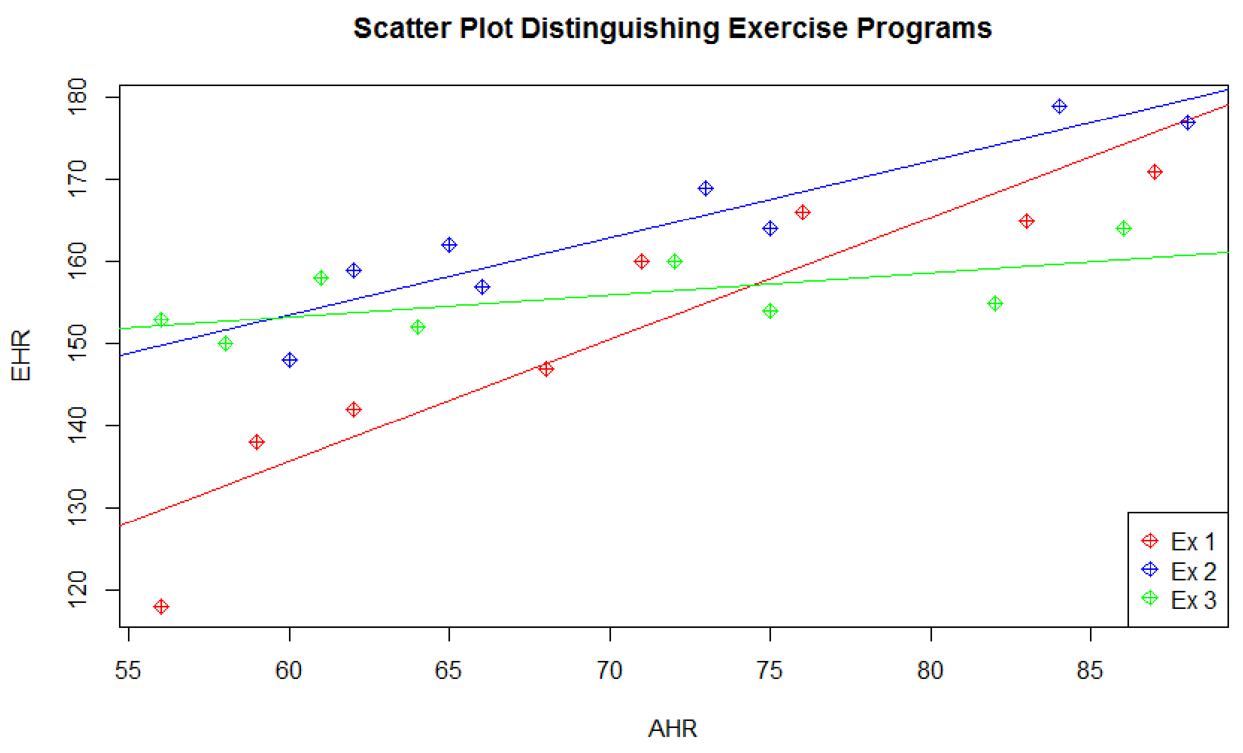
```
> coefficients(lm.3)
 (Intercept)     Rainfall   VarietyKB2   VarietyKB3
-22.49646919   0.50150269   7.13839528   0.07528619
```

Section 12.5

```
>fix(Ex.12.5)
> Ex.12.5(Table.12.5.1)
```

```
function (data=Table.12.5.1)
{

with(data, plot(x=AHR, y= EHR, ty= "n", main = "Scatter Plot Distinguishing
Exercise Programs"))

with(data, points(data[data[,1]=="EX1" ,3:2],pch=9, col ="red"))

with(data, abline(lm(EHR ~ AHR, data = data[data[,1]=="EX1",]),col="red"))

with(data, points(data[data[,1]=="EX2",3:2],pch=9, col ="blue"))

with(data, abline(lm(EHR ~ AHR, data =data[data[,1]=="EX2",]),col="blue"))

with(data, points(data[data[,1]=="EX3",3:2],pch=9, col ="green"))

with(data, abline(lm(EHR ~ AHR, data =data[data[,1]=="EX3",]),col="green"))

out1 <- lm(EHR ~ ExerciseProgram + AHR + AHR:ExerciseProgram, data=data)

out2 <- lm(EHR ~ AHR + ExerciseProgram, data=data)

list(summary(out1), anova(out1), summary(out2), anova(out2))

legend(“bottomright”,pch=c(9,9,9),legend=c("Ex 1","Ex 2","Ex 3"),col
= c("red", "blue","green"))


}
```

