Statistical Analysis Using R Unit-2

Mohammad Wasiq , GL0427, 18STB-028

9/29/2020

Unit - II

Generate automated reports, giving detailed description of statistics,correlation and lines of regression.

Defining new function — ' function()'

The structure is of the type:

$$name = function(argument1, argument2, ...)expression$$

Example1:Define a function for coefficient of variation 'cv'

```
cv=function(x) sd(x)/mean(x)*100
dump("cv", file="cv.txt")
source("cv.txt")
# create a data vector of height and compute its cv
height=c(168,153,154,170)
cv(x=height)
```

```
## [1] 5.578524
```

Example 2: Define a function for standard error of mean—'sem()'

```
# define standard error of mean
sem=function(x) sd(x)/sqrt(length(x))
dump("sem",file="sem.txt" )
source("sem.txt")
sem(height)
```

```
## [1] 4.497685
```

```
sem(trees$Girth)
```

```
## [1] 0.5636263
```

```
sem(trees$Volume)
```

```
## [1] 2.952324
```

Descriptive statistics of a data vector—'describe()'

Define a function which takes a data vector as input and returns mean, sd, sem and cv.

```r
describe=function(x) {
# x is a data vector
n=length(x)
m=mean(x)
s=sd(x)
sem=s/sqrt(n)
cv=s/m*100
out=list(Mean=m,SD=s,SEM=sem,CV=cv)
return(out)
}
dump("describe","describe.txt")
describe(height)
```

```
## $Mean
## [1] 161.25
##
## $SD
## [1] 8.995369
##
## $SEM
## [1] 4.497685
##
## $CV
## [1] 5.578524
```

```r
OUT1=describe(height)
OUT1$Mean
```

```
## [1] 161.25
```

Suppose you define a new vector of weight as 'weight' and need to compute its descriptive statistics using 'describe()'

```r
weight=c(55,60,65,78,80,64,63)
describe(weight)
```

```
## $Mean
## [1] 66.42857
##
## $SD
## [1] 9.216962
##
## $SEM
## [1] 3.483684
##
## $CV
## [1] 13.875
```

```r
describe(trees$Volume)
```

```
## $Mean
## [1] 30.17097
##
## $SD
## [1] 16.43785
```

```
##
## $SEM
## [1] 2.952324
##
## $CV
## [1] 54.48233
```

Define 'describe()' which can handle missing values—'na.omit()'

Add the feature for handling missing values. Note that the command is 'na.omit()'. NOw we are going to redefine describe with names 'describe_na'

```r
# a function which handles missing values
describe_na=function(x) {
# x is a data vector
x=na.omit(x)
n=length(x)
m=mean(x)
s=sd(x)
sem=s/sqrt(n)
cv=s/m*100
out=list(Mean=m,SD=s,SEM=sem,CV=cv)
return(out)
}
dump("describe_na",file="describe_na.txt")
y1=c(12,15,16,NA,20,25,30)
describe(x=y1)
```

```
## $Mean
## [1] NA
##
## $SD
## [1] NA
##
## $SEM
## [1] NA
##
## $CV
## [1] NA
```

```r
describe_na(x=y1)
```

```
## $Mean
## [1] 19.66667
##
## $SD
## [1] 6.772493
##
## $SEM
## [1] 2.764859
##
## $CV
## [1] 34.43641
```

Extend the function 'describe()' for data frame

We shall extend the definition of 'describe()' when input is a data frame and not a vector. Moreover, it will return the data frame as output and not as list. First we will create a dataframe which will be used as argument in the function names as 'describeDF()'.

```
# Create a data frame of height and weight of 5 students of B.Sc.
# Vth Semester
height=c(166,170,165,165,166)
weight=c(55,60,58,59,64)
heightWeight=data.frame(Height=height,Weight=weight)
heightWeight
```

```
##   Height Weight
## 1    166     55
## 2    170     60
## 3    165     58
## 4    165     59
## 5    166     64
```

```
# define the function 'describeDF()' which will take data frame as input and return data frame as outpu
describeDF=function(x) {
# x is a data frame , not a vector
x=data.frame(x)  # to make sure that x is a data frame
n=nrow(x)  # number of rows
m=apply(x,2,mean) # compute mean of each column
s=apply(x, 2,sd)
cv=s/m*100
se=s/sqrt(n)
min=min(x)
max=max(x)
out=(data.frame(Mean=m,SD=s,CV=cv,SEM=se,Max=max,Min=min))
return(out)
dump("describeDF",file="describeDF.txt")  # to save it
source("describeDF.txt")
}
```

Data Frame

Now we shall make use of this function to return the descriptive statistics of a data frame.

```
outDF1=describeDF(x=heightWeight)
outDF1
```

```
##         Mean       SD       CV       SEM Max Min
## Height 166.4 2.073644 1.246180 0.9273618 170  55
## Weight  59.2 3.271085 5.525482 1.4628739 170  55
```

```
write.csv(outDF1,file="outDF1.csv")
```

Use the tree data and find ots summary using 'describeDF()'

```
describeDF(trees)
```

```
##            Mean        SD        CV       SEM Max Min
## Girth  13.24839  3.138139 23.686948 0.5636263  87 8.3
## Height 76.00000  6.371813  8.383964 1.1444114  87 8.3
```

```
## Volume 30.17097 16.437846 54.482331 2.9523244  87 8.3
```

```
out1=describeDF(trees)
round(out1,3)
```

```
##          Mean     SD     CV   SEM Max Min
## Girth  13.248  3.138 23.687 0.564  87 8.3
## Height 76.000  6.372  8.384 1.144  87 8.3
## Volume 30.171 16.438 54.482 2.952  87 8.3
```

Excercise 1:

Modify the function 'describeDF()' by adding columnof 'min()' and 'max()' in the beginning. * Ans: These (min & max) are added in line 91 where the chunk 'describeDF()' starts.

Excercise 2:

- (i) Create data.frame using the function 'fix()' and save it as '.txt'file and analyze that using 'describeDF2()'.
- (ii) Analyze same data using the 'summary()' function of R. Compare the results obtained.

Matrix operations in R

```
# 3x1-4x2=6
# x1+2x2=-3
# Ax=b
# x=A(^-1)b
# solve(A,x)
A=matrix(c(3,1,-4,2),ncol=2)
A
```

```
##      [,1] [,2]
## [1,]    3   -4
## [2,]    1    2
```

```
b=c(6,-3)
x=solve(A,b)
x
```

```
## [1] -2.960595e-16 -1.500000e+00
```

```
solve(A)%*%b
```

```
##               [,1]
## [1,] -2.220446e-16
## [2,] -1.500000e+00
```

```
# Create of a matrix using cbind and rbind function
x1=c(2,4)
x2=c(3,8)
x12=cbind(x1,x2)
x12
```

```
##      x1 x2
## [1,]  2  3
## [2,]  4  8
```

```
xr12=rbind(x1,x2)
xr12
```

```
##    [,1] [,2]
## x1    2    4
## x2    3    8
```

Linear Models with R

$$(y, X\beta, \sigma^2 I)$$

This Gauss Markov set up can be rewritten as:

$$y = X\beta + e$$

This model is termed as general linear model. It is to be noted that y is the vector of responces,X is termed as model matrix, and β is known as vector of regression coefficients. However,σ^2 is known as residual variance, I stands stands for indentity marix of order $n \times n$.

