Knowledge Alone Liberates

ज्ञानादेव तु कैवल्यम्

Somaiya Vidyavihar

Sr. no.	**Practical name**
1	**Installation and Basics of R**
2	**Time Series Analysis**
3	**Time Series Frequency Analysis** **1. obline()** **2. aggregate()** **3. boxplot()**
4	**Analysis Variance**
5	**Linear Regression**
6	**Hypothesis Testing**
7	**Decision Tree**
8	**Logistic Regression**
9	**K Means Clustering**

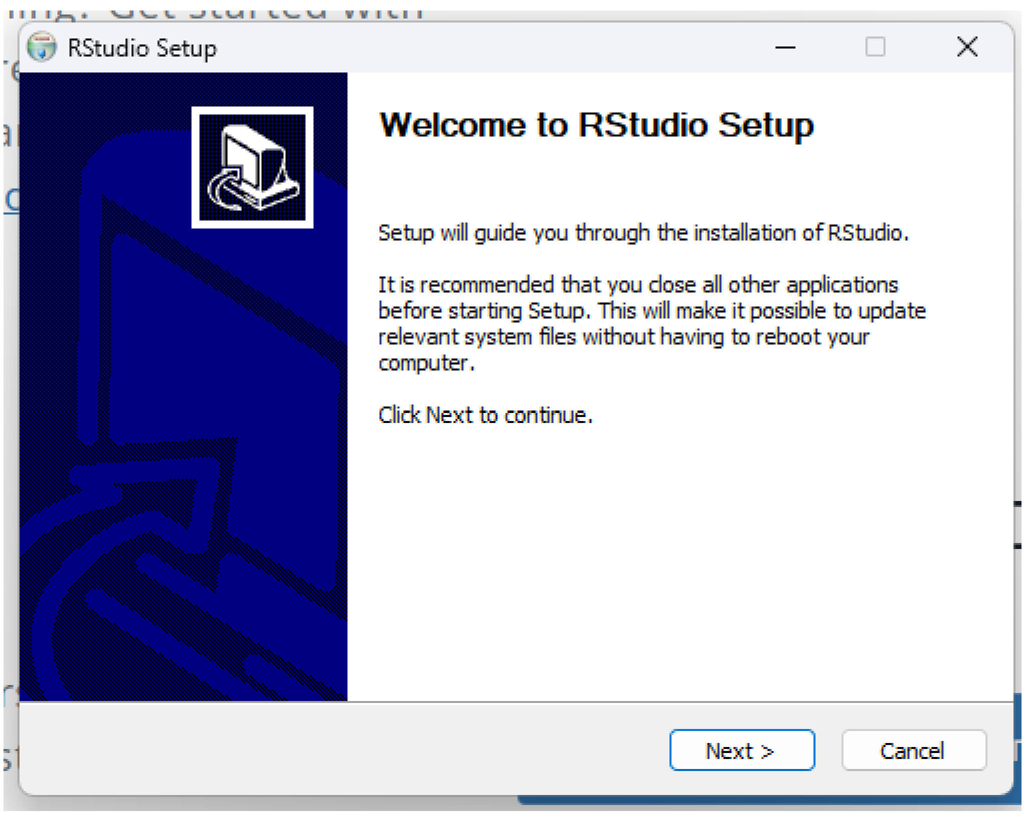
Practical 01

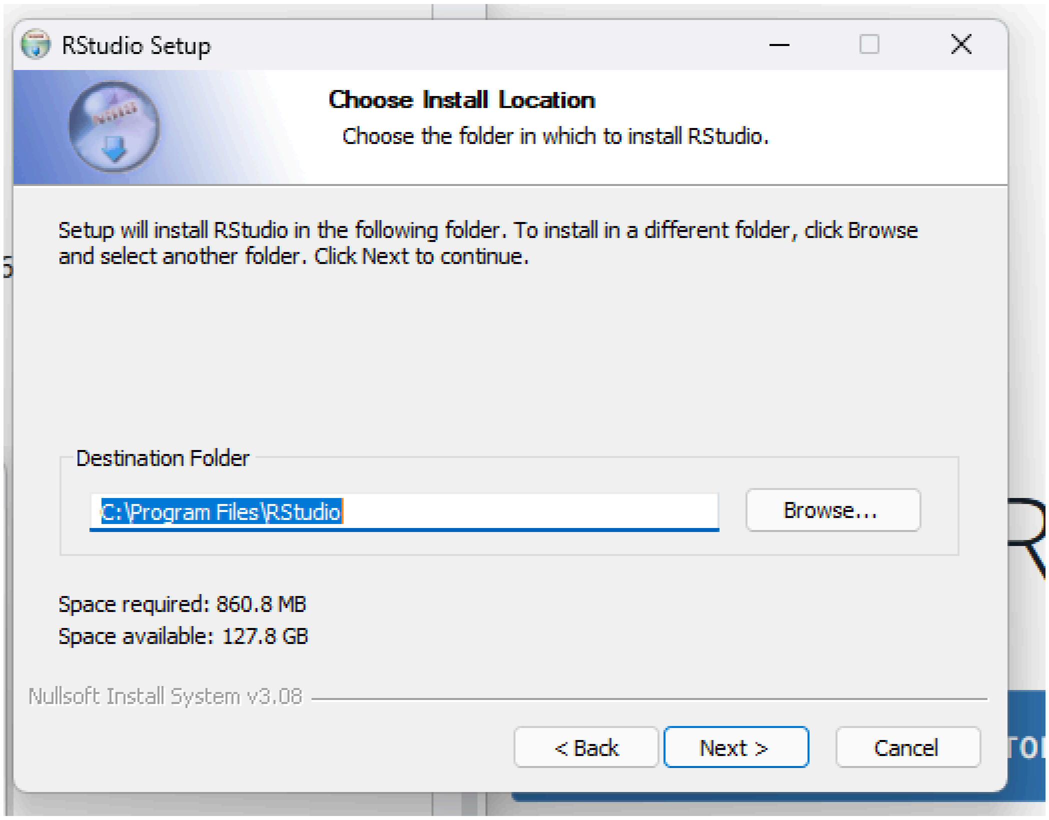
Installation

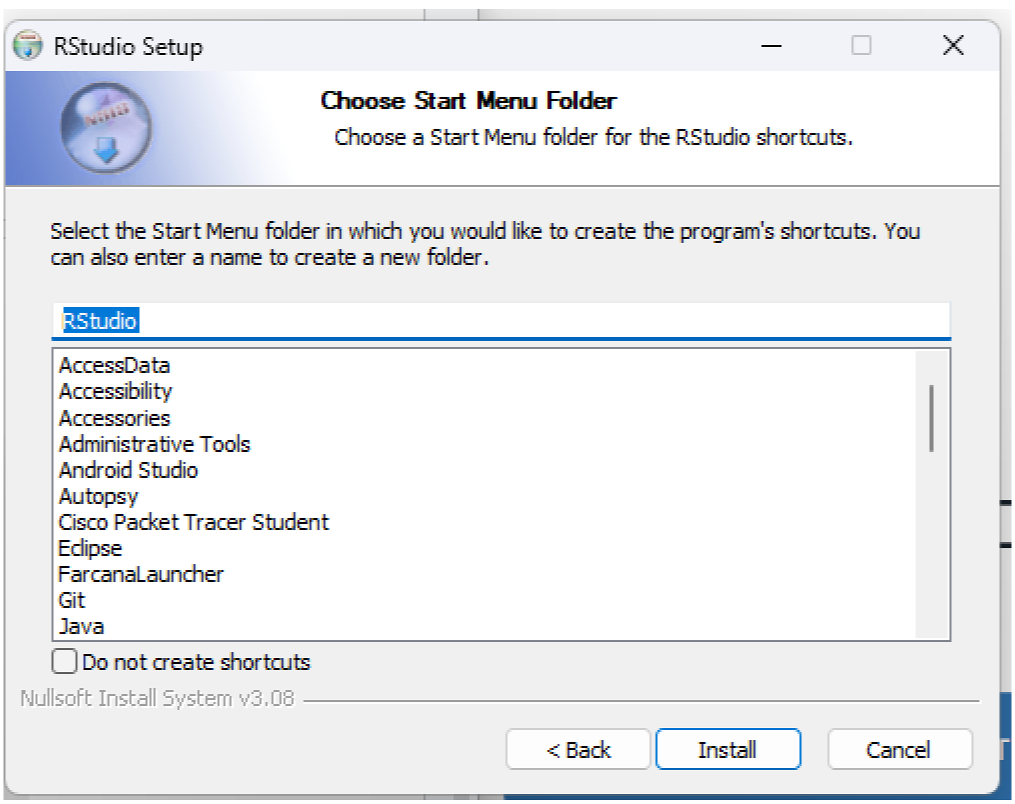
2: Install RStudio

DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS

Size: 215.66 MB | SHA-256: 93C7F307 | Version: 2023.12.0+369 | Released: 2023-12-20




RStudio Setup
Choose Install Location
Choose the folder in which to install RStudio.
Setup will install RStudio in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Program Files\RStudio
Browse...
Space required: 860.8 MB
Space available: 127.8 GB
Nullsoft Install System v3.08
< Back
Next >
Cancel


RStudio Setup
Choose Start Menu Folder
Choose a Start Menu folder for the RStudio shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
RStudio
AccessData
Accessibility
Accessories
Administrative Tools
Android Studio
Autopsy
Cisco Packet Tracer Student
Eclipse
FarcanaLauncher
Git
Java
Do not create shortcuts
Nullsoft Install System v3.08
< Back
Install
Cancel

1] Simple variable declaration

```
> name <- "Tanish"
> age <- 20
> name
[1] "Tanish"
> age
[1] 20
>
```

Values	
age	20
name	"Tanish"

2] For Loop

```
> for (i in 1:10) print(i)
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
>
```

3] Paste

```
> text <- "Pawsome"
> paste ("Py is", text)
[1] "Py is Pawsome"
>
```

4] Declaration of multiple variables

```
> text <- "Pawsome"
> paste ("Py is", text)
[1] "Py is Pawsome"
> var1 <- var2 <- var3 <- "Tanishhhh"
> var1
[1] "Tanishhhh"
> var3
[1] "Tanishhhh"
> var2
[1] "Tanishhhh"
> 
```

5] If else statement

```
> if(x> 10){
+     print(paste(x,"is greater than 10"))
+     }else if(x <10){print(paste(x,"is less than 10"))}else{print("x is equal to 10")}
[1] "5 is less than 10"
```

6] Maximum of 2 numbers

```
> a = 7
> b = 10
> if (a > b){
+     print("A is greater than B")
+ } else { print("B is greater than A")}
[1] "B is greater than A"
> 
```

7] Vector of strings

```
> fruits <- c("banana", "apple", "orange")
> fruits
[1] "banana" "apple"  "orange"
```

```
> numbers <- 1:10
> numbers
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
> sort(fruits)
[1] "apple"  "banana" "orange"
> sort(numbers)
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
> fruits[1]
[1] "banana"
> fruits[3]
[1] "orange"
> length(fruits)
[1] 3
> 
```

```
> fruits[c(1,3)]
[1] "banana" "orange"
> 
```

Practical 02

Aim: Time Series

Time Series Analysis in R is used to see how an object behaves over a period of time. In R Programming Language, it can be easily done by the **ts()** function with some parameters. Time series takes the data vector and each data is connected with a timestamp value as given by the user. This function is mostly used to learn and forecast the behavior of an asset in business for a period of time. For example, sales analysis of a company, inventory analysis, price analysis of a particular stock or market, population analysis, etc.

