Date: 11/Jan/2024	PARTITION BASED CLUSTERING
EXPERIMENT – 01	

AIM: To perform partition based clustering and understand about clustering

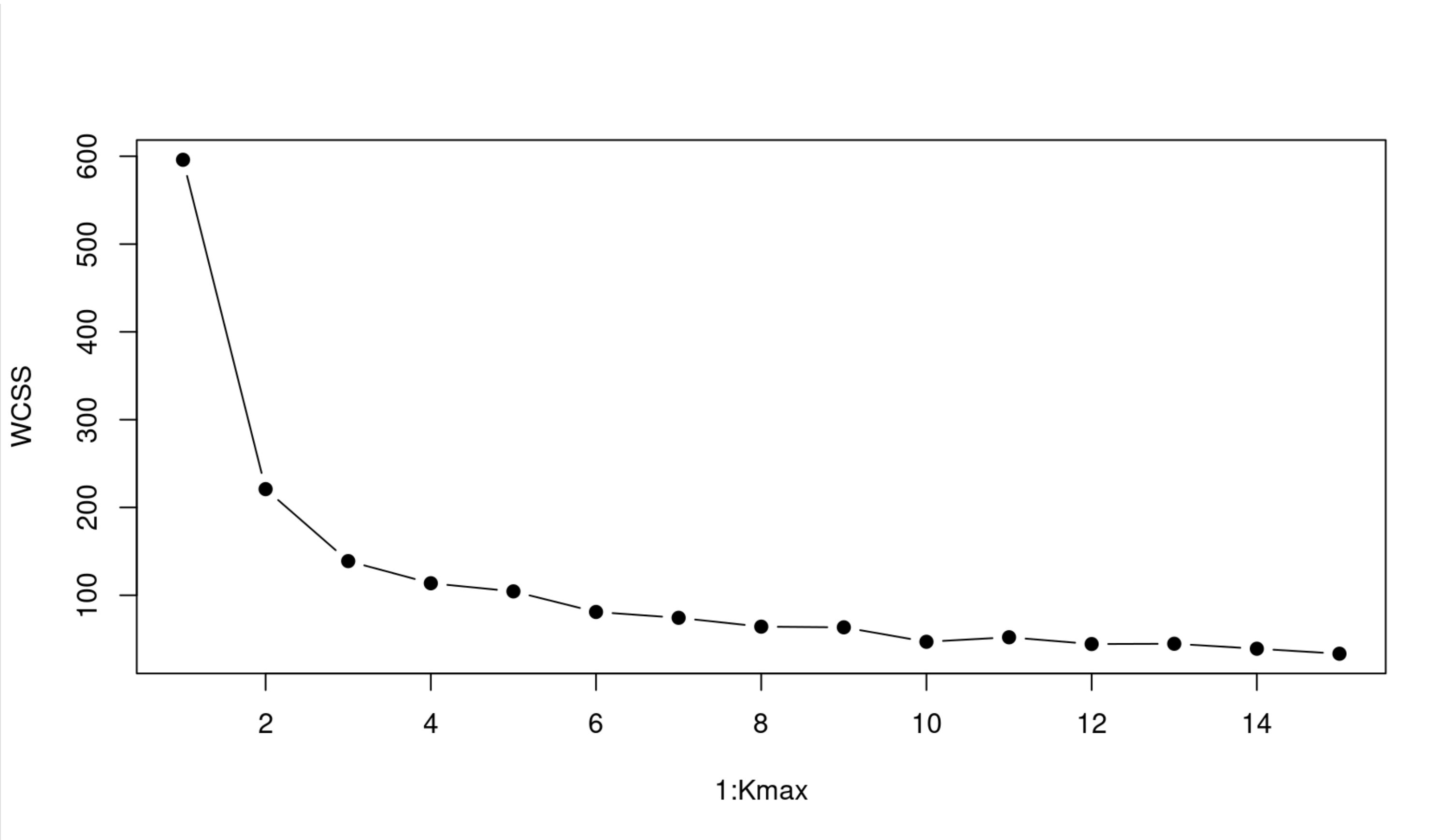
SOFTWARE REQUIRED: RStudio

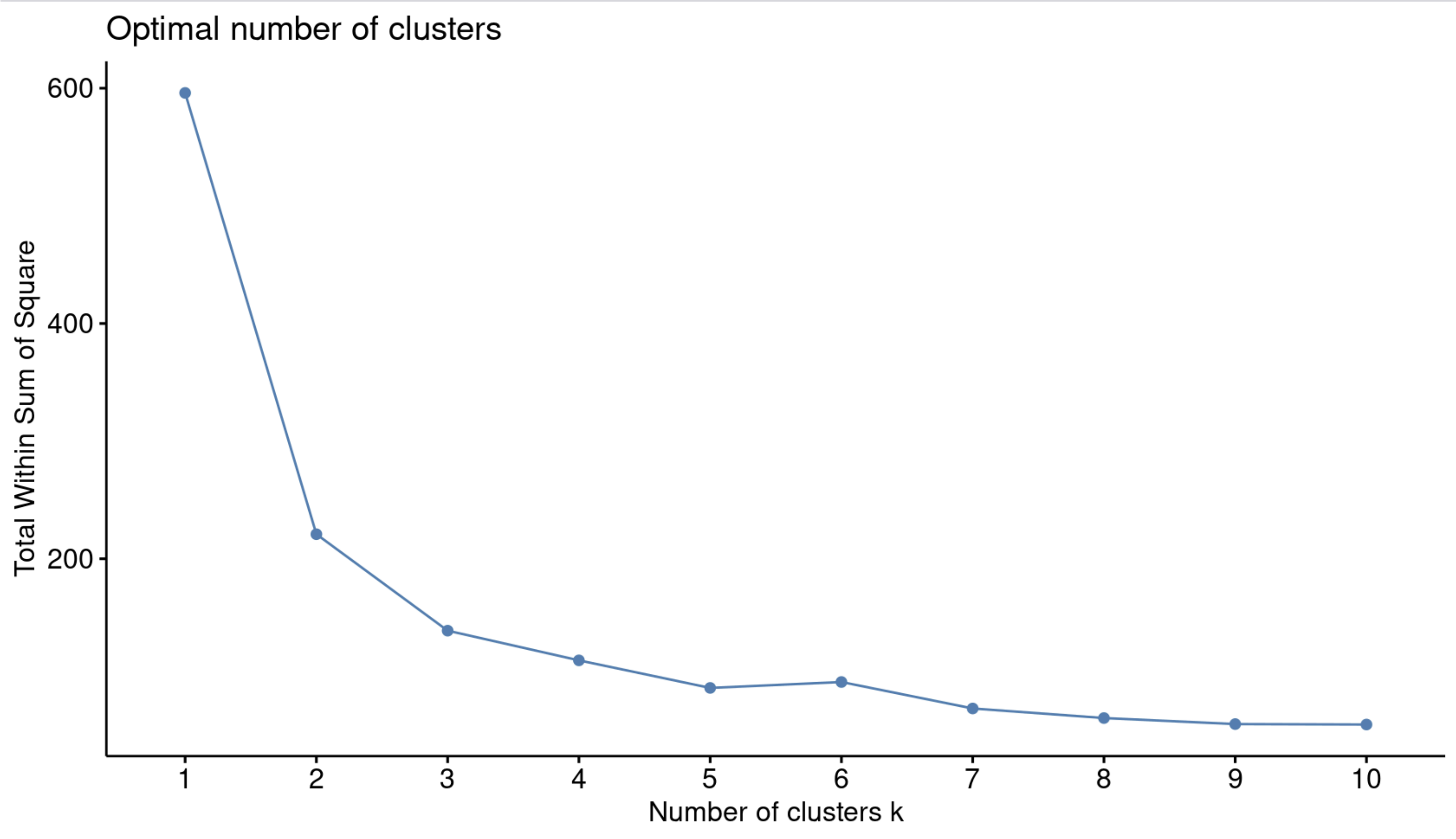
R CODE:

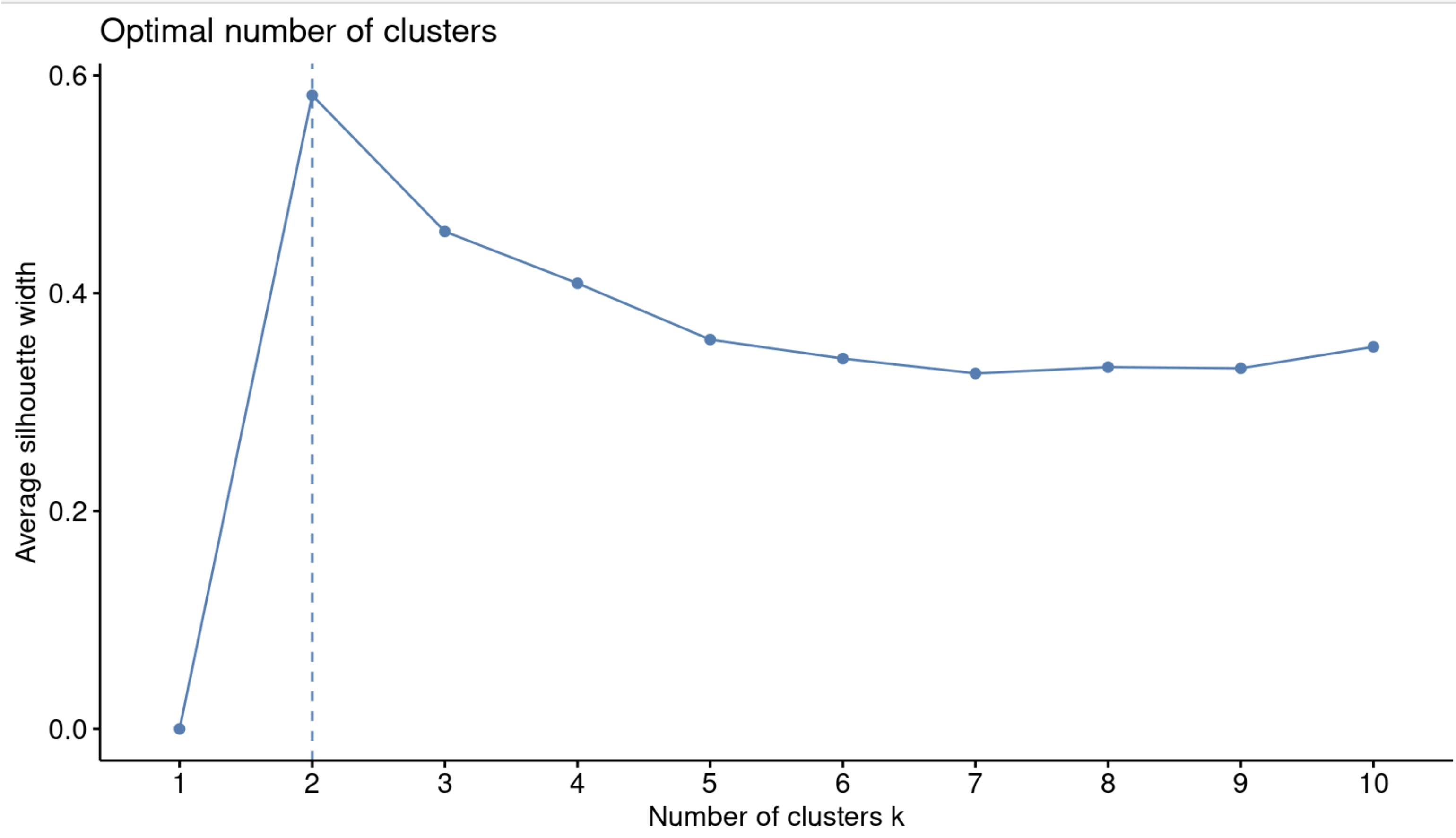
```
data <- read.csv("iris.csv", row.names = 1)
df <- scale(data)
set.seed(112)
fit <- kmeans(df,3)
fit$size
fit$withinss
fit$tot.withinss
Kmax <- 15
WCSS <- rep(NA, Kmax)
nClust <- list()
for (i in 1:Kmax) {
  fit <- kmeans(df, i)
  WCSS[i] <- fit$tot.withinss
  nClust[[i]] <- fit$size
}
plot(1:Kmax, WCSS, type = "b", pch=19)
library(factoextra)
fviz_nbclust(df, kmeans, method = "wss")
library(cluster)
fit <- pam(df, 3, metric = "manhattan")
fviz_nbclust(df, pam, method = "silhouette")
```