The method of least squares is used to estimate β. This method states that we will choose that value of β which will minimize error sum of squares defined as:

$$errorSS = e^T e = (y - X\beta)^T (y - X\beta)$$

and the result is solution normal equations defined as:

$$(X^T X)\hat{\beta} = X^T y$$

alternatively least squares estimate of β is defined as:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

This implies that variance covariance matrix of $\hat{\beta}$ is :

$$Var(\hat{\beta}) = \sigma^2 (X^T X)^{-1}$$

and its estimate is

$$\widehat{Var(\hat{\beta})} = \hat{\sigma^2} (X^T X)^{-1}$$

The diagonal elements of this matrix are variances and non-diagonals are covariances. Thus standard error of β is

$$SE(\hat{\beta}) = \sqrt{diag(\widehat{Var(\hat{\beta})})}$$

, where

$$\hat{\sigma}^2 = \frac{ResidSS}{n - (p + 1)} = MSresidual$$

where

$$ResidSS = (y - X\hat{\beta})^T (y - X\hat{\beta})$$

Explanation and implementations with R

We will make illustration with 'forbes' data available with the **MASS** package.

```
library(MASS)
data(forbes)
forbes
```

```
##       bp  pres
## 1  194.5 20.79
## 2  194.3 20.79
## 3  197.9 22.40
## 4  198.4 22.67
## 5  199.4 23.15
## 6  199.9 23.35
## 7  200.9 23.89
## 8  201.1 23.99
## 9  201.4 24.02
## 10 201.3 24.01
## 11 203.6 25.14
## 12 204.6 26.57
## 13 209.5 28.49
## 14 208.6 27.76
## 15 210.7 29.04
## 16 211.9 29.88
## 17 212.2 30.06
```

```r
# Step by step commands
y=forbes$bp # response vector
y
```

```
##  [1] 194.5 194.3 197.9 198.4 199.4 199.9 200.9 201.1 201.4 201.3 203.6 204.6
## [13] 209.5 208.6 210.7 211.9 212.2
```

```r
x=forbes$pres # regressor
x
```

```
##  [1] 20.79 20.79 22.40 22.67 23.15 23.35 23.89 23.99 24.02 24.01 25.14 26.57
## [13] 28.49 27.76 29.04 29.88 30.06
```

```r
X=cbind(1,x)  # model matrix
X
```

```
##             x
##  [1,] 1 20.79
##  [2,] 1 20.79
##  [3,] 1 22.40
##  [4,] 1 22.67
##  [5,] 1 23.15
##  [6,] 1 23.35
##  [7,] 1 23.89
##  [8,] 1 23.99
##  [9,] 1 24.02
## [10,] 1 24.01
## [11,] 1 25.14
## [12,] 1 26.57
## [13,] 1 28.49
## [14,] 1 27.76
## [15,] 1 29.04
## [16,] 1 29.88
## [17,] 1 30.06
```

```r
n=nrow(X)
n
```

```
## [1] 17
```

```r
p1=ncol(X)
p1
```

```
## [1] 2
```

```r
XtX=crossprod(X,X) # X^TX
XtX
```

```
##           x
##    17   426
## x 426 10821
```

```r
Xty=crossprod(X,y) # X^Ty
Xty
```

```
##      [,1]
##    3450.2
## x 86735.5
```

```r
beta= solve(XtX,Xty) # $$(\hat{\beta})$$
beta
```

```
##         [,1]
##   155.296483
## x   1.901784
```

```r
resid=y-X%*%beta # residual
resid
```

```
##               [,1]
##  [1,] -0.334562921
##  [2,] -0.534562921
##  [3,]  0.003565605
##  [4,] -0.009915946
##  [5,]  0.077227962
##  [6,]  0.196871257
##  [7,]  0.169908154
##  [8,]  0.179729802
##  [9,]  0.422676296
## [10,]  0.341694131
## [11,]  0.492678749
## [12,] -1.226871690
## [13,]  0.021703944
## [14,]  0.510005916
## [15,]  0.175723006
## [16,] -0.221775155
## [17,] -0.264096189
```

```r
rss= sum(resid^2) # Residual SS
rss
```

```
## [1] 2.957438
```

```r
msresid=rss/(n-p1)
msresid
```

```
## [1] 0.1971625
```

```r
sebeta=sqrt(diag(msresid*solve(XtX)))
tratio=beta/sebeta
pvalue=2*(1-pt(abs(tratio),df=n-p1)) # p-value
# Note that if p value is less than 0,05, we reject H0 for beta.
beta
```

```
##          [,1]
##   155.296483
## x   1.901784
```

```r
sebeta
```

```
##                       x
## 0.92733686 0.03675601
```

```r
tratio
```

```
##         [,1]
##   167.46502
## x  51.74075
```

```r
pvalue
```

```
##   [,1]
##      0
## x    0
```

Regression Coefficient : Change in responce y for unit change in regressor x

Organizing simple linear regression analysis—'slr()'

We can organize all the steps in the form of a function,names 'slr()' as:

```r
slr=function(X,y) {
# A function which returns Simple Regression Analysis
X=cbind(1,x)  # model matrix
n=nrow(X)
p1=ncol(X)
XtX=crossprod(X,X)
Xty=crossprod(X,y)
beta= solve(XtX,Xty)
resid=y-X%*%beta
rss= sum(resid^2)
msresid=rss/(n-p1)
sebeta=sqrt(diag(msresid*solve(XtX)))
tratio=beta/sebeta
pvalue=2*(1-pt(abs(tratio),df=n-p1))
out=data.frame(RegCoeff=beta,SEbeta=sebeta,Tvalue=tratio,pvalue=pvalue)
return(out)
}
dump("slr",file="slr.txt")
source("slr.txt")
require(MASS)  # To load the MASS package as
               # It contains 'forbes' data object.
y=forbes$bp  # To extract y from forbes
x=forbes$pres #To extract X from forbes
# Now we can fit the Model using 'slr()'
```

```r
M1=slr(X=x,y=y )
print(M1)
```

```
##     RegCoeff     SEbeta    Tvalue pvalue
##   155.296483 0.92733686 167.46502      0
## x   1.901784 0.03675601  51.74075      0
```

From above output it is evident that regression coefficient is statistically significant as its corresponding 'pvalue' is less than 0.05 .