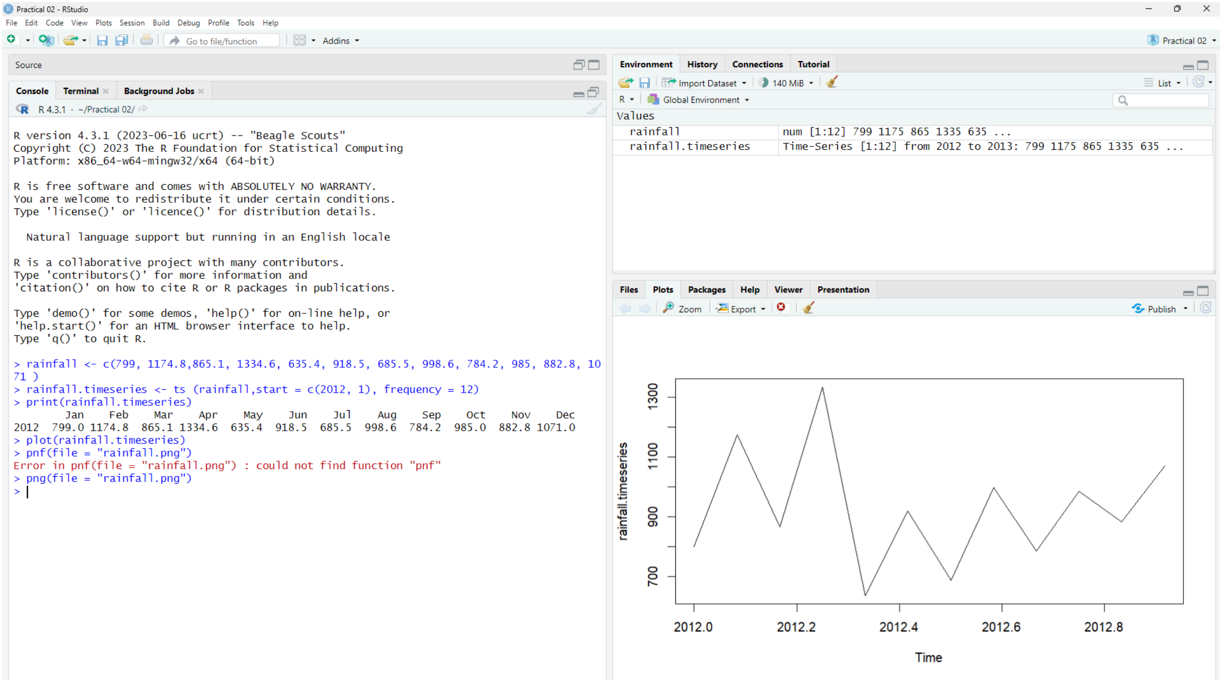
Syntax: objectName <- ts(data, start, end, frequency)

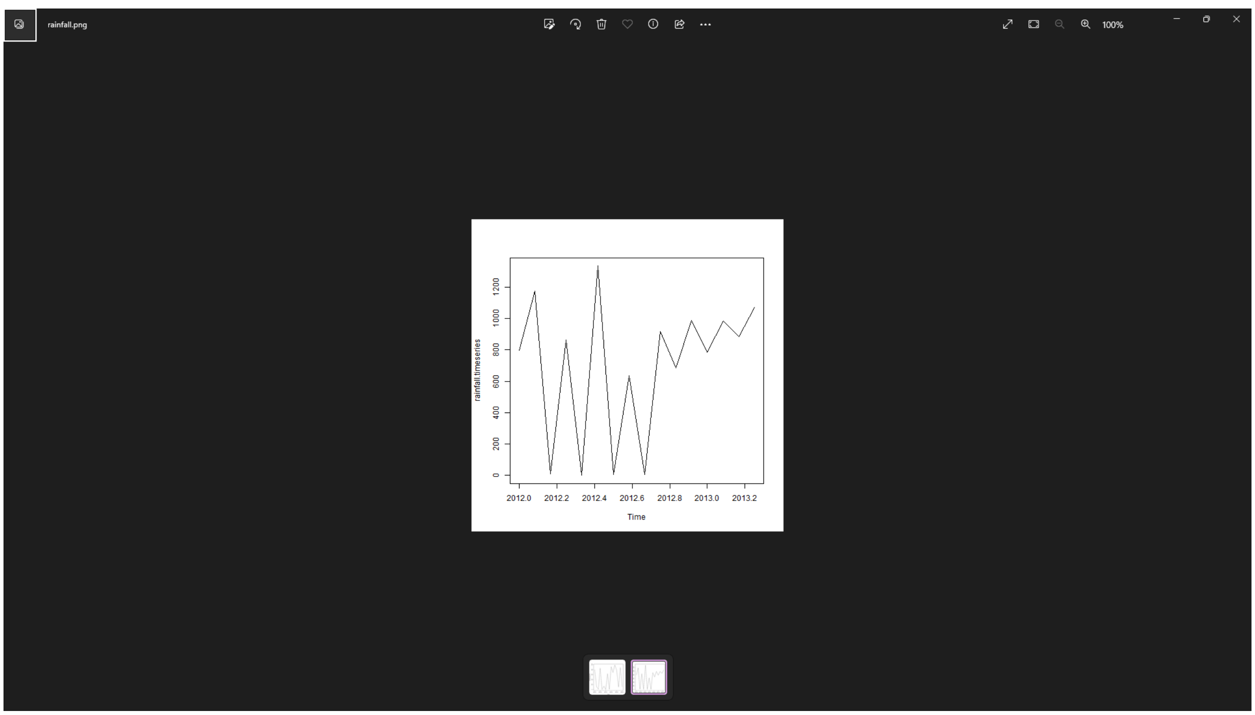
where,

- **data** – represents the data vector
- **start** – represents the first observation in time series
- **end** – represents the last observation in time series
- **frequency** – represents number of observations per unit time. For example, frequency=1 for monthly data.

A] Rainfall

```
rainfall <- c(799, 1174.8, 865.1, 1334.6, 635.4, 918.5, 685.5, 998.6, 784.2, 985. 882.8, 1071)

rainfall.timeseries <- ts (rainfall, start = c(2012, 1), frequency = 12)

print(rainfall.timeseries)

pnf(file = "rainfall.png")
```


Practical 02 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Practical 02
Source
Console
Terminal
Background Jobs
R 4.3.1 · ~/Practical 02/
R version 4.3.1 (2023-06-16 ucrt) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
> rainfall <- c(799, 1174.8,865.1, 1334.6, 635.4, 918.5, 685.5, 998.6, 784.2, 985, 882.8, 10
71)
> rainfall.timeseries <- ts (rainfall,start = c(2012, 1), frequency = 12)
> print(rainfall.timeseries)
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2012 799.0 1174.8 865.1 1334.6 635.4 918.5 685.5 998.6 784.2 985.0 882.8 1071.0
> plot(rainfall.timeseries)
> pnf(file = "rainfall.png")
Error in pnf(file = "rainfall.png") : could not find function "pnf"
> png(file = "rainfall.png")
>
Environment
History
Connections
Tutorial
Import Dataset
140 MiB
List
R
Global Environment
Values
rainfall
num [1:12] 799 1175 865 1335 635 ...
rainfall.timeseries
Time-Series [1:12] from 2012 to 2013: 799 1175 865 1335 635 ...
Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
rainfall.timeseries
700
900
1100
1300
2012.0
2012.2
2012.4
2012.6
2012.8
Time


rainfall.png
100%
rainfall.timeseries
0
200
400
600
800
1000
1200
2012.0
2012.2
2012.4
2012.6
2012.8
2013.0
2013.2
Time

Practical 03

Aim: Time Series Frequency Analysis

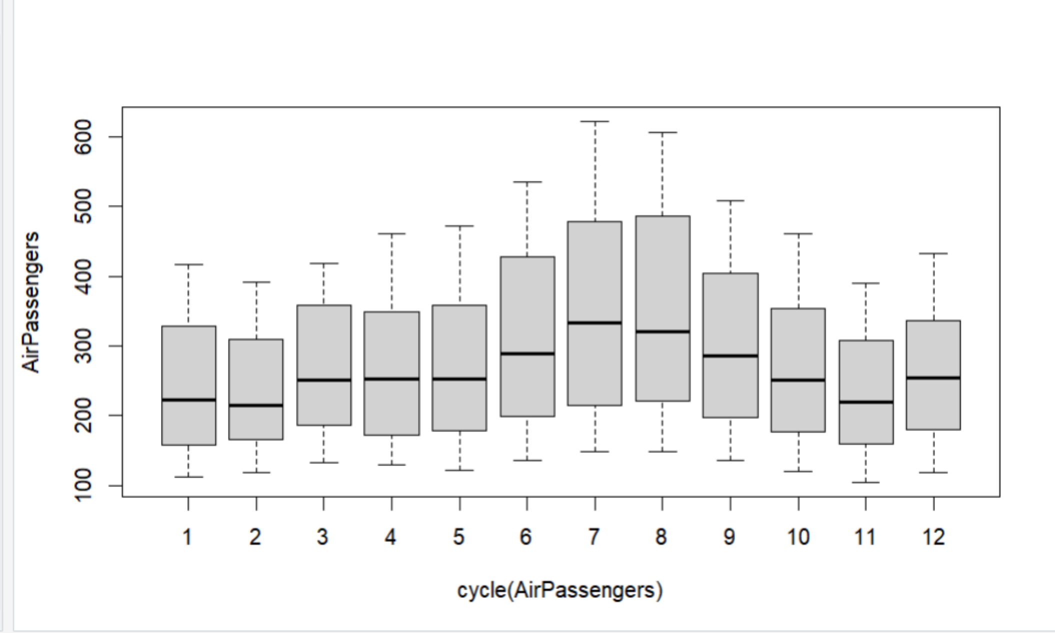
Time Series Analysis in R is used to see how an object behaves over a period of time. In R Programming Language, it can be easily done by the ts() function with some parameters. Time series takes the data vector and each data is connected with a timestamp value as given by the user. This function is mostly used to learn and forecast the behavior of an asset in business for a period of time. For example, sales analysis of a company, inventory analysis, price analysis of a particular stock or market, population analysis, etc.

abline() function in R Language is used to add one or more straight lines to a graph. The abline() function can be used to add vertical, horizontal or regression lines to plot.

> Syntax:
> abline(a=NULL, b=NULL, h=NULL, v=NULL, …)
>
> Parameters:
> a, b: It specifies the intercept and the slope of the line
> h: specifies y-value for horizontal line(s)
> v: specifies x-value(s) for vertical line(s)
>
> Returns: a straight line in the plot