Figure 22: Scatter Plot Exercise Programs



Interpretation:

We see that each exercise has a different (interaction/slope) effect on HER as AHR rises. We However we see a positive effect overall for each exercise.

```
> Ex.12.5(Table.12.5.1)

[[1]]

Call:
lm(formula = EHR ~ ExerciseProgram + AHR + AHR:ExerciseProgram,
    data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-11.7106  -3.3351   0.1057   3.6979   8.0111

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)             46.5384    12.7191   3.659 0.001796 **
ExerciseProgramEX2      50.6525    19.0101   2.665 0.015797 *
ExerciseProgramEX3      90.9466    17.8530   5.094 7.58e-05 ***
AHR                      1.4852     0.1791   8.294 1.46e-07 ***
ExerciseProgramEX2:AHR  -0.5472     0.2651  -2.064 0.053732 .
ExerciseProgramEX3:AHR  -1.2215     0.2531  -4.826 0.000135 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.311 on 18 degrees of freedom
Multiple R-squared:   0.87,Adjusted R-squared:  0.8339
F-statistic:  24.1 on 5 and 18 DF,  p-value: 2.166e-07
```

```
[[2]]
Analysis of Variance Table

Response: EHR
                    Df  Sum Sq Mean Sq F value    Pr(>F)    
ExerciseProgram      2  747.75  373.87  13.257 0.0002891 ***
AHR                  1 1991.62 1991.62  70.618 1.208e-07 ***
ExerciseProgram:AHR  2  658.98  329.49  11.683 0.0005593 ***
Residuals           18  507.65   28.20                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

$H_0: \beta_1 = \beta_2 = \beta_3$ $\quad$ (*There is no interaction*)
H_A: *At least two* β_j*'s are different* $\quad$ (*There is an interaction*)

p-value=**0.0005593, we reject null hypothesis and conclude that there is significant interaction. Hence we estimate the coefficients below.**

$$Y_{ij} = \mu + \alpha_i + \beta_{ij}x_{ij} + \epsilon \quad i = 1,2,\dots k; j = 1,2,\dots,n_i$$

We fit the model above and estimate the coefficients below.

```
# Note not part of function following lines were fit separately, after
#data was reimported into R
>coefficients(one.way.ancova.1)
(Intercept) AHR ProgEx2 ProgEx3 AHR:ProgEx2 AHR:ProgEx3
46.5383741 1.4852189 50.6525458 90.9465947 -0.5472212 -1.2214639

> Sep.regions <- lm(EHR ~ -1 + AHR/Prog)
> coefficients(Sep.regions)
        AHR AHR:ProgEX1 AHR:ProgEX2 AHR:ProgEX3
 2.20450492 -0.07125417  0.06646854          NA
```

Variable	**Estimated coefficient**
AHR	2.20450492
AHR: Programme EX1	-0.07125417
AHR: Programme EX2	0.06646854
AHR: Programme EX3	Baseline

Difference between the estimated coefficient of AHR and the coefficients of AHR:Prog EX1 and EX2. This reinforces our conclusion of rejecting the null hypothesis.

$$H_0: \beta_1 = \beta_2 = \beta_3$$

Program 3 is the baseline hence no value is outputted.