Multiple Linear Regression—'mlr()'

The basic difference between simple and multiple regression is that in simple there is only one predictor x, whereas in multiple regression it must be 2 or more. We shall write a function to implement multiple regression analysis with 2 regressors or covariates.

```r
mlr=function(y,x1,x2) {
# Define a function to implement multiple linear regression
# y is the response variable
# x1 is one regressor
# x2 is another regressor
# This function returns Multiple Linear Analysis
X=cbind(1,x1,x2)  # model matrix
n=nrow(X)
p1=ncol(X)
XtX=crossprod(X,X)
Xty=crossprod(X,y)
beta= solve(XtX,Xty)
resid=y-X%*%beta
rss= sum(resid^2)
msresid=rss/(n-p1)
sebeta=sqrt(diag(msresid*solve(XtX)))
tratio=beta/sebeta
pvalue=2*(1-pt(abs(tratio),df=n-p1))
out=data.frame(RegCoeff=beta,SEbeta=sebeta,Tvalue=tratio,pvalue=pvalue)
out=round(out,3) # Round upto 3 digits
return(out)
}
dump("mlr",file="mlr.txt")
## Analyse the data 'trees' using 'volume' as response and
 # 'Girth' and 'Height' as regressors.
data(trees)
names(trees)
```

```
## [1] "Girth"  "Height" "Volume"
```

```r
y=trees$Volume
x1=trees$Girth
x2=trees$Height
M2=mlr(y,x1,x2)
print(M2)
```

```
##    RegCoeff SEbeta Tvalue pvalue
##     -57.988  8.638 -6.713  0.000
## x1    4.708  0.264 17.816  0.000
## x2    0.339  0.130  2.607  0.014
```

Exercises based on above concepts.

Exercise 1: Create a data frame consisting of weight(in kg.),Height(in cm.),and Age(in years) of all the students of B.Sc. (Vth Semester) . Fit a model :-

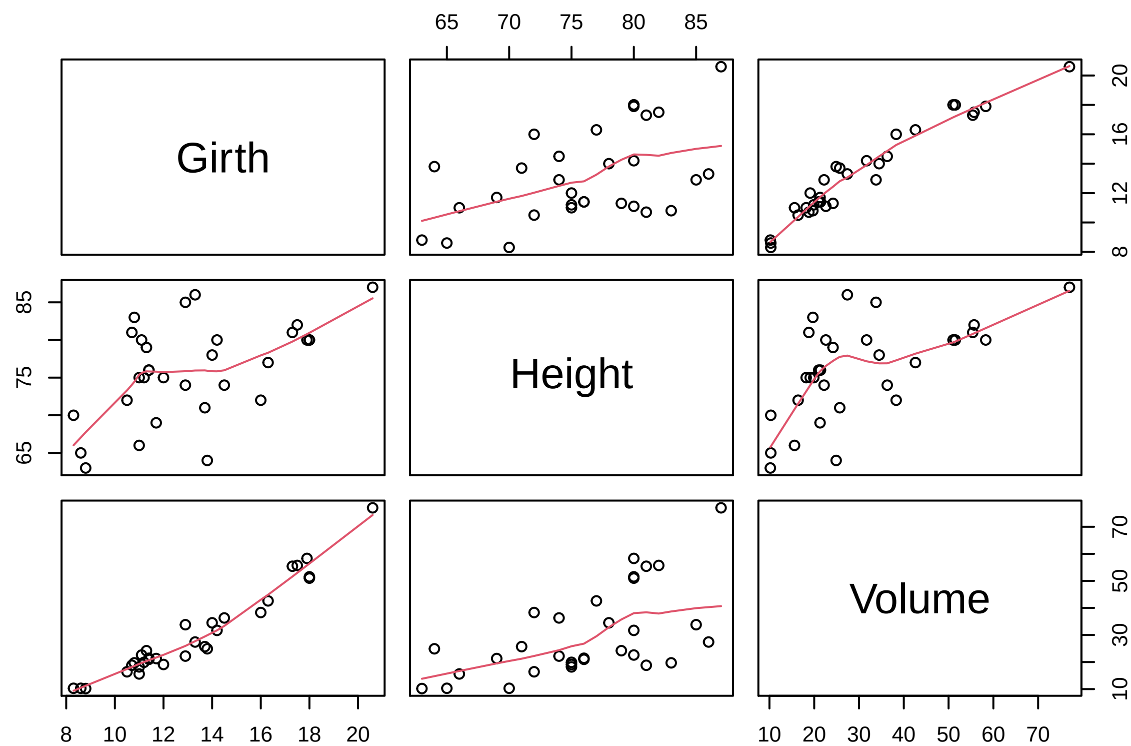
$$Weight = \beta_0 + \beta_1 Height + \beta_2 Age + e$$

and write your comments and conclusions about the analysis.

Scatter plot matrix—'pairs()'

Scatter plot is an extension of scatter diagram for more than two continuous variables. For making this scatter plot matrix, we have the function 'pairs()' in **R**. We shall make use of 'trees' data for the purpose of illustration.

```
pairs(trees,panel=panel.smooth)
```



From above plot it is evident that there is strong linear relationship among the variables of data 'trees'. Using the function 'pairs()' with argument 'panel=panel.smooth()' we seen relationship among pairing variables. However, we want a linear relationship among the pairs. For this purpose we need a function which will plot fitted regression line in each panel. It will be define as:

```
# Define a function pan1 which will implement these ideas.
pan1=function(x,y) {
points(x,y,pch=18) # to add points
m=lm(y~x) # to fit a simple regression model
abline(m,lwd=2)
}
dump("pan1",file="pan1.txt") # to save it
```

```r
source("pan1.txt")  # to source it at workplace
# Use it with pairs as
pairs(trees,panel=pan1) # to add fitted line in each panel
```