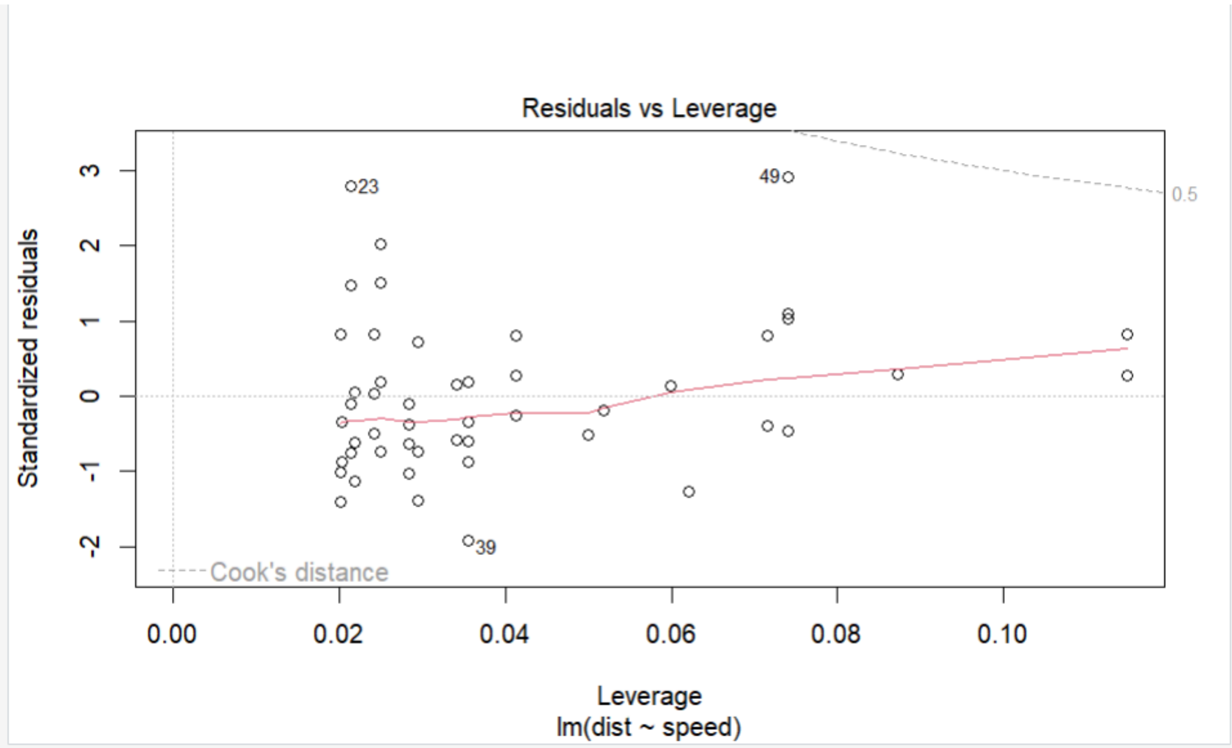
B] Air Passenger

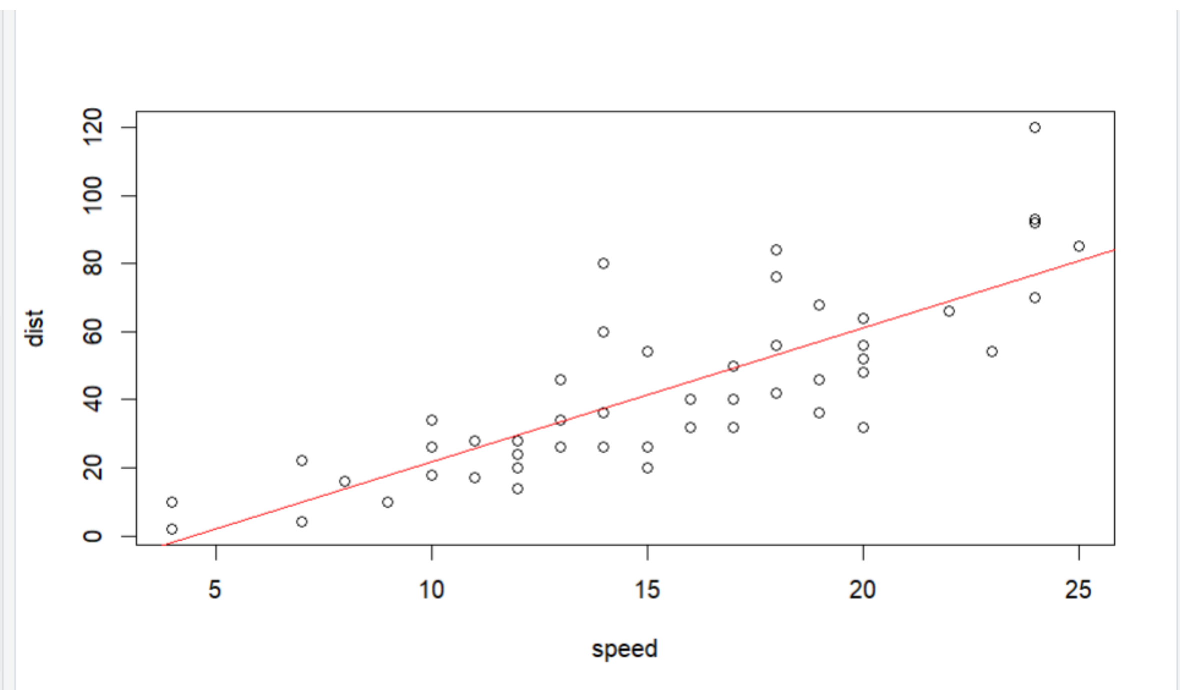
```
rainfall <- c(799, 1174.8, 865.1, 1334.6, 635.4, 918.5, 685.5, 998.6,
784.2, 985. 882.8, 1071)

rainfall.timeseries <- ts (rainfall, start = c(2012, 1), frequency = 12)

print(rainfall.timeseries)

pnf(file = "rainfall.png")
```



C] Cars

```
data("cars")
class(cars)
start(cars)
frequency(cars)
summary(cars)
plot(cars)
reg <- lm(dist  ~ speed, data = cars)
plot(cars)
```


Residuals vs Leverage
Standardized residuals
Leverage
lm(dist ~ speed)
Cook's distance
0.5
23
49
39
3
2
1
0
-1
-2
0.00
0.02
0.04
0.06
0.08
0.10


dist
speed
120
100
80
60
40
20
0
5
10
15
20
25

PRACTICAL 04

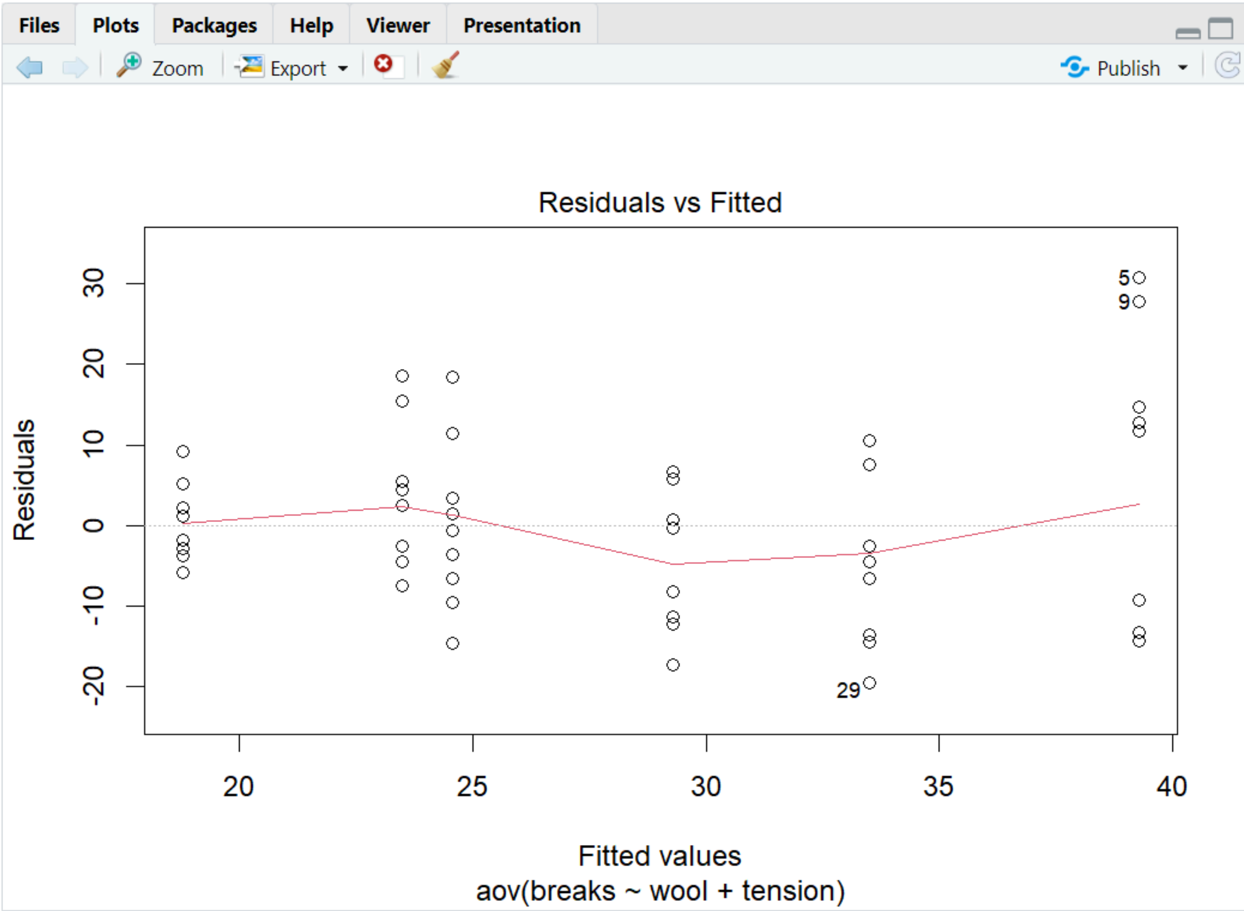
Q.1)

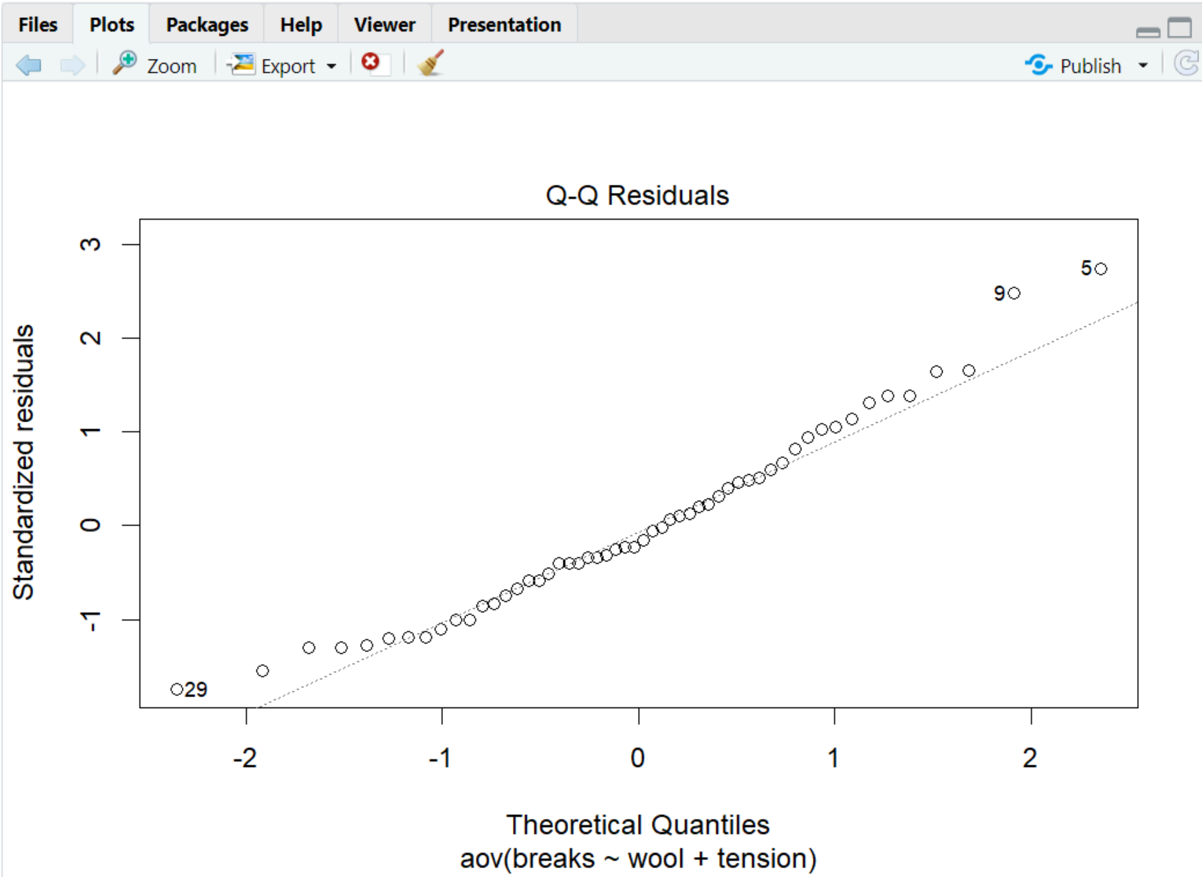
Code:

```
data("warpbreaks")

#Data Exploration
head(warpbreaks)
summary(warpbreaks)

Model_1<-aov(breaks~wool+tension, data = warpbreaks)
summary(Model_1)
plot(Model_1)

Model_2<-aov(breaks~wool+tension+wool:tension, data = warpbreaks)
summary(Model_2)
plot(Model_2)
```