OUTPUT:

```
Console   Terminal ×   Background Jobs ×
R 4.3.2 · /cloud/project/
> data <- read.csv("iris.csv", row.names = 1)
> df <- scale(data)
> set.seed(112)
> fit <- kmeans(df,3)
> fit$size
[1] 47 53 50
> fit$withinss
[1] 47.45019 44.08754 47.35062
> fit$tot.withinss
[1] 138.8884
> Kmax <- 15
> WCSS <- rep(NA, Kmax)
> nClust <- list()
> for (i in 1:Kmax) {
+   fit <- kmeans(df, i)
+   WCSS[i] <- fit$tot.withinss
+   nClust[[i]] <- fit$size
+ }
> plot(1:Kmax, WCSS, type = "b", pch=19)
> library(factoextra)
> fviz_nbclust(df, kmeans, method = "wss")
> library(cluster)
> fit <- pam(df, 3, metric = "manhattan")
> fviz_nbclust(df, pam, method = "silhouette")
```




Optimal number of clusters
600
400
200
Total Within Sum of Square
1
2
3
4
5
6
7
8
9
10
Number of clusters k


Optimal number of clusters
0.6
0.4
0.2
0.0
Average silhouette width
1
2
3
4
5
6
7
8
9
10
Number of clusters k

Date: 18/Jan/2024	HIERARCHICAL CLUSTERING
EXPERIMENT – 02	

AIM: To perform hierarchical clustering

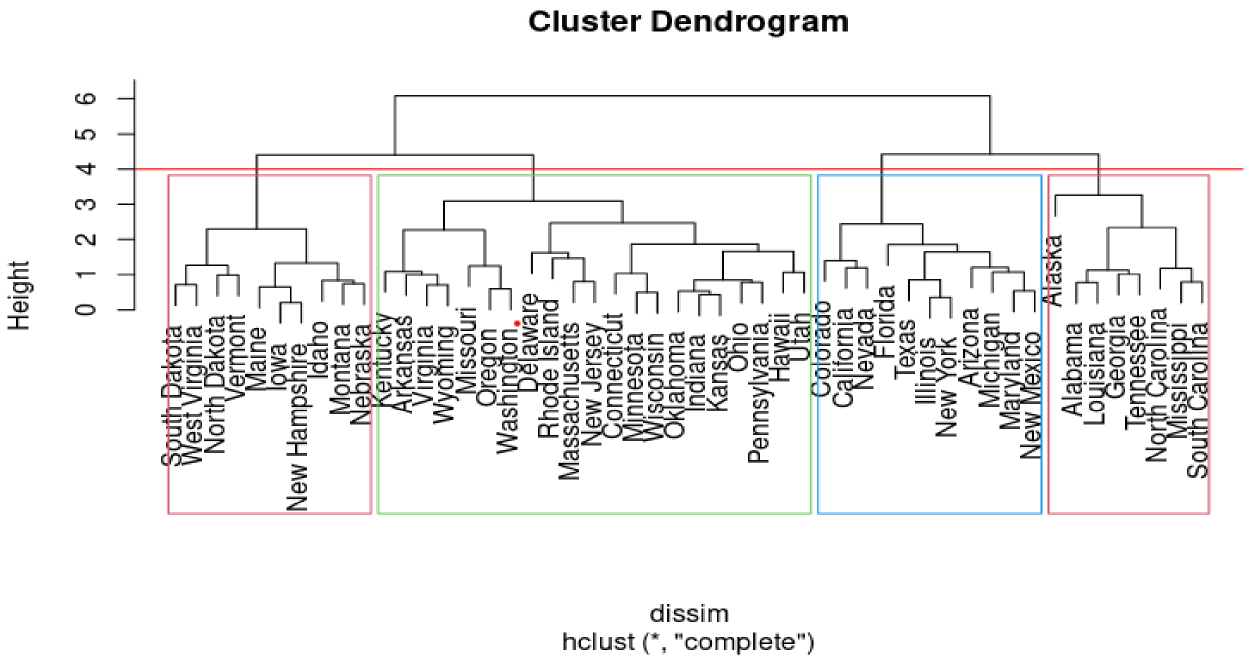
SOFTWARE REQUIRED: RStudio

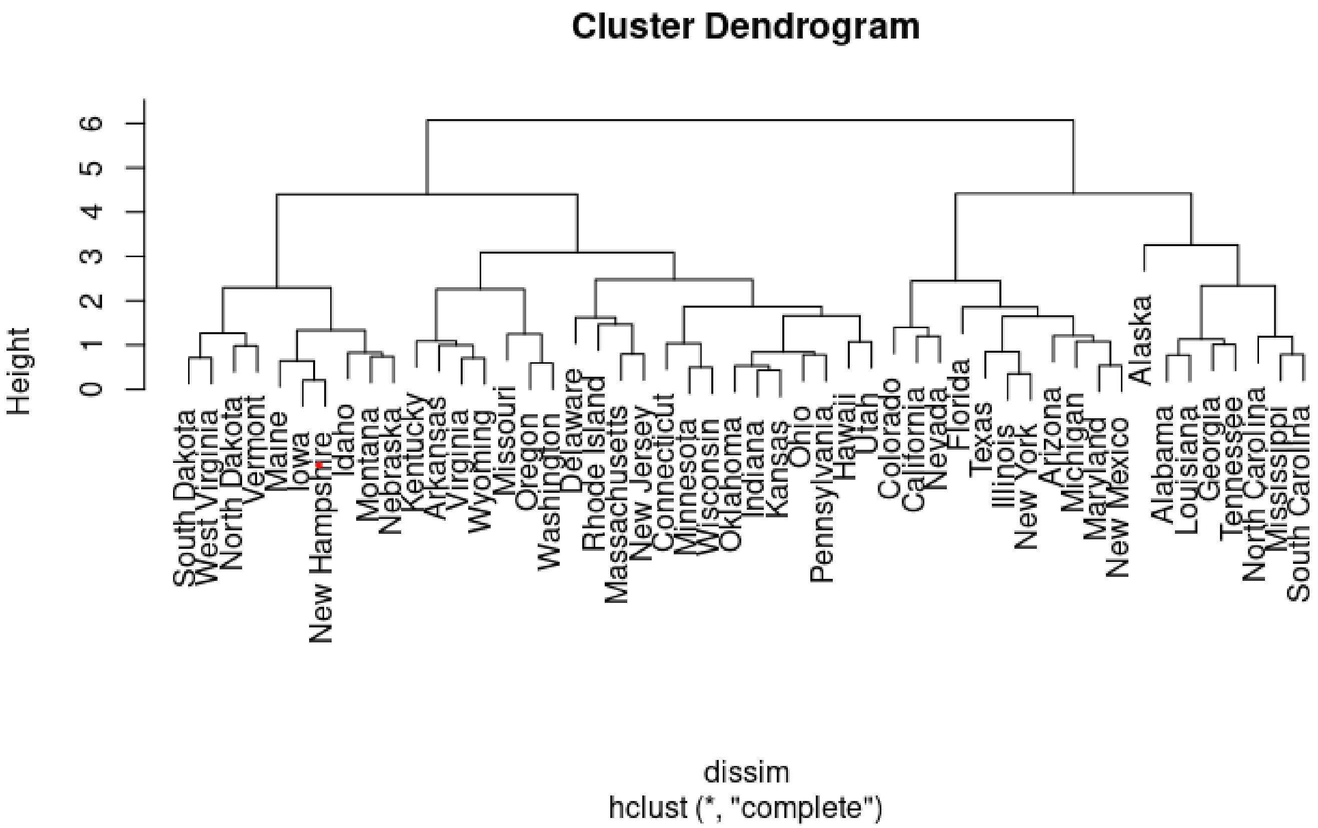
R CODE:

```
rm(list=ls())
data<-read.csv("USArrests.csv",row.names=1)
df
df<-scale(data)
dissim<-dist(df,method='euclidean')
hierClust<-hclust(dissim,method='complete')
plot(hierClust)

cluster<-cutree(hierClust,k=4)

library(clValid)

dunn(dissim,cluster)

rect.hclust(hierClust,k=4,border=2:4)

abline(h=4,col='red')
```