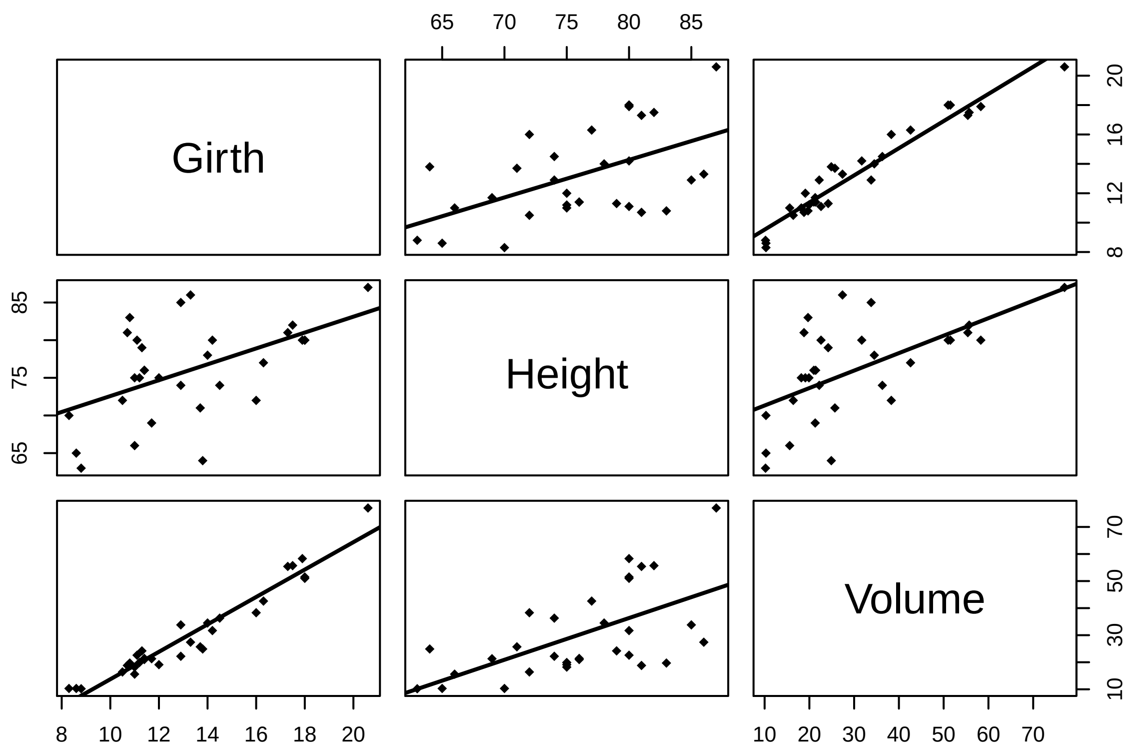


Figure 1: Scatter plot with fitted lines

The Function 'lm()'

This function is meant for fitting ordinary least square regression model, namely

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p xip + e_i$$

for $i = 1, \cdots, n$. The function 'lm()' has two main arguments, 'formula' and 'data' specifies the data frame from which data is taken. It returns almost all the important things which are desired in the data analysis.

Example ('trees')

The data frame 'trees' , which is available in the package **datasets** in the **base R**, is a data frame consisting of columns of 'Volume'. 'Girth',and 'Height' of black cherry trees. To fit this data to an intercept model.

$$y_i = \beta_0 + e_i$$

or

$$Volume = \beta_0 + e_i$$

This model known as intercept model. We use the function 'lm()' to fit this model.

```r
M0=lm(Volume~1,data=trees) # To fit
print(M0)  # To print brief
```

```
##
## Call:
## lm(formula = Volume ~ 1, data = trees)
##
## Coefficients:
## (Intercept)
##       30.17
```

```r
summary(M0) # To print detailed
```

```
##
## Call:
## lm(formula = Volume ~ 1, data = trees)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.971 -10.771  -5.971   7.129  46.829
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   30.171      2.952   10.22 2.75e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.44 on 30 degrees of freedom
```

Fitting of Simple Regression Model using 'lm()'

The simple regression model is one in which there is only one regressor or input. The model for 'trees' data with 'Grith' as input is

$$Volume = \beta_0 + \beta_1 Girth + e$$

We fit it using 'lm()' as

```r
M1=lm(Volume~1+Girth,data=trees)
# This is same as lm(Volume~Girth,data=trees)
print(M1)
```

```
##
## Call:
## lm(formula = Volume ~ 1 + Girth, data = trees)
##
## Coefficients:
## (Intercept)        Girth
##     -36.943        5.066
```

```r
summary(M1)
```

```
##
## Call:
## lm(formula = Volume ~ 1 + Girth, data = trees)
##
## Residuals:
##    Min     1Q Median     3Q    Max
```

```
## -8.065 -3.107  0.152  3.495  9.587
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
## Girth         5.0659     0.2474   20.48  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.252 on 29 degrees of freedom
## Multiple R-squared:  0.9353, Adjusted R-squared:  0.9331
## F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

We can get a lot information from the fitted object 'M1' by using the following command :

```r
names(M1)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```r
names(summary(M1))
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

One can extract any of these using the component command \$ or [] . For example to extract 'coefficients' we use

```r
M1$coefficients
```

```
## (Intercept)       Girth
##  -36.943459    5.065856
```

Similarly, to extract multiple R^2 use

```r
summary(M1)$r.squared
```

```
## [1] 0.9353199
```