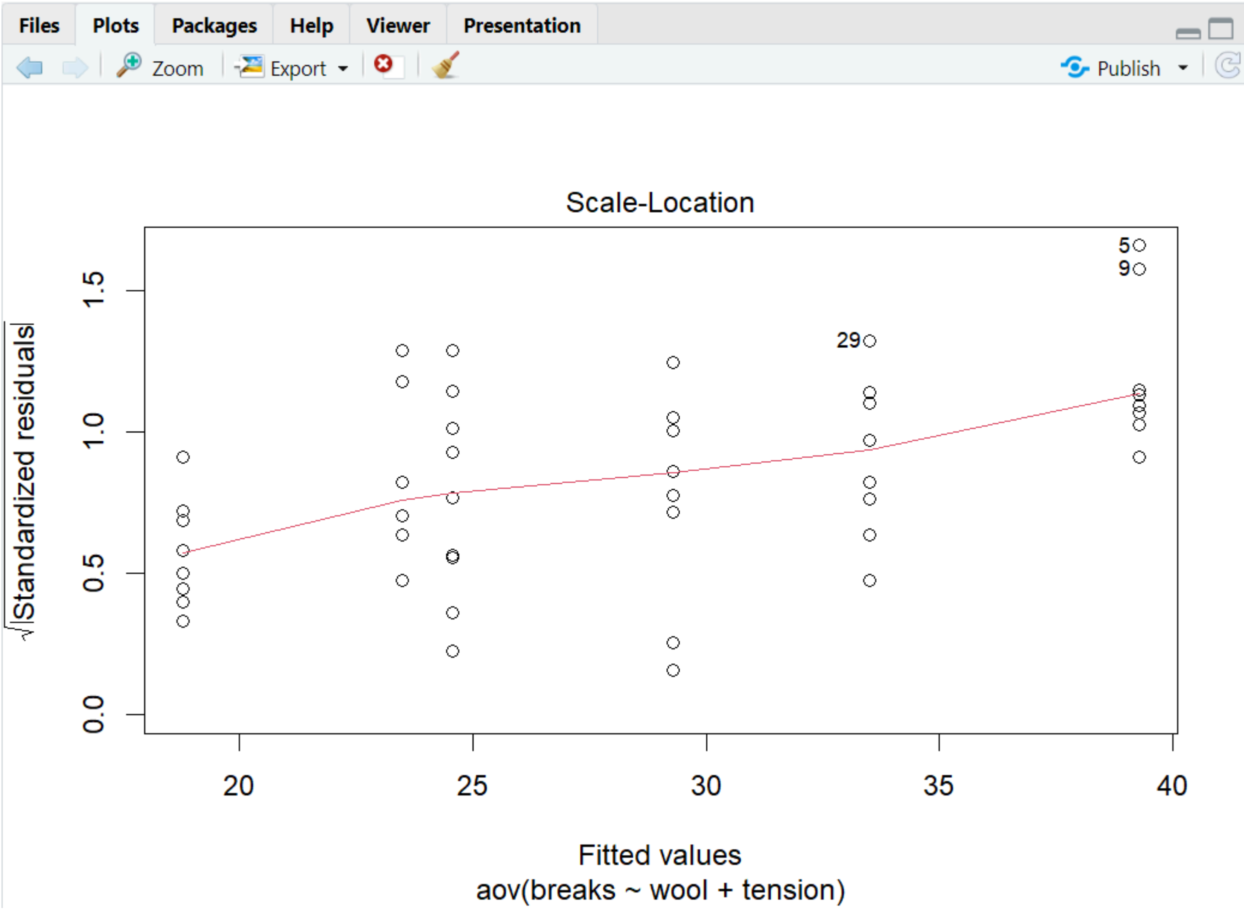
O/P:

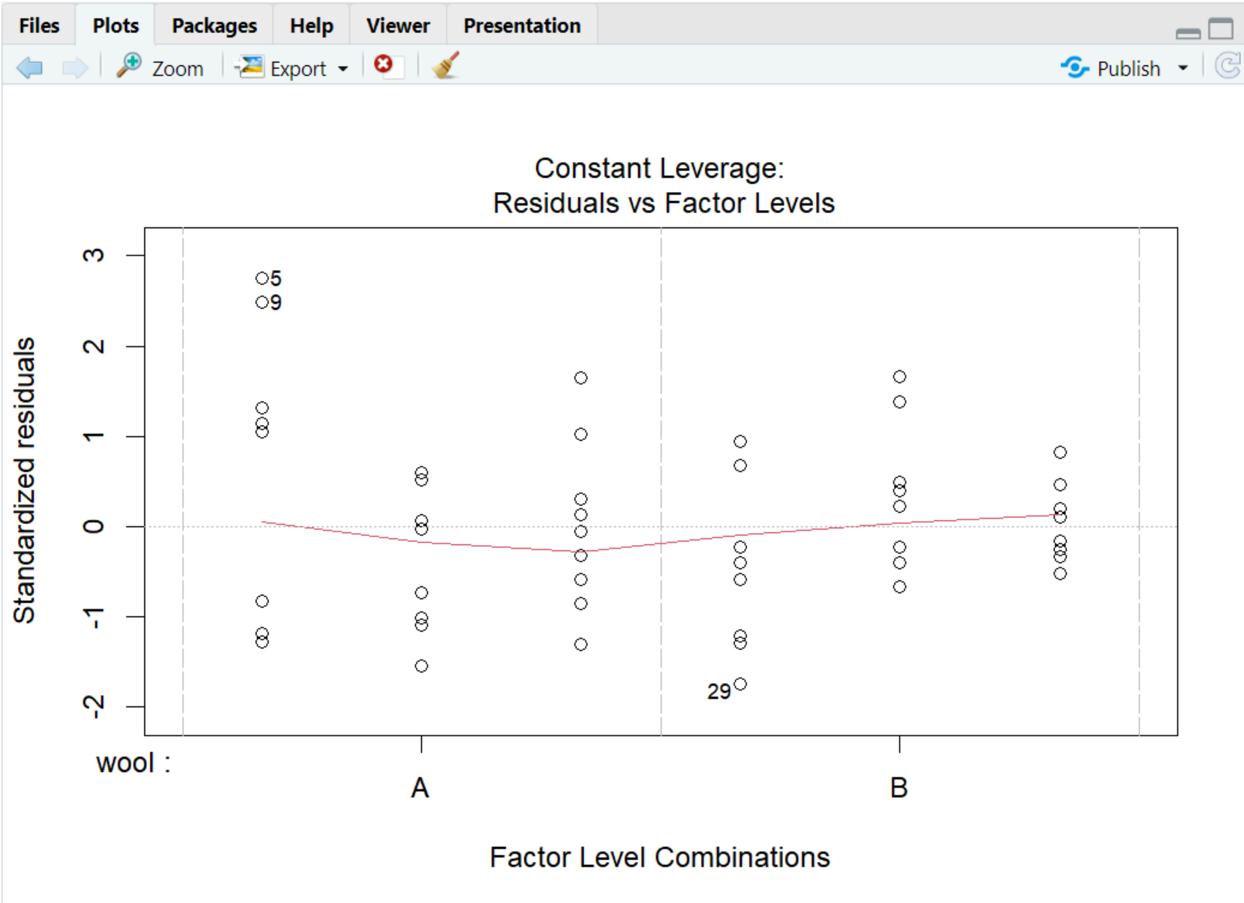
```
> #Data Exploration
> head(warpbreaks)
  breaks wool tension
1     26    A       L
2     30    A       L
3     54    A       L
4     25    A       L
5     70    A       L
6     52    A       L
> summary(warpbreaks)
     breaks      wool   tension
 Min.   :10.00   A:27   L:18
 1st Qu.:18.25   B:27   M:18
 Median :26.00          H:18
 Mean   :28.15
 3rd Qu.:34.00
 Max.   :70.00
> summary(Model_1)
            Df Sum Sq Mean Sq F value  Pr(>F)
wool         1    451   450.7   3.339 0.07361 .
tension      2   2034  1017.1   7.537 0.00138 **
Residuals   50   6748   135.0
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Residuals vs Fitted
Residuals
30
20
10
0
-10
-20
5
9
29
20
25
30
35
40
Fitted values
aov(breaks ~ wool + tension)

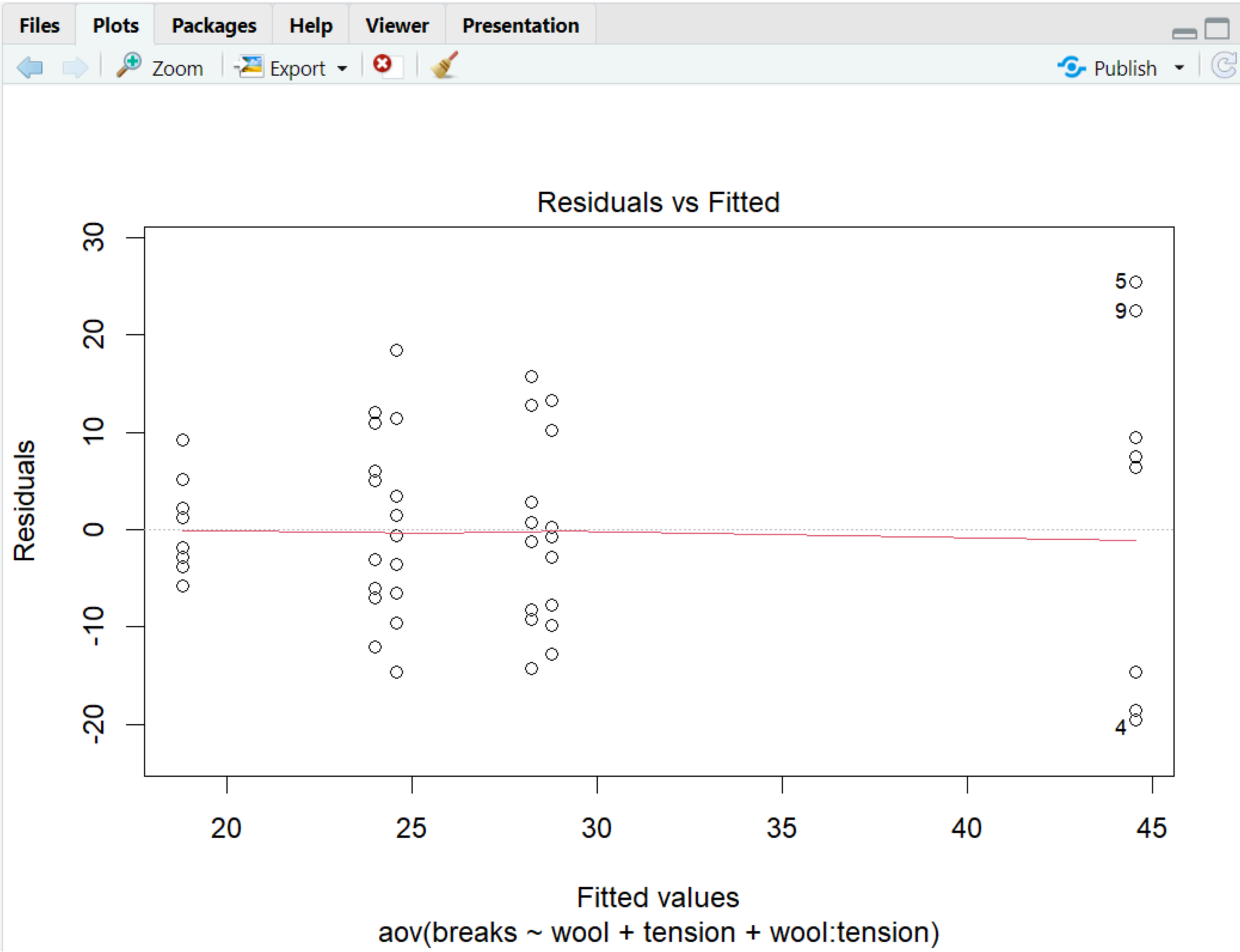


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Q-Q Residuals
Standardized residuals
Theoretical Quantiles
aov(breaks ~ wool + tension)
-2
-1
0
1
2
3
29
9
5

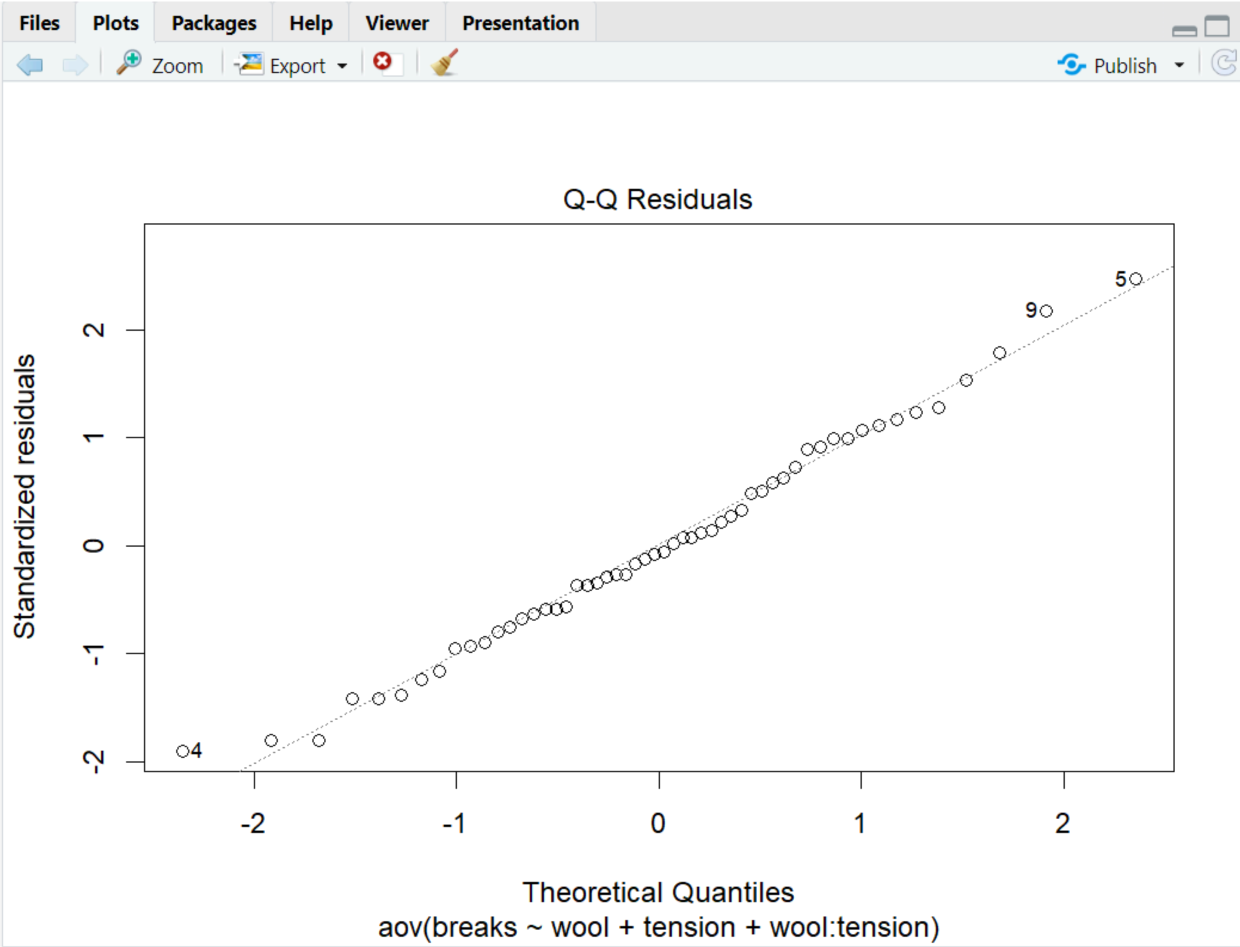

Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
20
25
30
35
40
5
9
29
Fitted values
aov(breaks ~ wool + tension)

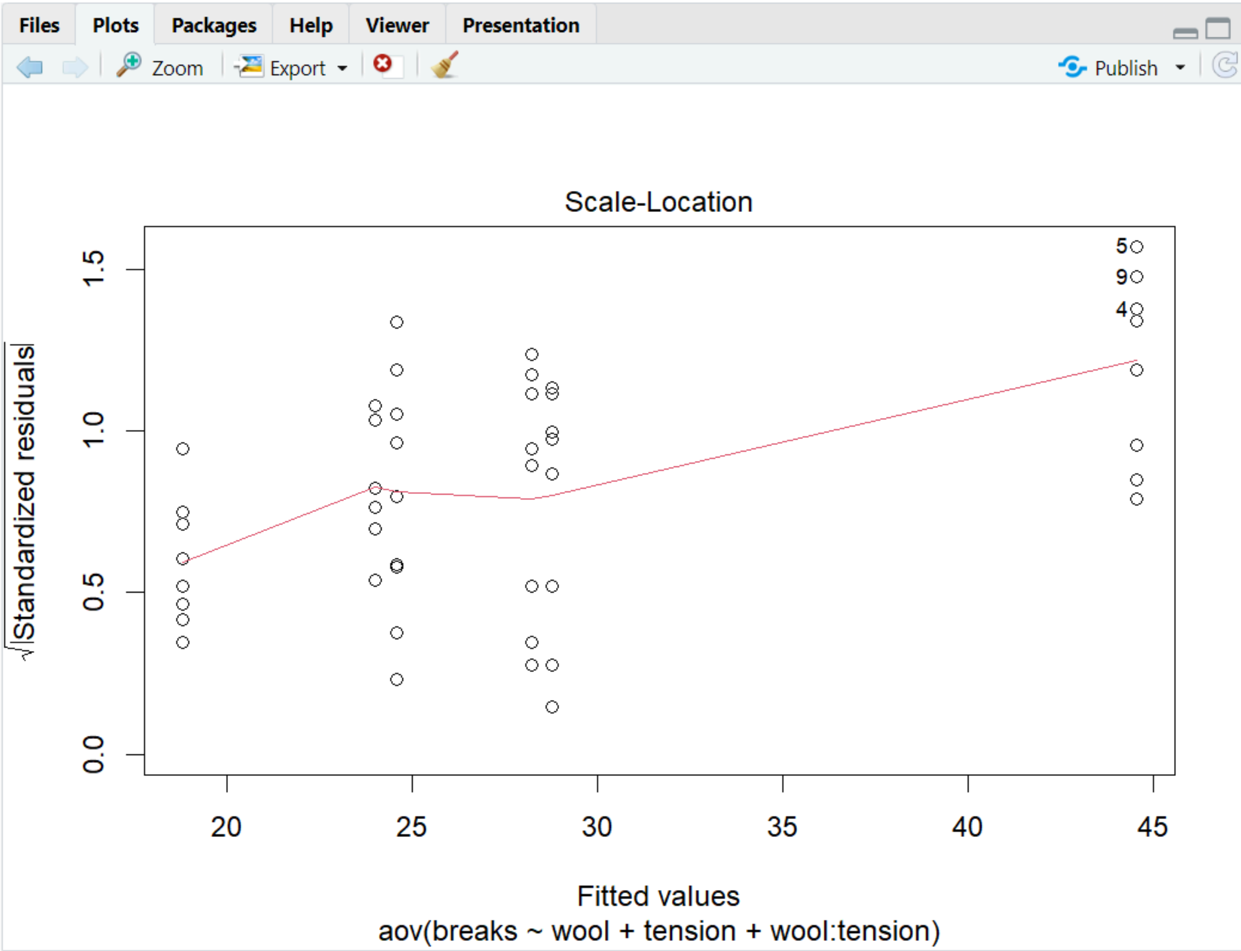


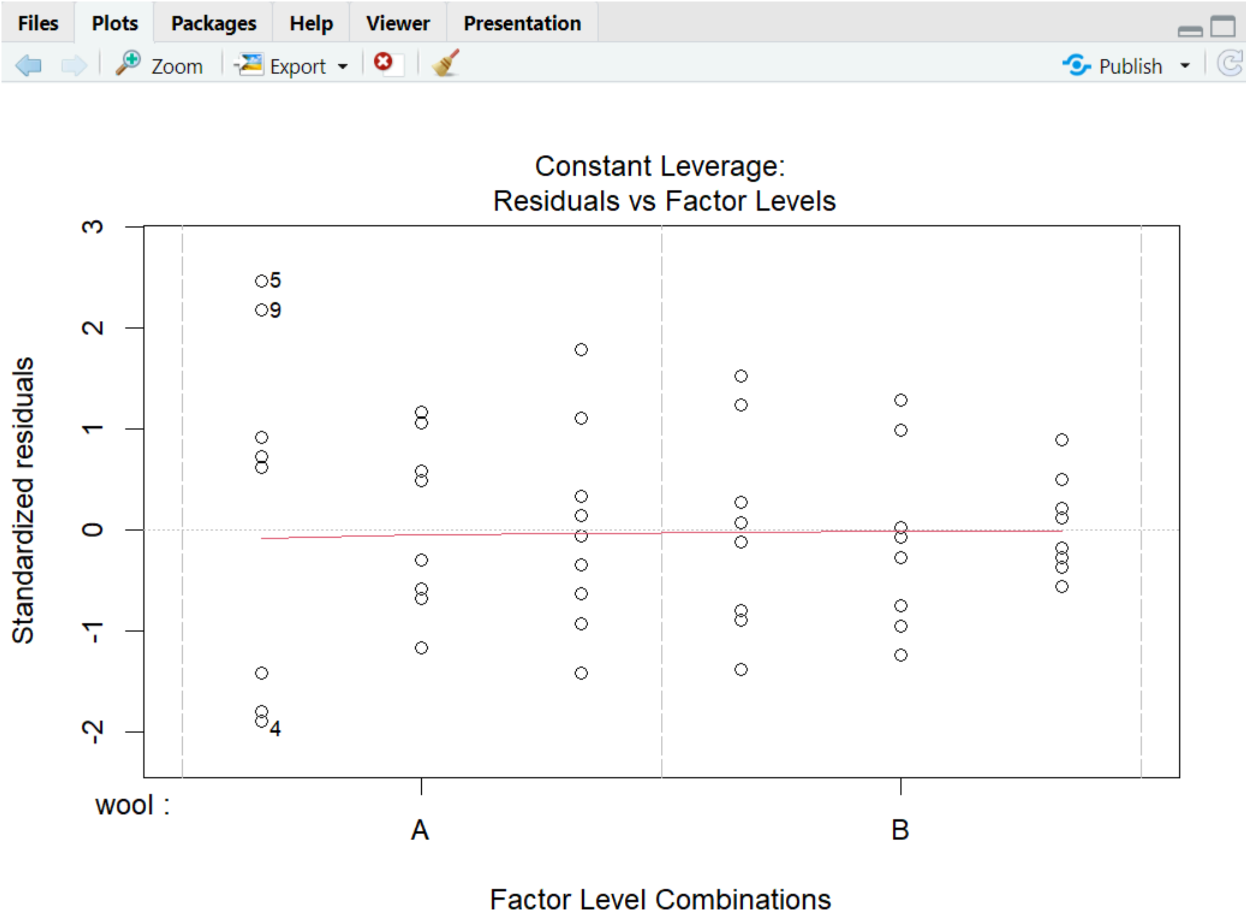

```
> summary(Model_2)
             Df Sum Sq Mean Sq F value   Pr(>F)    
wool          1    451   450.7   3.765 0.058213 .  
tension       2   2034  1017.1   8.498 0.000693 ***
wool:tension  2   1003   501.4   4.189 0.021044 *  
Residuals    48   5745   119.7                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> 
```



Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Residuals vs Fitted
Residuals
Fitted values
aov(breaks ~ wool + tension + wool:tension)
30
20
10
0
-10
-20
20
25
30
35
40
45
5
9
4


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Q-Q Residuals
Standardized residuals
Theoretical Quantiles
aov(breaks ~ wool + tension + wool:tension)
4
9
5
-2
-1
0
1
2


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
20
25
30
35
40
45
5
9
4
Fitted values
aov(breaks ~ wool + tension + wool:tension)



Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Constant Leverage:
Residuals vs Factor Levels
Standardized residuals
3
2
1
0
-1
-2
5
9
4
wool :
A
B
Factor Level Combinations

Q.2)

Code:

```
data("PlantGrowth")

#Data Exploration
head(PlantGrowth)
summary(PlantGrowth)

#Levels in PlantGrowth Group
levels(PlantGrowth$group)

#Extracting Variables
weight = PlantGrowth$weight
group = PlantGrowth$group

#Mean of weight
mean(weight)
mean(weight[group=="ctrl"])
mean(weight[group=="trt1"])
mean(weight[group=="trt2"])

#tapply() function

tapply(weight, group, mean)
tapply(weight, group, length)
```

O/P:

```
> #Data Exploration
> head(PlantGrowth)
  weight group
1   4.17  ctrl
2   5.58  ctrl
3   5.18  ctrl
4   6.11  ctrl
5   4.50  ctrl
6   4.61  ctrl
> summary(PlantGrowth)
     weight       group
 Min.   :3.590   ctrl:10
 1st Qu.:4.550   trt1:10
 Median :5.155   trt2:10
 Mean   :5.073
 3rd Qu.:5.530
 Max.   :6.310
> #Levels in PlantGrowth Group
> levels(PlantGrowth$group)
[1] "ctrl" "trt1" "trt2"
> #Extracting Variables
> weight = PlantGrowth$weight
> group = PlantGrowth$group
```