```
[[3]]

Call:
lm(formula = EHR ~ AHR + ExerciseProgram, data = data)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-20.153  -2.971   1.366   3.756   9.992 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)         88.1592    11.0676   7.966 1.25e-07 ***
AHR                  0.8928     0.1528   5.843 1.02e-05 ***
ExerciseProgramEX2  12.2725     3.8245   3.209  0.00441 ** 
ExerciseProgramEX3   5.7678     3.8218   1.509  0.14689    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 7.638 on 20 degrees of freedom
Multiple R-squared:  0.7013,     Adjusted R-squared:  0.6565 
F-statistic: 15.65 on 3 and 20 DF,  p-value: 1.784e-05


[[4]]
Analysis of Variance Table

Response: EHR
                Df  Sum Sq Mean Sq F value   Pr(>F)    
AHR              1 2138.18 2138.18 36.6556 6.43e-06 ***
ExerciseProgram  2  601.19  300.59  5.1532  0.01567 *  
Residuals       20 1166.63   58.33                      
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 
```

Section 13.3 (b)

```
# Class example modified first , then changed to include calculations in
# second function which is the answer
```

```
>fix(13.3.b)
```

```
function (init1 =2, init2 =2,sample.size=100)
{
###Generation of sample data
set.seed(326396)
bdata <- rbeta(sample.size,4,2)

optim(par=c(init1,init2),fn= function(parvec, data=bdata)
{-sum(log(dbeta(x=data, shape1=parvec[1], shape2=parvec[2])))})

###Note the following in the call to optim()
### 1. First argument contains the initial values
###    for the two parameters to be estimated
### 2. First argument of function fn in optim
###    is a vector representing the the parameters to be estimated
### 3. Second argument of function fn in optimum
###    contains the sample data for evaluating the function
###    to be minimised.
}
```

```
> Ex.13.3.b()   # Initial Sample Size 100
$`par`
[1] 3.473381 1.585293
$value
[1] -33.62722
$counts
function gradient
      65       NA
$convergence
[1] 0
$message
NULL
```

```
> Ex.13.3.b(sam=100)  # Sample Size 100
$`par`
[1] 3.473381 1.585293

$value
[1] -33.62722

$counts
function gradient
      65       NA

$convergence
[1] 0

$message
NULL
```

```
> Ex.13.3.b(sam=10000) # Sample Size 10 000
$`par`
[1] 3.931074 1.988288

$value
[1] -3546.106

$counts
function gradient
      73       NA

$convergence
```

```
[1] 0

$message
NULL
```

Note as we increase the sample we get better results. If we use the same estimates and increase the sample size. Then it should be less variable and we get a better result. The stabilizer is much better.

As the sample size increases the parameter estimates attained are closer to their actual values.

```
>fix(function1) # Ex 13.3 b Answer

function(n, a, b){
# This function investigates the use of function optim and finding
# the maximum of likelihood function as well as the maximum likelihood estimates
# for parameters in case of beta(4,2) distribution.
# n : sample size
# a : initial estimate for parameters alpha and beta
# b : initial estimate for parameter beta
set.seed(12345)

sample.var <- rbeta( n , 4 , 2)

#max of likelihood function of b is max of log-likelihood
#minimizing the log-likelihood is the same as maximising the negative log-likeli
hood
estimate <- optim( par=c(a, b), fn = function( par , data=sample.var ) { -sum( l
og( dbeta(x=data, par[1], par[2]) ) )})

a <- estimate$par
b <- estimate$value
list1 <- list(parameter_estimates=a,maximum_likelihood=b)

return(list1)
}
```

# Change in sample size

```
> function1(n=100,a=2,b=4)
$`parameter_estimates`
[1] 3.206550 1.744424

$maximum_likelihood
[1] -27.6269

> function1(n=1000,a=2,b=4)
```

```
$`parameter_estimates`
[1] 4.162208 2.086381

$maximum_likelihood
[1] -375.8531

> function1(n=10000,a=2,b=4)
$`parameter_estimates`
[1] 3.977062 1.990553

$maximum_likelihood
[1] -3603.243
```

# Change in initial estimates

```
> function1(n=10000,a=4,b=6)
$`parameter_estimates`
[1] 3.976578 1.990430

$maximum_likelihood
[1] -3603.242

> function1(n=10000,a=5,b=7)
$`parameter_estimates`
[1] 3.977190 1.990779

$maximum_likelihood
[1] -3603.243
> function1(n=1000,a=4.5,b=2.5)
$`parameter_estimates`
[1] 4.162993 2.086911