Extract important information from a fitted model

From a fitted model, extract the information about regression coefficient $\hat{\beta}$, model matrix x, residual sum of squares $\hat{e}^T\hat{e}$, and variance covariance matrix of $\hat{\beta}$.

```r
# for beta hat
coef(M1)
```

```
## (Intercept)       Girth
##  -36.943459    5.065856
```

```r
summary(M1)$coef
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -36.943459   3.365145 -10.97827 7.621449e-12
## Girth         5.065856   0.247377  20.47829 8.644334e-19
```

```r
# for residuals
residuals(M1)
```

```
##          1          2          3          4          5          6          7 
##  5.1968508  3.6770939  2.5639226  0.1519667  1.5387954  1.9322098 -3.1809615 
##          8          9         10         11         12         13         14 
## -0.5809615  3.3124528  0.1058672  3.8992815  0.1926959  0.5926959 -1.0270610 
##         15         16         17         18         19         20         21 
## -4.7468179 -6.2060887  5.3939113 -3.0324313 -6.7587739 -8.0653595  0.5214692 
##         22         23         24         25         26         27         28 
## -3.2917021 -0.2114590 -5.8102436 -3.0300006  4.7041430  3.9909717  4.5646292 
##         29         30         31 
## -2.7419565 -3.2419565  9.5868168
```

```r
# for residual sum of squares
sum(residuals(M1)^2)
```

```
## [1] 524.3025
```

```r
# There is simple function deviance for it
deviance(M1)
```

```
## [1] 524.3025
```

```r
# for model matrix X
X=model.matrix(M1)
X
```

```
##    (Intercept) Girth
## 1            1   8.3
## 2            1   8.6
## 3            1   8.8
## 4            1  10.5
## 5            1  10.7
## 6            1  10.8
## 7            1  11.0
## 8            1  11.0
## 9            1  11.1
## 10           1  11.2
## 11           1  11.3
## 12           1  11.4
## 13           1  11.4
## 14           1  11.7
## 15           1  12.0
## 16           1  12.9
## 17           1  12.9
## 18           1  13.3
## 19           1  13.7
## 20           1  13.8
## 21           1  14.0
## 22           1  14.2
## 23           1  14.5
## 24           1  16.0
## 25           1  16.3
## 26           1  17.3
## 27           1  17.5
## 28           1  17.9
## 29           1  18.0
## 30           1  18.0
## 31           1  20.6
```

```
## attr(,"assign")
## [1] 0 1
```

```r
X[c(1:3,31),]
```

```
##    (Intercept) Girth
## 1            1   8.3
## 2            1   8.6
## 3            1   8.8
## 31           1  20.6
```

```r
# For MSresidual
deviance(M1)/df.residual(M1)
```

```
## [1] 18.0794
```

```r
# Square root of it can be obtained as
summary(M1)$sigma
```

```
## [1] 4.251988
```

```r
# To get fitted value Xbetahat
X_betahat=fitted(M1)
X_betahat
```

```
##         1         2         3         4         5         6         7         8
##  5.103149  6.622906  7.636077 16.248033 17.261205 17.767790 18.780962 18.780962
##         9        10        11        12        13        14        15        16
## 19.287547 19.794133 20.300718 20.807304 20.807304 22.327061 23.846818 28.406089
##        17        18        19        20        21        22        23        24
## 28.406089 30.432431 32.458774 32.965360 33.978531 34.991702 36.511459 44.110244
##        25        26        27        28        29        30        31
## 45.630001 50.695857 51.709028 53.735371 54.241956 54.241956 67.413183
```

```r
# To get predicted values
pred1=predict(M1) ;pred1
```

```
##         1         2         3         4         5         6         7         8
##  5.103149  6.622906  7.636077 16.248033 17.261205 17.767790 18.780962 18.780962
##         9        10        11        12        13        14        15        16
## 19.287547 19.794133 20.300718 20.807304 20.807304 22.327061 23.846818 28.406089
##        17        18        19        20        21        22        23        24
## 28.406089 30.432431 32.458774 32.965360 33.978531 34.991702 36.511459 44.110244
##        25        26        27        28        29        30        31
## 45.630001 50.695857 51.709028 53.735371 54.241956 54.241956 67.413183
```

```r
# To get variance covariance matrix of beta
vcov(M1)
```

```
##             (Intercept)       Girth
## (Intercept)  11.3242005 -0.81073976
## Girth        -0.8107398  0.06119536
```

Centered form of Simple Regression Model

The form of the model

$$y_i = \beta_0 + \beta_1(x_i - \bar{x}) + e_i$$

is called the centered form of simple linear regression model. Note that in this form $\hat{\beta}_1$ remains same, but $\hat{\beta}_0 = \bar{y}$ in this form. Moreover, x_i is replaced by $(x_i - \bar{x})$ in the centered form. Thus to implement 'slr()'

function 'x-mean(x)' , and call it into 'slr()' as argument 'x'. Similarly changes are also required in 'lm()' to implement it. We shall make use of the 'transform()' function for this data manipulation. Following set of commands will make the things more clear:

```
# Example of trees data with Girth as predictor .
# Use the function transform () to transform the variable .
d1=trees # Assign trees to d1
d1=transform(d1,Girth.c=Girth-mean(Girth))
head(d1)
```

```
##   Girth Height Volume   Girth.c
## 1   8.3     70   10.3 -4.948387
## 2   8.6     65   10.3 -4.648387
## 3   8.8     63   10.2 -4.448387
## 4  10.5     72   16.4 -2.748387
## 5  10.7     81   18.8 -2.548387
## 6  10.8     83   19.7 -2.448387
```

```
# Fit the centered form
M1c=lm(Volume~Girth.c,data=d1)
# Fit the non-centered form
M1=lm(Volume~Girth,data=d1)
summary(M1c) # See that beta1 is not change and beta0=mean(y)
```

```
##
## Call:
## lm(formula = Volume ~ Girth.c, data = d1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.065 -3.107  0.152  3.495  9.587
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  30.1710     0.7637   39.51   <2e-16 ***
## Girth.c       5.0659     0.2474   20.48   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.252 on 29 degrees of freedom
## Multiple R-squared:  0.9353, Adjusted R-squared:  0.9331
## F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

```
summary(M1)
```

```
##
## Call:
## lm(formula = Volume ~ Girth, data = d1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.065 -3.107  0.152  3.495  9.587
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
```

```
## Girth         5.0659     0.2474   20.48  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.252 on 29 degrees of freedom
## Multiple R-squared:  0.9353, Adjusted R-squared:  0.9331
## F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

```
## See what happens in correlation matrix of beta.
cov2cor(vcov(M1c))
```

```
##                (Intercept)      Girth.c
## (Intercept) 1.000000e+00 1.033469e-16
## Girth.c     1.033469e-16 1.000000e+00
```

```
cov2cor(vcov(M1))
```

```
##             (Intercept)      Girth
## (Intercept)   1.0000000 -0.9739092
## Girth        -0.9739092  1.0000000
```

Center Form of 'SLR()'

```
slr3=function(y,x1) {
X=cbind(1,x1)  # model matrix
n=nrow(X)
p1=ncol(X)
XtX=crossprod(X,X)
Xty=crossprod(X,y)
beta= solve(XtX,Xty)
resid=y-X%*%beta
rss= sum(resid^2)
msresid=rss/(n-p1)
sebeta=sqrt(diag(msresid*solve(XtX)))
tratio=beta/sebeta
pvalue=2*(1-pt(abs(tratio),df=n-p1))
out=data.frame(RegCoeff=beta,SEbeta=sebeta,Tvalue=tratio,pvalue=pvalue)
out=round(out,3) # Round upto 3 digits
return(out)
}
dump("slr3",file="slr3.txt")
## Analyse the data 'trees' using 'volume' as response and
 # 'Girth' and 'Height' as regressors.
d1=trees
names(trees)
```

```
## [1] "Girth"  "Height" "Volume"
```

```
y=d1$Volume
x1=d1$Girth-mean(d1$Girth)
M3c=slr3(y,x1)
print(M3c)
```

```
##    RegCoeff SEbeta Tvalue pvalue
##      30.171  0.764 39.507      0
## x1    5.066  0.247 20.478      0
```

```r
summary(M3c)
```

```
##     RegCoeff          SEbeta           Tvalue          pvalue 
##  Min.   : 5.066   Min.   :0.2470   Min.   :20.48   Min.   :0  
##  1st Qu.:11.342   1st Qu.:0.3762   1st Qu.:25.24   1st Qu.:0  
##  Median :17.619   Median :0.5055   Median :29.99   Median :0  
##  Mean   :17.619   Mean   :0.5055   Mean   :29.99   Mean   :0  
##  3rd Qu.:23.895   3rd Qu.:0.6348   3rd Qu.:34.75   3rd Qu.:0  
##  Max.   :30.171   Max.   :0.7640   Max.   :39.51   Max.   :0
```

Note that estimates are highly correlated in non-centered form, whereas they are not in centered form. Moreover, $\hat{\beta}_0$ is simply $\bar{y}$, which is mean of the response vector y. For these reasons, *centered form* is prefered over *non-centered form* of the model. These ideas can be extended to *Multiple Regression Model* also.