```
> #Mean of weight
> mean(weight)
[1] 5.073
> mean(weight[group=="ctrl"])
[1] 5.032
> mean(weight[group=="trt1"])
[1] 4.661
> mean(weight[group=="trt2"])
[1] 5.526
> tapply(weight, group, mean)
 ctrl  trt1  trt2
5.032 4.661 5.526
> tapply(weight, group, length)
ctrl trt1 trt2
  10   10   10
```

PRACTICAL-5
Linear Regression

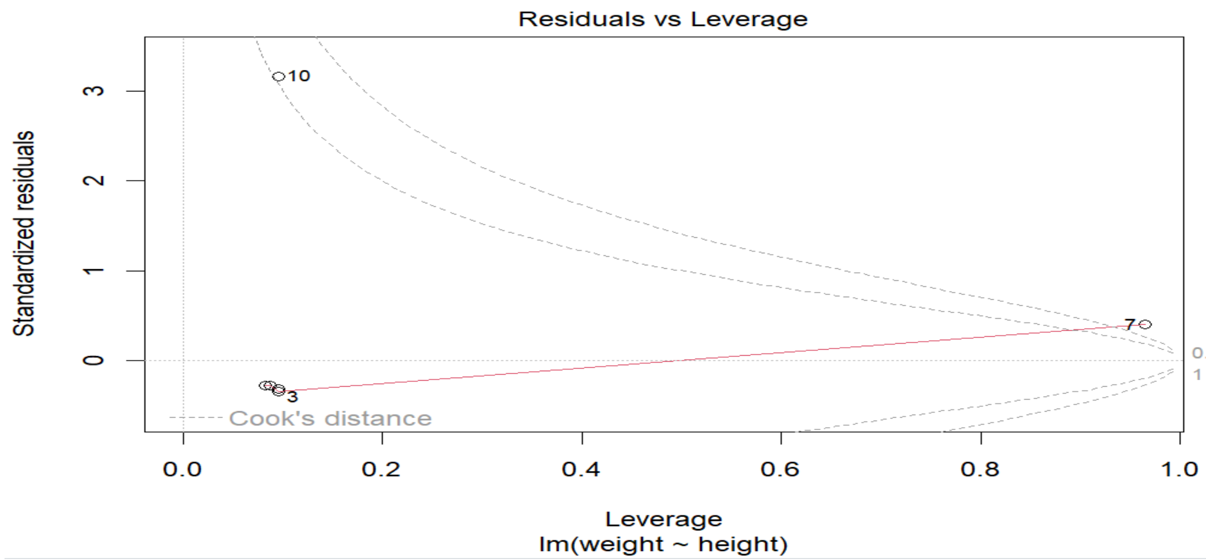
1.

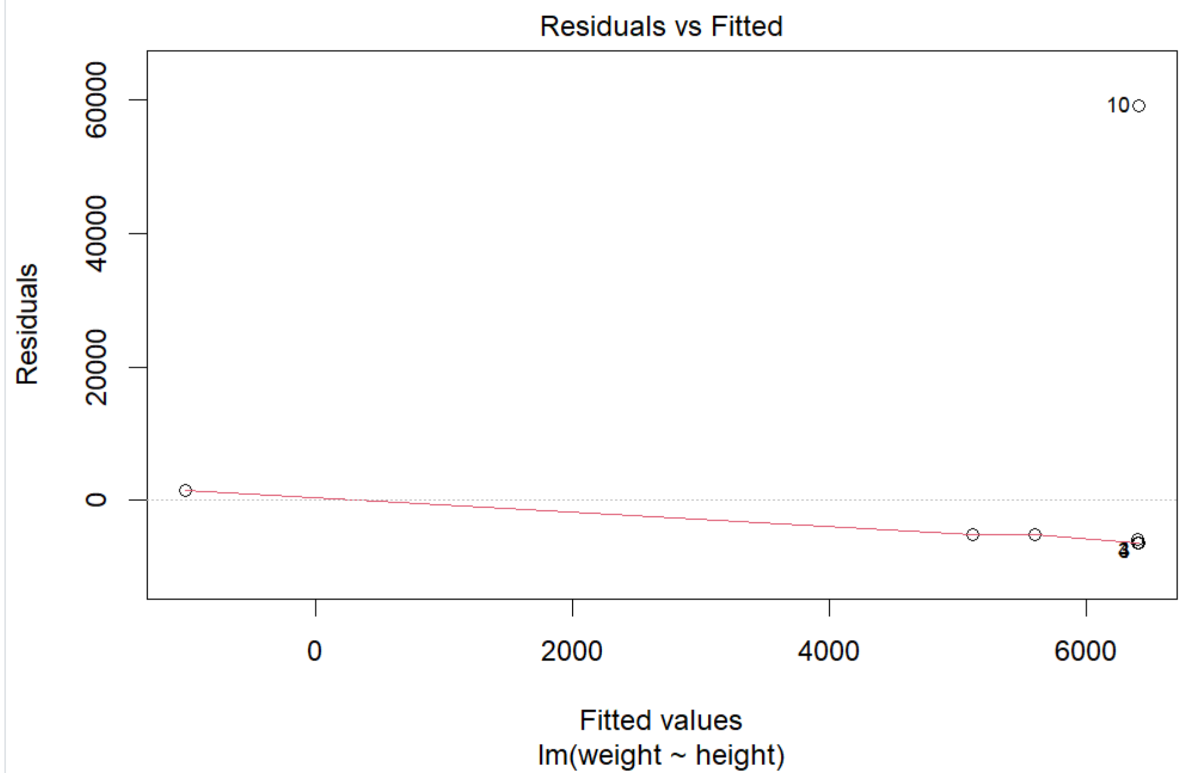
```
CODE:
height<- c(43,65,6,6,36,56,43556,43,7564,7,4764,75)
weight<- c(43,5,6,6,6,36,465,65,7,65467,547,647)
student<- lm(weight~height)
print(student)
```

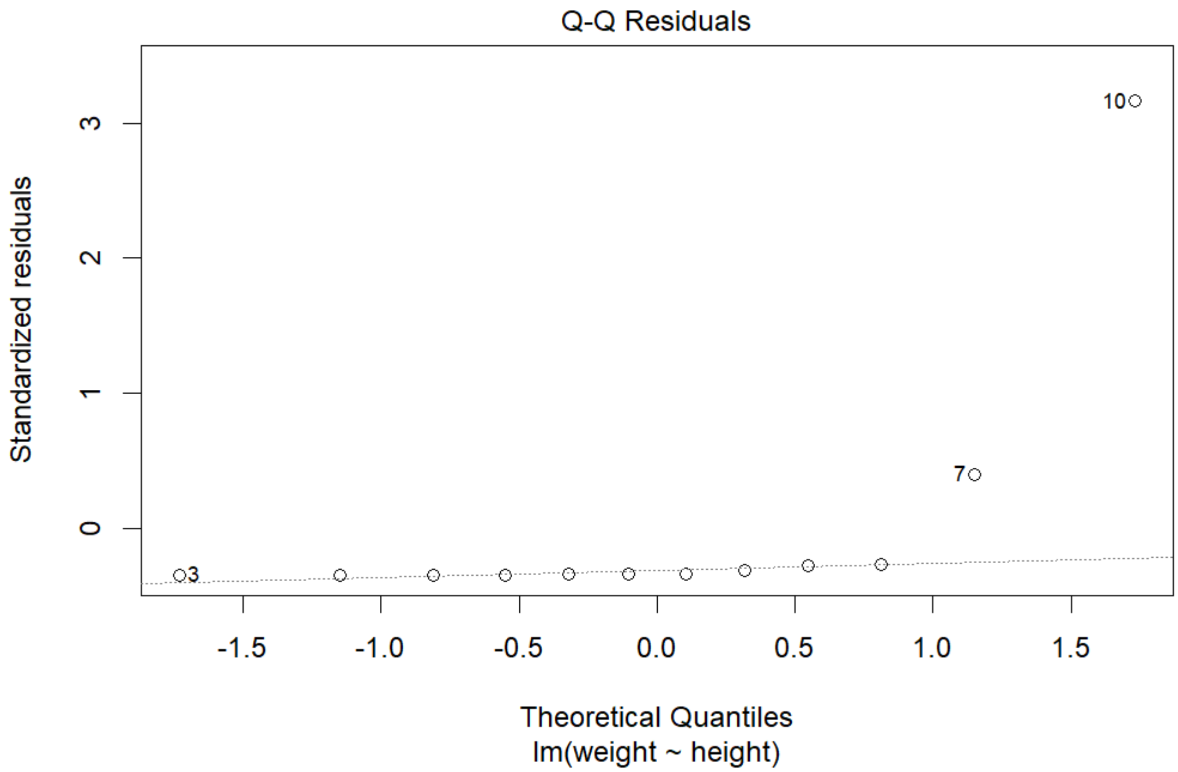
OUTPUT:

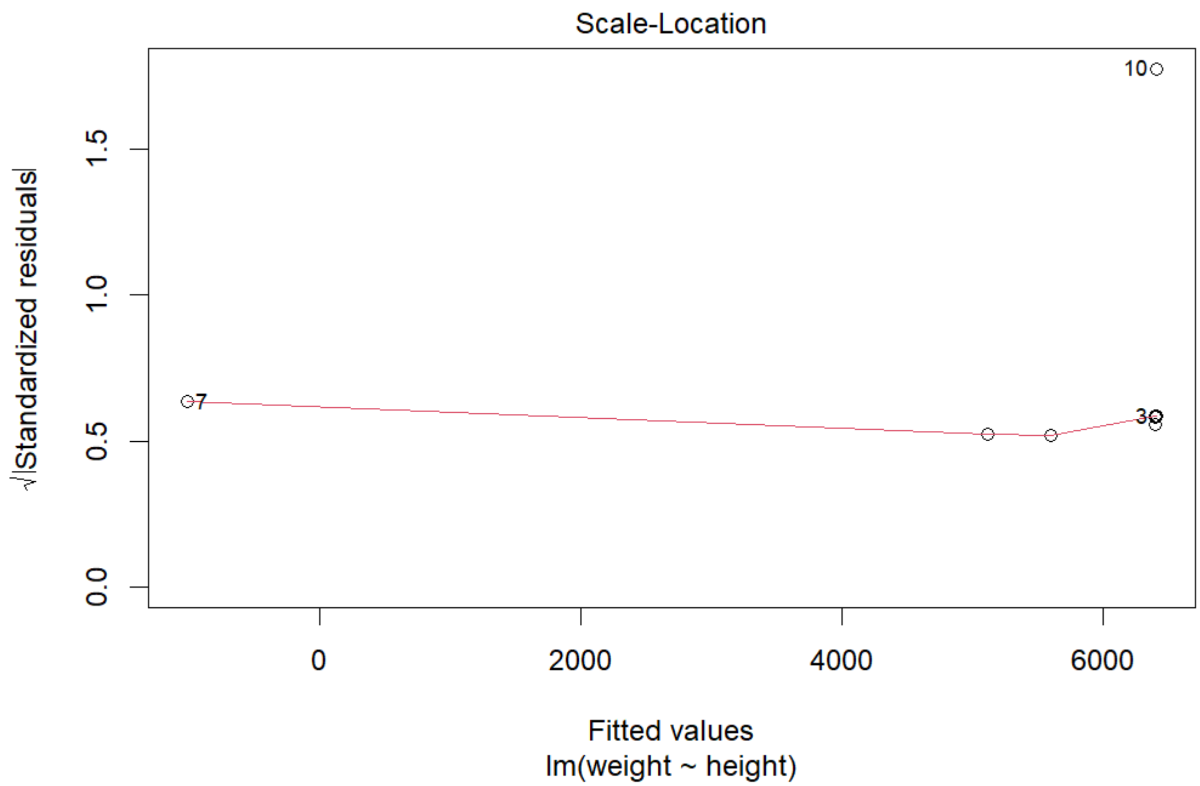
```
> height<- c(43,65,6,6,36,56,43556,43,7564,7,4764,75)
> weight<- c(43,5,6,6,6,36,465,65,7,65467,547,647)
> student<- lm(weight~height)
> print(student)

Call:
lm(formula = weight ~ height)