$maximum_likelihood
[1] -375.8532
```

While the sample size increases the parameter estimates attained are closer to their actual values.

While the initial parameter estimates move closer to the actual parameter values the functions estimates of the parameter values also move to the actual values.

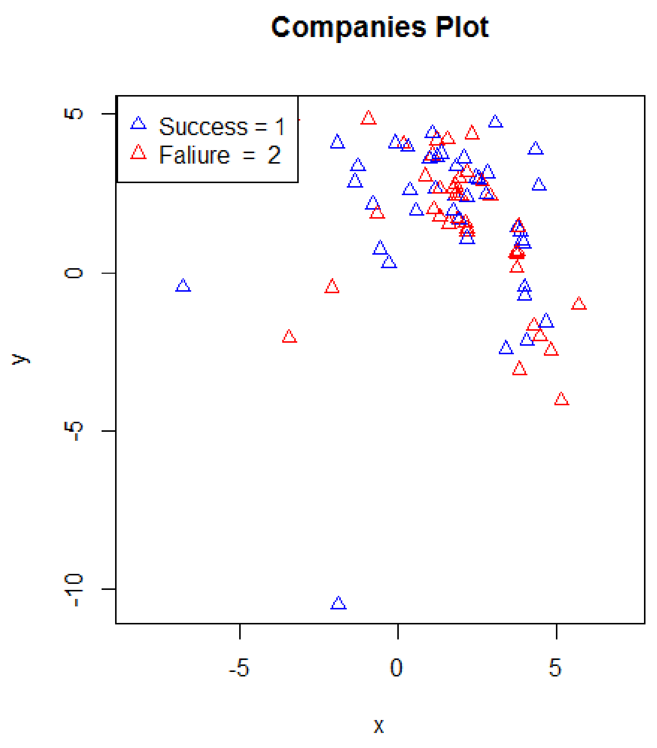
Section 13.3 (c)

```
>fix(Ex.13.3.c)
```

```
function (company=company.10var)
{
# Remove success or faliure column 1

#Creating the distance matrix
Dist_matrix <- dist(scale(company))

#Generating random numbers from a uniform distribution to begin with
start.data <- runif(79*2)

#Making the distance matrix a vector
Vec_Dist_matrix <- as.vector(Dist_matrix)

#for loop using estimated paramters as new values to begin with
for(i in 1:400){
param <- optim(par=start.data,fn=function(x){
x <- matrix(x, ncol=2,byrow=T)
dis <- as.vector(dist(x))
(1/sum(Vec_Dist_matrix)) * sum(((dis-Vec_Dist_matrix)^2)/Vec_Dist_matrix)
})

start.data <- param$par
}
#distance matrix to make the 79 dimensions 2 dimensions
dist.mat <- matrix(param$par,ncol=2,byrow=T)
par(pty="s")
#2-D plot
plot(dist.mat[,1],dist.mat[,2],col=c("blue","red"),ylab="y",xlab="x",pch=
2,
main="Companies Plot",asp=1)

#Adding the legend
legend("topleft",legend=c("Success = 1","Faliure  =
2"),pch=2,col=c("blue","red"))
list(x=dist.mat,stress=param$value)
}
```

```
> Ex.13.3.c()
```

Figure 23: Company Plot



Interpretation:

We can not see a large difference between the successful companies. However in some iterations we do see a difference between the successful and the failed companies in the y space but not in the x space.

iv)

Number of dimensions need to represent the exact distance between the 79 companies is 78 (79-1 dimensions).

*****