Fitting of Multiple Regression Model with 'lm()'

For the 'trees' data one can fit a multiple regression model using the function 'lm()'. The model is

$$Volume = \beta_0 + \beta_1 Girth + \beta_2 Height + e$$

This model can be fitted as :

```r
# Fit a model M2
M2=lm(Volume~Girth+Height,data=trees)
print(M2)
```

```
## 
## Call:
## lm(formula = Volume ~ Girth + Height, data = trees)
## 
## Coefficients:
## (Intercept)        Girth       Height  
##    -57.9877       4.7082       0.3393
```

```r
summary(M2)
```

```
## 
## Call:
## lm(formula = Volume ~ Girth + Height, data = trees)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -6.4065 -2.6493 -0.2876  2.2003  8.4847 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -57.9877     8.6382  -6.713 2.75e-07 ***
## Girth         4.7082     0.2643  17.816  < 2e-16 ***
## Height        0.3393     0.1302   2.607   0.0145 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 3.882 on 28 degrees of freedom
## Multiple R-squared:  0.948,  Adjusted R-squared:  0.9442 
## F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```

```r
names(M2)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```r
coef(M2)
```

```
## (Intercept)       Girth      Height
## -57.9876589   4.7081605   0.3392512
```

```r
coef(summary(M2))
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -57.9876589  8.6382259 -6.712913 2.749507e-07
## Girth         4.7081605  0.2642646 17.816084 8.223304e-17
## Height        0.3392512  0.1301512  2.606594 1.449097e-02
```

```r
residuals(M2)[c(1,3,31)] # Only First, Third and 31st residuals
```

```
##        1        3       31
## 5.462340 5.383019 8.484695
```

```r
X=model.matrix(M2)
X
```

```
##    (Intercept) Girth Height
## 1            1   8.3     70
## 2            1   8.6     65
## 3            1   8.8     63
## 4            1  10.5     72
## 5            1  10.7     81
## 6            1  10.8     83
## 7            1  11.0     66
## 8            1  11.0     75
## 9            1  11.1     80
## 10           1  11.2     75
## 11           1  11.3     79
## 12           1  11.4     76
## 13           1  11.4     76
## 14           1  11.7     69
## 15           1  12.0     75
## 16           1  12.9     74
## 17           1  12.9     85
## 18           1  13.3     86
## 19           1  13.7     71
## 20           1  13.8     64
## 21           1  14.0     78
## 22           1  14.2     80
## 23           1  14.5     74
## 24           1  16.0     72
## 25           1  16.3     77
## 26           1  17.3     81
## 27           1  17.5     82
## 28           1  17.9     80
## 29           1  18.0     80
## 30           1  18.0     80
```

```
## 31           1  20.6     87
## attr(,"assign")
## [1] 0 1 2
```

```
X[c(1:4,31)] # to print first four and last row of X
```

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

```
deviance(M2) # Residual sum of square
```

```
## [1] 421.9214
```

```
df.residual(M2) # residual degrees of freedom
```

```
## [1] 28
```

```
msresid=deviance(M2)/df.residual(M2)
msresid # hatsigma^2
```

```
## [1] 15.06862
```

```
sqrt(msresid) # hatsigma
```

```
## [1] 3.881832
```

```
summary(M2)$sigma
```

```
## [1] 3.881832
```

```
fitted(M2)[c(1:4,31)]
```

```
##         1         2         3         4        31
##  4.837660  4.553852  4.816981 15.874115 68.515305
```

```
predict(M2)[c(1:4,31)]
```

```
##         1         2         3         4        31
##  4.837660  4.553852  4.816981 15.874115 68.515305
```

```
vcov(M2)
```

```
##             (Intercept)       Girth      Height
## (Intercept)  74.6189461  0.43217138 -1.05076889
## Girth         0.4321714  0.06983578 -0.01786030
## Height       -1.0507689 -0.01786030  0.01693933
```

```
cov2cor(vcov(M2)) # Covariance to Correlation Matrix
```

```
##             (Intercept)      Girth     Height
## (Intercept)   1.0000000  0.1893182 -0.9346189
## Girth         0.1893182  1.0000000 -0.5192801
## Height       -0.9346189 -0.5192801  1.0000000
```

Exercise: Centered form with two predictors

- We can extend *centered form* for multiple regression also.
- Fit centered model for the 'trees' data, that is

$$y_i = \beta_0 + \beta_1(x_{i1} - \bar{x_1}) + \beta_2(x_{i2} - \bar{x_2}) + e_i$$

  or equivalently for 'trees' data

$$Volume = \beta_0 + \beta_1(Girth\ mean(Girth)) + \beta_2(Height - mean(Height)) + error$$

Using R commands.

Hint: Use the function 'transform()' to get centered form of 'Girth' and 'Height', and then fit the centered form for it.

Solution

```
d2=trees # Assign trees to d1
d2=transform(d1,Girth.c=Girth-mean(Girth),Height.c=Height-mean(Height))
head(d2)
```

```
##   Girth Height Volume   Girth.c Height.c
## 1   8.3     70   10.3 -4.948387       -6
## 2   8.6     65   10.3 -4.648387      -11
## 3   8.8     63   10.2 -4.448387      -13
## 4  10.5     72   16.4 -2.748387       -4
## 5  10.7     81   18.8 -2.548387        5
## 6  10.8     83   19.7 -2.448387        7
```

```
# Fit the centered form
M2c=lm(Volume~Girth.c+Height.c,data=d2)
# Fit the non-centered form
M2=lm(Volume~Girth+Height,data=d2)
summary(M2c) # See that beta1 is not change and beta0=mean(y)
```

```
##
## Call:
## lm(formula = Volume ~ Girth.c + Height.c, data = d2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4065 -2.6493 -0.2876  2.2003  8.4847
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  30.1710     0.6972  43.275   <2e-16 ***
## Girth.c       4.7082     0.2643  17.816   <2e-16 ***
## Height.c      0.3393     0.1302   2.607   0.0145 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.882 on 28 degrees of freedom
## Multiple R-squared:  0.948,  Adjusted R-squared:  0.9442
## F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```

```
summary(M2)
```

```
##
## Call:
## lm(formula = Volume ~ Girth + Height, data = d2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4065 -2.6493 -0.2876  2.2003  8.4847
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -57.9877     8.6382  -6.713 2.75e-07 ***
## Girth         4.7082     0.2643  17.816  < 2e-16 ***
## Height        0.3393     0.1302   2.607   0.0145 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.882 on 28 degrees of freedom
## Multiple R-squared:  0.948,  Adjusted R-squared:  0.9442
## F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```