Coefficients:
(Intercept)       height
  6406.3248      -0.1703
```

plot(student)

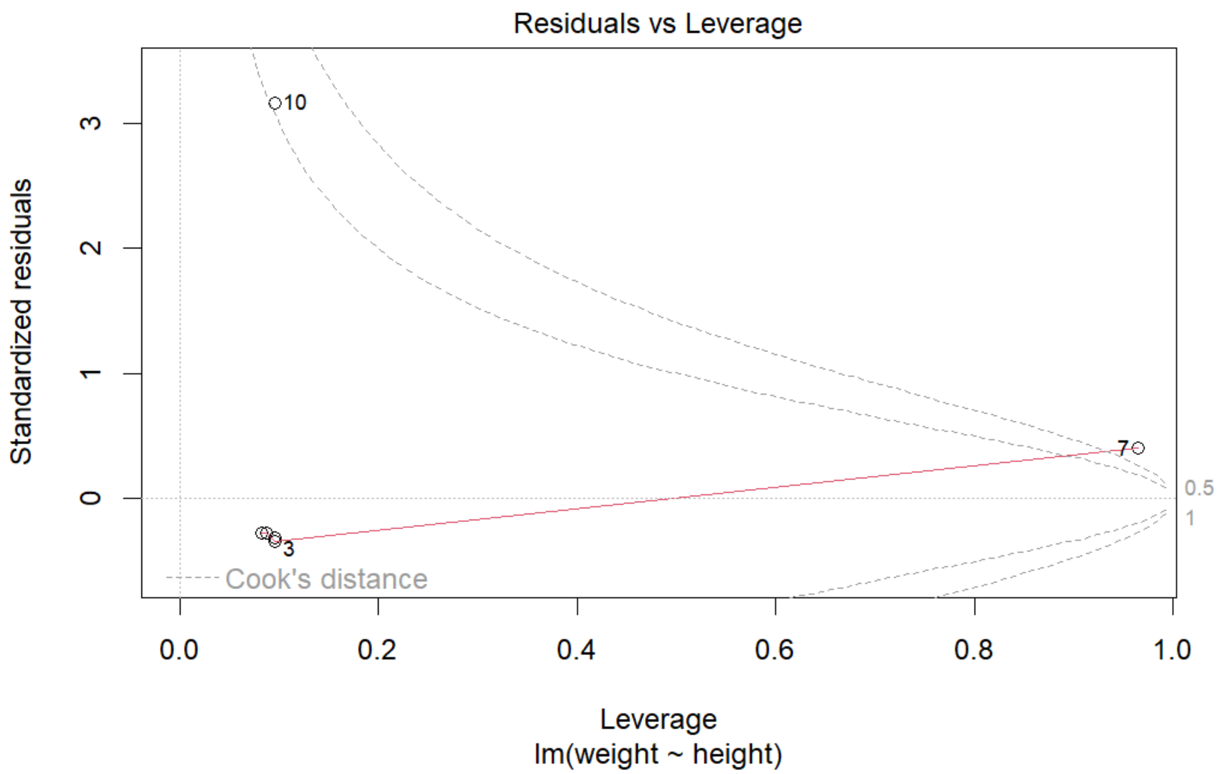

Residuals vs Leverage
Standardized residuals
3
2
1
0
10
7
3
0.5
1
Cook's distance
0.0
0.2
0.4
0.6
0.8
1.0
Leverage
lm(weight ~ height)


Residuals vs Fitted
Residuals
60000
40000
20000
0
10
0
2000
4000
6000
Fitted values
lm(weight ~ height)


Q-Q Residuals
Standardized residuals
3
2
1
0
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
Theoretical Quantiles
lm(weight ~ height)
10
7
3



Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
0
2000
4000
6000
Fitted values
lm(weight ~ height)
10
7
3



predict(student,data.frame(height = 199),interval ="confidence")

```
> predict(student,data.frame(height = 199),interval ="confidence")
       fit       lwr      upr
1 5856.637 -6336.768 18050.04
```

2.

```
df <- datasets::cars
my_linear_model <- lm(dist ~ speed, data = df)
print(my_linear_model)
```

```
> df <- datasets::cars
> my_linear_model <- lm(dist ~ speed, data = df)
> print(my_linear_model)

Call:
lm(formula = dist ~ speed, data = df)

Coefficients:
(Intercept)        speed
    -17.579        3.932
```

```
lm(formula = dist ~ speed,data = df)
```

```
> lm(formula = dist ~ speed,data = df)

Call:
lm(formula = dist ~ speed, data = df)

Coefficients:
(Intercept)        speed
    -17.579        3.932
```

3.

```
variable_speed <-
data.frame(speed=c(11,12,432,354,4,56,54,6,56))
```

```
linear_model <- lm(dist ~ speed,data = df)
predict(linear_model,newdata = variable_speed)
```

```
> variable_speed <- data.frame(speed=c(11,12,432,354,4,56,54,6,56))
> linear_model <- lm(dist ~ speed,data = df)
> predict(linear_model,newdata = variable_speed)
          1           2           3           4           5           6 
  25.677401   29.609810 1681.221489 1374.493606   -1.849460  202.635796 
          7           8           9 
 194.770978    6.015358  202.635796 
```

```
predict(linear_model,newdata = variable_speed,interval =
"confidence")
```

```
> predict(linear_model,newdata = variable_speed,interval = "confidence")
          fit         lwr         upr
1   25.677401   19.964525   31.390278
2   29.609810   24.395138   34.824483
3 1681.221489 1333.147866 2029.295112
4 1374.493606 1091.578321 1657.408891
5   -1.849460  -12.329543    8.630624
6  202.635796  168.436003  236.835589
7  194.770978  162.227656  227.314300
8    6.015358   -2.973341   15.004056
9  202.635796  168.436003  236.835589
```

PRACTICAL 6

HYPOTHESIS TESTING

Code:

```
x <- rnorm(100)
t.test(x,mu=5)
```

output:

```
> x <- rnorm(100)
> t.test(x,mu=5)

        One Sample t-test

data:  x
t = -53.473, df = 99, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 5
95 percent confidence interval:
 -0.2719046  0.1053460
sample estimates:
  mean of x
-0.08327929
```

—--

2 sample testing:

```
Code:
x <- rnorm(100)
y <- rnorm(100)
t.test(x,y)
```

Output:

```
> x <- rnorm(100)
> y <- rnorm(100)
> t.test(x,y)

        Welch Two Sample t-test

data:  x and y
t = -0.11474, df = 197.94, p-value = 0.9088
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3064858  0.2727830
sample estimates:
 mean of x  mean of y
0.06604895 0.08290039
```

Directional hypothesis

```
Code:
x <- rnorm(100)
t.test(x,mu=2, alternative='greater')
```

Output:

```
> x <- rnorm(100)
> y <- rnorm(100)
> t.test(x,y)

        Welch Two Sample t-test

data:  x and y
t = -0.11474, df = 197.94, p-value = 0.9088
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3064858  0.2727830
sample estimates:
 mean of x  mean of y
0.06604895 0.08290039
```

```
> x <- rnorm(100)
> t.test(x,mu=2, alternative='greater')

        One Sample t-test

data:  x
t = -18.533, df = 99, p-value = 1
alternative hypothesis: true mean is greater than 2
95 percent confidence interval:
 -0.1586271        Inf
sample estimates:
 mean of x
0.01886732
```

Code:

```
dataf<-seq(1,20,by=1)
dataf
mean(dataf)
sd(dataf)
a<-t.test(dataf,alternate="two-sided",mu=10,conf.int=0.95)
a
```

Output:

```
        One Sample t-test

data:  dataf
t = 0.37796, df = 19, p-value = 0.7096
alternative hypothesis: true mean is not equal to 10
95 percent confidence interval:
  7.731189 13.268811
sample estimates:
mean of x
     10.5
```

PRACTICAL NO 7

DECISION TREE

Cmd:

```
install.packages("party")
library(party)
```

Code:

```
print(head(readingSkills))
```

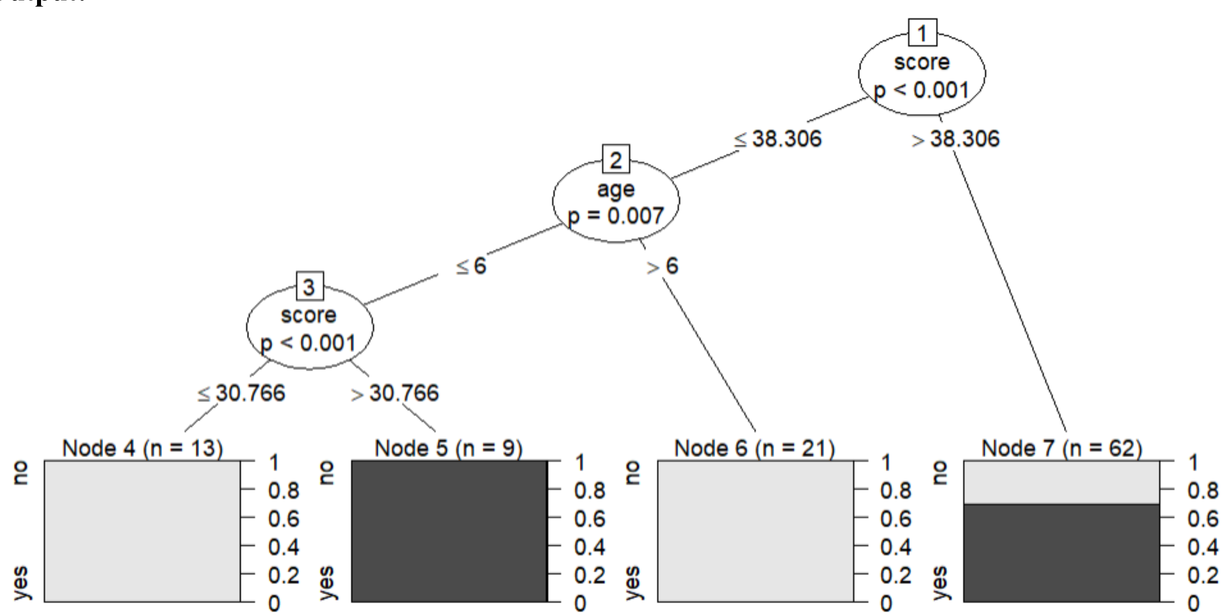
Output:

```
> print(head(readingSkills))
  nativeSpeaker age shoeSize    score
1           yes   5 24.83189 32.29385
2           yes   6 25.95238 36.63105
3            no  11 30.42170 49.60593
4           yes   7 28.66450 40.28456
5           yes  11 31.88207 55.46085
6           yes  10 30.07843 52.83124
>
```

Code:

```
input.dat<-readingSkills[c(1:105),]
output.tree<-ctree(nativeSpeaker ~ age+shoeSize +score,data=input.dat)
plot(output.tree)
```

Output:



PRACTICAL NO 8

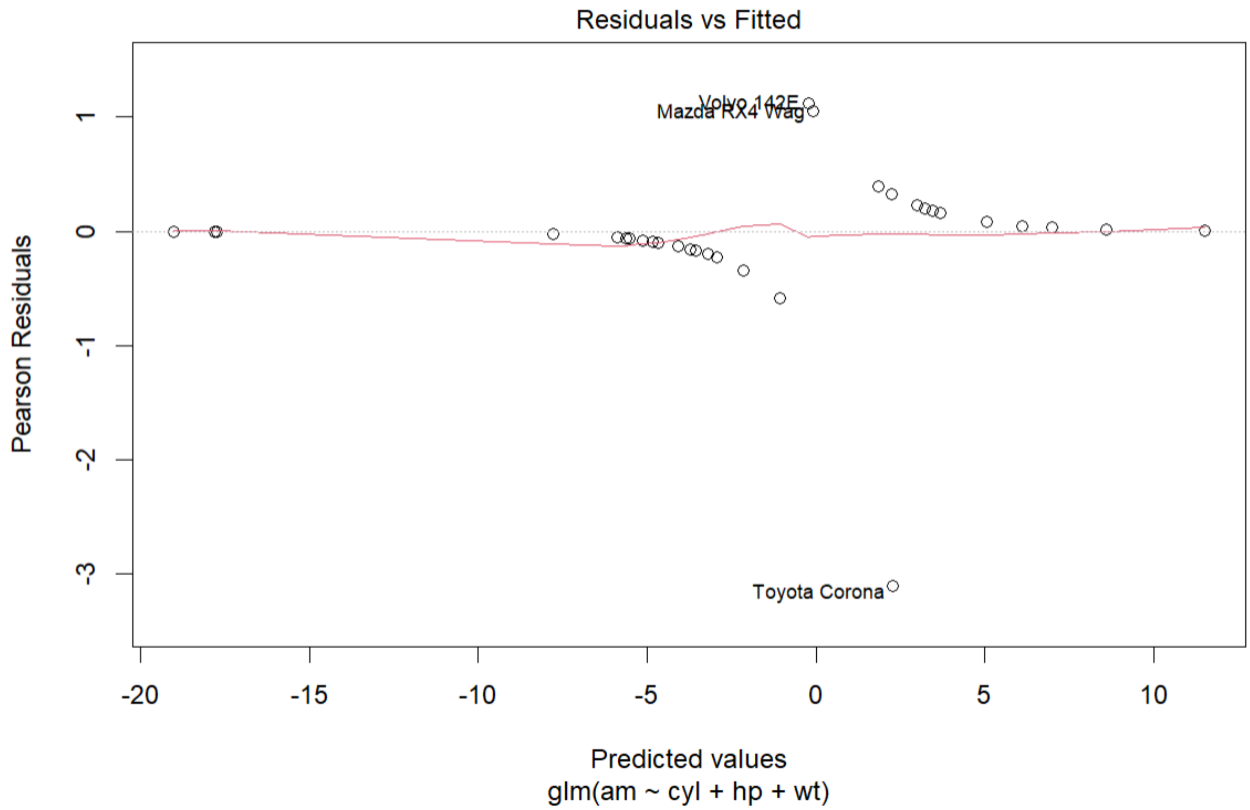
LOGISTIC REGRESSION (glm func)

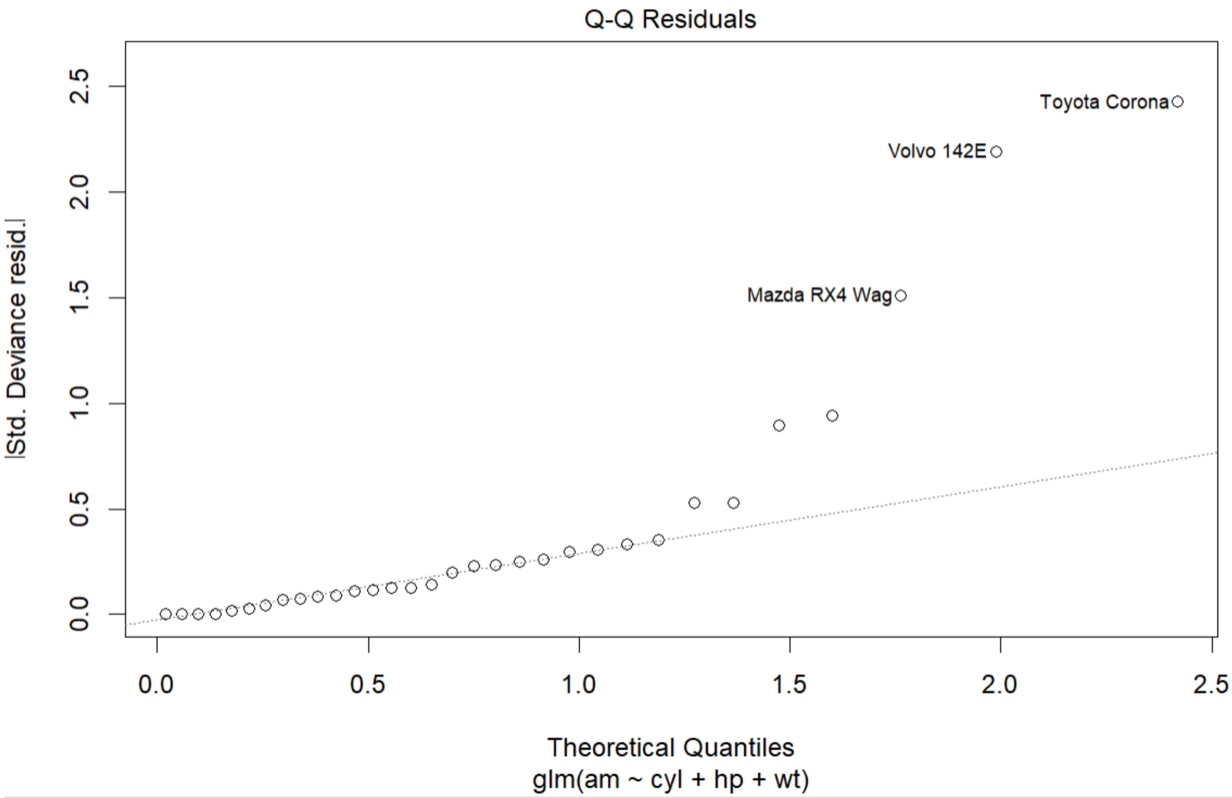
Code:

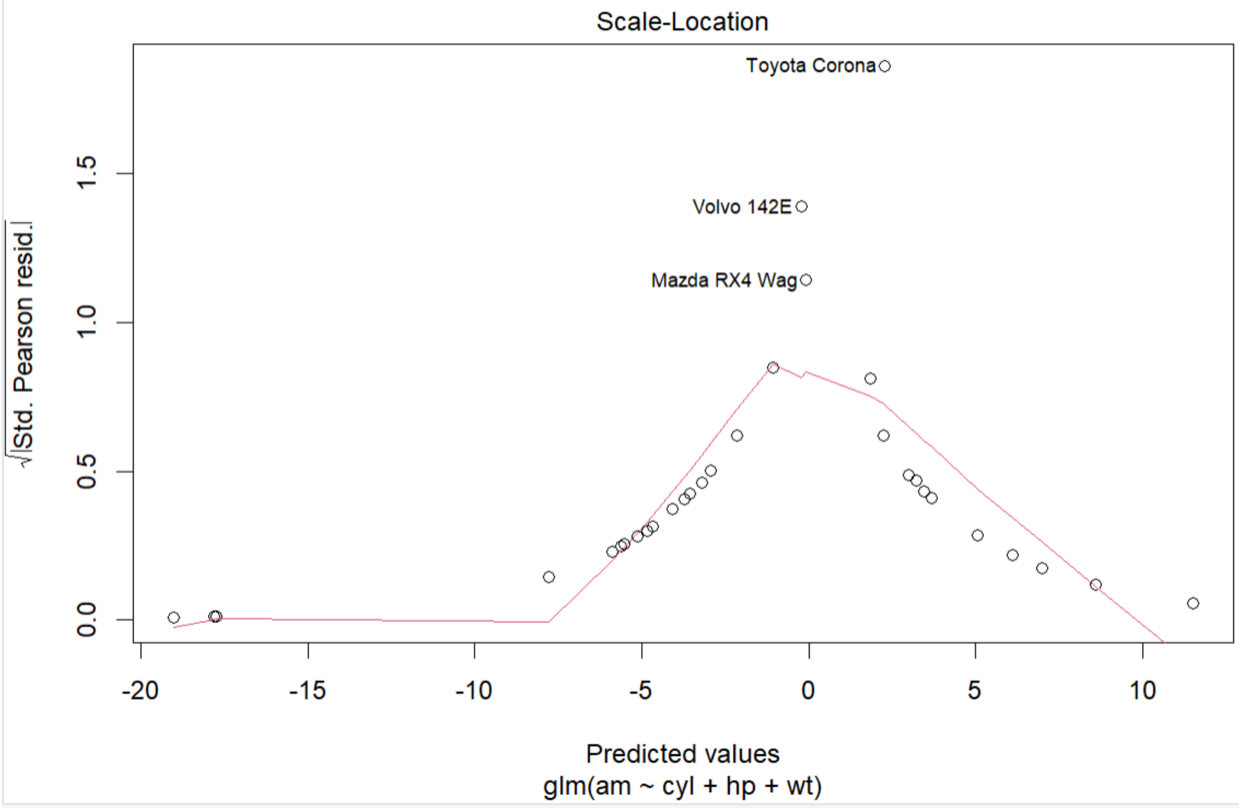
```
input <-mtcars[,c("am","cyl","hp","wt")]
am.data=glm(formula=am~cyl+hp+wt,data=input,family=binomial)
print(summary(am.data))
```

Output:

```
> input <-mtcars[,c("am","cyl","hp","wt")]
> am.data=glm(formula=am~cyl+hp+wt,data=input,family=binomial)
> print(summary(am.data))

Call:
glm(formula = am ~ cyl + hp + wt, family = binomial, data = input)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)  
(Intercept) 19.70288    8.11637   2.428   0.0152 *
cyl          0.48760    1.07162   0.455   0.6491  
hp           0.03259    0.01886   1.728   0.0840 .
wt          -9.14947    4.15332  -2.203   0.0276 *
---
Signif. codes:  
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.2297  on 31  degrees of freedom
Residual deviance:  9.8415  on 28  degrees of freedom
AIC: 17.841

Number of Fisher Scoring iterations: 8
```


Residuals vs Fitted
Volvo 142E
Mazda RX4 Wag
Toyota Corona
Pearson Residuals
1
0
-1
-2
-3
-20
-15
-10
-5
0
5
10
Predicted values
glm(am ~ cyl + hp + wt)


Q-Q Residuals
Toyota Corona
Volvo 142E
Mazda RX4 Wag
|Std. Deviance resid.|
0.0
0.5
1.0
1.5
2.0
2.5
Theoretical Quantiles
glm(am ~ cyl + hp + wt)


Scale-Location
Toyota Corona
Volvo 142E
Mazda RX4 Wag
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
-20
-15
-10
-5
0
5
10
Predicted values
glm(am ~ cyl + hp + wt)

PRACTICAL 9

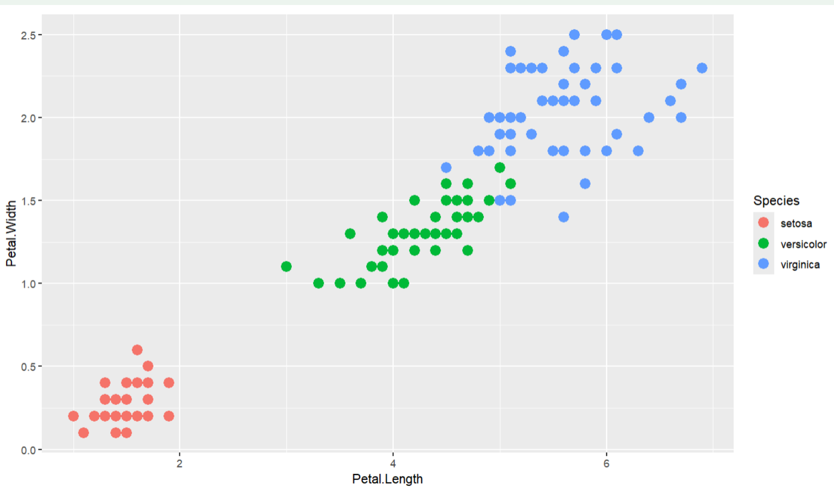
K MEANS CLUSTERING

```
install.packages("ggplot2")
Code:
library(ggplot2)
df<-iris
head(iris)
```

Output:

```
> df<-iris
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
library(ggplot2)
df<-iris
head(iris)
ggplot(df,aes(Petal.Length,Petal.Width))+geom_point(aes(col=Species),size=4)
```



Code:

```
set.seed(101)
irisCluster<-kmeans(df[,1:4],center=3,nstart=20)
irisCluster
```

```
K-means clustering with 3 clusters of sizes 38, 62, 50

Cluster means:
  Sepal.Length Sepal.Width
1     6.850000    3.073684
2     5.901613    2.748387
3     5.006000    3.428000
  Petal.Length Petal.Width
1     5.742105    2.071053
2     4.393548    1.433871
3     1.462000    0.246000

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [15] 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [29] 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [43] 3 3 3 3 3 3 3 3 2 2 1 2 2 2
 [57] 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [71] 2 2 2 2 2 2 2 1 2 2 2 2 2 2
 [85] 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [99] 2 2 1 2 1 1 1 1 2 1 1 1 1 1
[113] 1 2 2 1 1 1 1 2 1 2 1 2 1 1
[127] 2 2 1 1 1 1 1 2 1 1 1 1 2 1
[141] 1 1 2 1 1 1 2 1 1 2

Within cluster sum of squares by cluster:
[1] 23.87947 39.82097 15.15100
 (between_SS / total_SS =  88.4 %)
```

```
Available components:

[1] "cluster"      "centers"
[3] "totss"        "withinss"
[5] "tot.withinss" "betweenss"
[7] "size"         "iter"
[9] "ifault"
```

install.packages("cluster")

Code:

```
library(cluster)
clusplot(iris,irisCluster$cluster,color=T,shade=T,labels=0,lines=0)
```