```
## See what happens in correlation matrix of beta.
cov2cor(vcov(M2c))
```

```
##               (Intercept)      Girth.c      Height.c
## (Intercept)  1.000000e+00  9.000217e-17 -3.238109e-18
## Girth.c      9.000217e-17  1.000000e+00 -5.192801e-01
## Height.c    -3.238109e-18 -5.192801e-01  1.000000e+00
```

```
cov2cor(vcov(M2))
```

```
##             (Intercept)      Girth     Height
## (Intercept)   1.0000000  0.1893182 -0.9346189
## Girth         0.1893182  1.0000000 -0.5192801
## Height       -0.9346189 -0.5192801  1.0000000
```

Centered Model by Own Model

```
mlr4=function(y,x1,x2) {
X=cbind(1,x1,x2)  # model matrix
n=nrow(X)
p1=ncol(X)
XtX=crossprod(X,X)
Xty=crossprod(X,y)
beta= solve(XtX,Xty)
resid=y-X%*%beta
rss= sum(resid^2)
msresid=rss/(n-p1)
sebeta=sqrt(diag(msresid*solve(XtX)))
tratio=beta/sebeta
pvalue=2*(1-pt(abs(tratio),df=n-p1))
out=data.frame(RegCoeff=beta,SEbeta=sebeta,Tvalue=tratio,pvalue=pvalue)
out=round(out,3) # Round upto 3 digits
return(out)
}
dump("mlr4",file="mlr4.txt")
## Analyse the data 'trees' using 'volume' as response and
 # 'Girth' and 'Height' as regressors.
d2=trees
names(trees)
```

```
## [1] "Girth"  "Height" "Volume"
```

```
y=d2$Volume
x1=trees$Girth-mean(d2$Girth)
x2=trees$Height-mean(d2$Height)
M4c=mlr4(y,x1,x2)
```

```
print(M4c)
```

```
##    RegCoeff SEbeta Tvalue pvalue
##      30.171  0.697 43.275  0.000
## x1    4.708  0.264 17.816  0.000
## x2    0.339  0.130  2.607  0.014
```

```
summary(M4c)
```

```
##     RegCoeff          SEbeta           Tvalue           pvalue
##  Min.   : 0.339   Min.   :0.1300   Min.   : 2.607   Min.   :0.000000
##  1st Qu.: 2.523   1st Qu.:0.1970   1st Qu.:10.211   1st Qu.:0.000000
##  Median : 4.708   Median :0.2640   Median :17.816   Median :0.000000
##  Mean   :11.739   Mean   :0.3637   Mean   :21.233   Mean   :0.004667
##  3rd Qu.:17.439   3rd Qu.:0.4805   3rd Qu.:30.546   3rd Qu.:0.007000
##  Max.   :30.171   Max.   :0.6970   Max.   :43.275   Max.   :0.014000
```

Exercise 2 : Generate a data frame on 'Weight','Height' and 'Age' of the students of B.Sc.(V) Semester . Fit this data using "Weight' as response and 'Height' and 'Age' as regressors. Make your comments on the results obtained.

Solution

```
height=c(175,170,180,173,170,172,165,174,160,172,177,172,170,172,170,165,165,180,167,177,170,170,177,18
weight=c(65,63,90,55,60,62,53,68,43,62,70,55,62,75,60,67,45,58,75,78,65,55,80,60,63,60,62,59,58)
Age=c(21,20,21,20,20,21,22,22,19,21,20,20,21,22,20,23,20,23,23,20,21,22,23,21,22,22,21,22,21)
Vsem=data.frame("HEIGHT"=height,"WEIGHT"=weight,"AGE"=Age)
Vsem
```

```
##    HEIGHT WEIGHT AGE
## 1     175     65  21
## 2     170     63  20
## 3     180     90  21
## 4     173     55  20
## 5     170     60  20
## 6     172     62  21
## 7     165     53  22
## 8     174     68  22
## 9     160     43  19
## 10    172     62  21
## 11    177     70  20
## 12    172     55  20
## 13    170     62  21
## 14    172     75  22
## 15    170     60  20
## 16    165     67  23
## 17    165     45  20
## 18    180     58  23
## 19    167     75  23
## 20    177     78  20
## 21    170     65  21
## 22    170     55  22
## 23    177     80  23
## 24    180     60  21
## 25    172     63  22
## 26    165     60  22
```

```
## 27    170     62  21
## 28    167     59  22
## 29    165     58  21
```

```r
dump("Vsem",file = "Vsem.txt")
source("Vsem.txt")
# for mlr
V=lm(weight~height+Age)
V
```

```
##
## Call:
## lm(formula = weight ~ height + Age)
##
## Coefficients:
## (Intercept)       height          Age
##    -175.914        1.065        2.679
```

```r
print(V)
```

```
##
## Call:
## lm(formula = weight ~ height + Age)
##
## Coefficients:
## (Intercept)       height          Age
##    -175.914        1.065        2.679
```

```r
summary(V)
```

```
##
## Call:
## lm(formula = weight ~ height + Age)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.4058  -3.2067   0.6026   3.8272  17.9529
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -175.9143    55.5820  -3.165  0.00393 **
## height         1.0650     0.2931   3.633  0.00121 **
## Age            2.6794     1.3505   1.984  0.05791 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.862 on 26 degrees of freedom
## Multiple R-squared:  0.4158, Adjusted R-squared:  0.3709
## F-statistic: 9.254 on 2 and 26 DF,  p-value: 0.0009226
```

Model Comparision

- Check whether multiple Regression Model is better than Simple or not.
- This model comparison is based on the concept of extra sum of squares.
- _R_implemention is 'anova()'

Let M_1 and M_2 be the two models defined as

$$M_2: \quad y = \beta_0 + \beta_1 x_1 + e$$

and

$$M2: \quad y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + e$$

* It may be noted that residual sum of square for M_1 will be more than that of M_2.

- For 'trees' data with 'Girth' as predictor we have

```
M1=lm(Volume~Girth,data=trees) ;M1
```

```
##
## Call:
## lm(formula = Volume ~ Girth, data = trees)
##
## Coefficients:
## (Intercept)        Girth
##     -36.943        5.066
```

When both 'Girth' and 'Height' are used as predictors, we have

```
M2=lm(Volume~Girth+Height,data=trees) ;M2
```

```
##
## Call:
## lm(formula = Volume ~ Girth + Height, data = trees)
##
## Coefficients:
## (Intercept)        Girth       Height
##    -57.9877       4.7082       0.3393
```

Get the important summaries for these two models:

```
deviance(M1)
```

```
## [1] 524.3025
```

```
deviance(M2)
```

```
## [1] 421.9214
```

```
df.residual(M1)
```

```
## [1] 29
```

```
df.residual(M2)
```

```
## [1] 28
```

```
# make a model comparison
anova(M1,M2)
```

```
## Analysis of Variance Table
##
## Model 1: Volume ~ Girth
## Model 2: Volume ~ Girth + Height
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     29 524.30
## 2     28 421.92  1    102.38 6.7943 0.01449 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This shows that addition of 'Height' significantly improves the model.Hence model M_2 is better than model M_1 .

Exercises :

*(1) Fit the data set 'trees' using 'slr()' and 'mlr()' and compare your results with 'lm()'.

*(2) Fit the same that data 'trees' using 'slr()' and 'mlr()' as centered form, and compare your results with 'lm()' .

Soluton *(1)

```
slr3=function(X,y) {
# A function which returns Simple Regression Analysis for trees data
X=cbind(1,x)  # model matrix
n=nrow(X)
p1=ncol(X)
XtX=crossprod(X,X)
Xty=crossprod(X,y)
beta= solve(XtX,Xty)
resid=y-X%*%beta
rss= sum(resid^2)
msresid=rss/(n-p1)
sebeta=sqrt(diag(msresid*solve(XtX)))
tratio=beta/sebeta
pvalue=2*(1-pt(abs(tratio),df=n-p1))
out=data.frame(RegCoeff=beta,SEbeta=sebeta,Tvalue=tratio,pvalue=pvalue)
return(out)
}
dump("slr3",file="slr3.txt")
source("slr3.txt")
y=trees$Volume  # To extract y from trees
x=trees$Girth #To extract X from trees
# Now we can fit the Model using 'slr()'
A1=slr3(X=x,y=y )
print(A1)
```

```
##     RegCoeff   SEbeta    Tvalue       pvalue
##   -36.943459 3.365145 -10.97827 7.621459e-12
## x   5.065856 0.247377  20.47829 0.000000e+00
```

```
summary(A1)
```

```
##     RegCoeff           SEbeta           Tvalue            pvalue
##  Min.   :-36.943   Min.   :0.2474   Min.   :-10.978   Min.   :0.000e+00
##  1st Qu.:-26.441   1st Qu.:1.0268   1st Qu.: -3.114   1st Qu.:1.905e-12
##  Median :-15.939   Median :1.8063   Median :  4.750   Median :3.811e-12
##  Mean   :-15.939   Mean   :1.8063   Mean   :  4.750   Mean   :3.811e-12
##  3rd Qu.: -5.436   3rd Qu.:2.5857   3rd Qu.: 12.614   3rd Qu.:5.716e-12
##  Max.   :  5.066   Max.   :3.3651   Max.   : 20.478   Max.   :7.621e-12
```

```
A2=lm(Volume~Girth,data=trees)
A2
```

```
##
## Call:
## lm(formula = Volume ~ Girth, data = trees)
```

```
##
## Coefficients:
## (Intercept)        Girth
##     -36.943        5.066
```

```r
print(A2)
```

```
##
## Call:
## lm(formula = Volume ~ Girth, data = trees)
##
## Coefficients:
## (Intercept)        Girth
##     -36.943        5.066
```

```r
summary(A2)
```

```
##
## Call:
## lm(formula = Volume ~ Girth, data = trees)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.065 -3.107  0.152  3.495  9.587
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
## Girth         5.0659     0.2474   20.48  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.252 on 29 degrees of freedom
## Multiple R-squared:  0.9353, Adjusted R-squared:  0.9331
## F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

*(2)

```r
mlr3=function(y,x1,x2) {
# Define a function to implement multiple linear regression from trees data
# y is the response variable
# x1 is one regressor
# x2 is another regressor
# This function returns Multiple Linear Analysis
X=cbind(1,x1,x2)  # model matrix
n=nrow(X)
p1=ncol(X)
XtX=crossprod(X,X)
Xty=crossprod(X,y)
beta= solve(XtX,Xty)
resid=y-X%*%beta
rss= sum(resid^2)
msresid=rss/(n-p1)
sebeta=sqrt(diag(msresid*solve(XtX)))
tratio=beta/sebeta
pvalue=2*(1-pt(abs(tratio),df=n-p1))
```

```r
out=data.frame(RegCoeff=beta,SEbeta=sebeta,Tvalue=tratio,pvalue=pvalue)
out=round(out,3) # Round upto 3 digits
return(out)
}
dump("mlr3",file="mlr3.txt")
## Analyse the data 'trees' using 'volume' as response and
 # 'Girth' and 'Height' as regressors.
data(trees)
names(trees)
```

```
## [1] "Girth"  "Height" "Volume"
```

```r
y=trees$Volume
x1=trees$Girth
x2=trees$Height
M4=mlr(y,x1,x2)
print(M4)
```

```
##    RegCoeff SEbeta Tvalue pvalue
##     -57.988  8.638 -6.713  0.000
## x1    4.708  0.264 17.816  0.000
## x2    0.339  0.130  2.607  0.014
```

```r
summary(M4)
```

```
##     RegCoeff           SEbeta          Tvalue           pvalue
##  Min.   :-57.988   Min.   :0.130   Min.   :-6.713   Min.   :0.000000
##  1st Qu.:-28.825   1st Qu.:0.197   1st Qu.:-2.053   1st Qu.:0.000000
##  Median :  0.339   Median :0.264   Median : 2.607   Median :0.000000
##  Mean   :-17.647   Mean   :3.011   Mean   : 4.570   Mean   :0.004667
##  3rd Qu.:  2.523   3rd Qu.:4.451   3rd Qu.:10.211   3rd Qu.:0.007000
##  Max.   :  4.708   Max.   :8.638   Max.   :17.816   Max.   :0.014000
```

```r
A4=lm(Volume~Girth+Height,data=trees)
A4
```

```
##
## Call:
## lm(formula = Volume ~ Girth + Height, data = trees)
##
## Coefficients:
## (Intercept)        Girth       Height
##    -57.9877       4.7082       0.3393
```

```r
print(A4)
```

```
##
## Call:
## lm(formula = Volume ~ Girth + Height, data = trees)
##
## Coefficients:
## (Intercept)        Girth       Height
##    -57.9877       4.7082       0.3393
```

```r
summary(A4)
```

```
##
## Call:
```

```
## lm(formula = Volume ~ Girth + Height, data = trees)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4065 -2.6493 -0.2876  2.2003  8.4847
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -57.9877     8.6382  -6.713 2.75e-07 ***
## Girth         4.7082     0.2643  17.816  < 2e-16 ***
## Height        0.3393     0.1302   2.607   0.0145 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.882 on 28 degrees of freedom
## Multiple R-squared:  0.948,  Adjusted R-squared:  0.9442
## F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```