OUTPUT:

```
> rm(list=ls())
> data<-read.csv("USArrests.csv",row.names=1)
> df
function (x, df1, df2, ncp, log = FALSE)
{
    if (missing(ncp))
        .Call(C_df, x, df1, df2, log)
    else .Call(C_dnf, x, df1, df2, ncp, log)
}
<bytecode: 0x55f96c5e6100>
<environment: namespace:stats>
> df<-scale(data)
> dissim<-dist(df,method='euclidean')
> hierClust<-hclust(dissim,method='complete')
> plot(hierClust)
>
> cluster<-cutree(hierClust,k=4)
>
> library(clValid)
>
> dunn(dissim,cluster)
[1] 0.1621625
>
> rect.hclust(hierClust,k=4,border=2:4)
>
> abline(h=4,col='red')
>
```


Cluster Dendrogram
Height
0 1 2 3 4 5 6
South Dakota
West Virginia
North Dakota
Vermont
Maine
Iowa
New Hampshire
Idaho
Montana
Nebraska
Kentucky
Arkansas
Virginia
Wyoming
Missouri
Oregon
Washington
Delaware
Rhode Island
Massachusetts
New Jersey
Connecticut
Minnesota
Wisconsin
Oklahoma
Indiana
Kansas
Ohio
Pennsylvania
Hawaii
Utah
Colorado
California
Nevada
Florida
Texas
Illinois
New York
Arizona
Michigan
Maryland
New Mexico
Alaska
Alabama
Louisiana
Georgia
Tennessee
North Carolina
Mississippi
South Carolina
dissim
hclust (*, "complete")


Cluster Dendrogram
Height
0 1 2 3 4 5 6
South Dakota
West Virginia
North Dakota
Vermont
Maine
Iowa
New Hampshire
Idaho
Montana
Nebraska
Kentucky
Arkansas
Virginia
Wyoming
Missouri
Oregon
Washington
Delaware
Rhode Island
Massachusetts
New Jersey
Connecticut
Minnesota
Wisconsin
Oklahoma
Indiana
Kansas
Ohio
Pennsylvania
Hawaii
Utah
Colorado
California
Nevada
Florida
Texas
Illinois
New York
Arizona
Michigan
Maryland
New Mexico
Alaska
Alabama
Louisiana
Georgia
Tennessee
North Carolina
Mississippi
South Carolina
dissim
hclust (*, "complete")

Date: 01/Feb/2024	Gradient Descent Optimization
EXPERIMENT – 03	

AIM: To perform Gradient Descent Optimization

SOFTWARE REQUIRED: RStudio

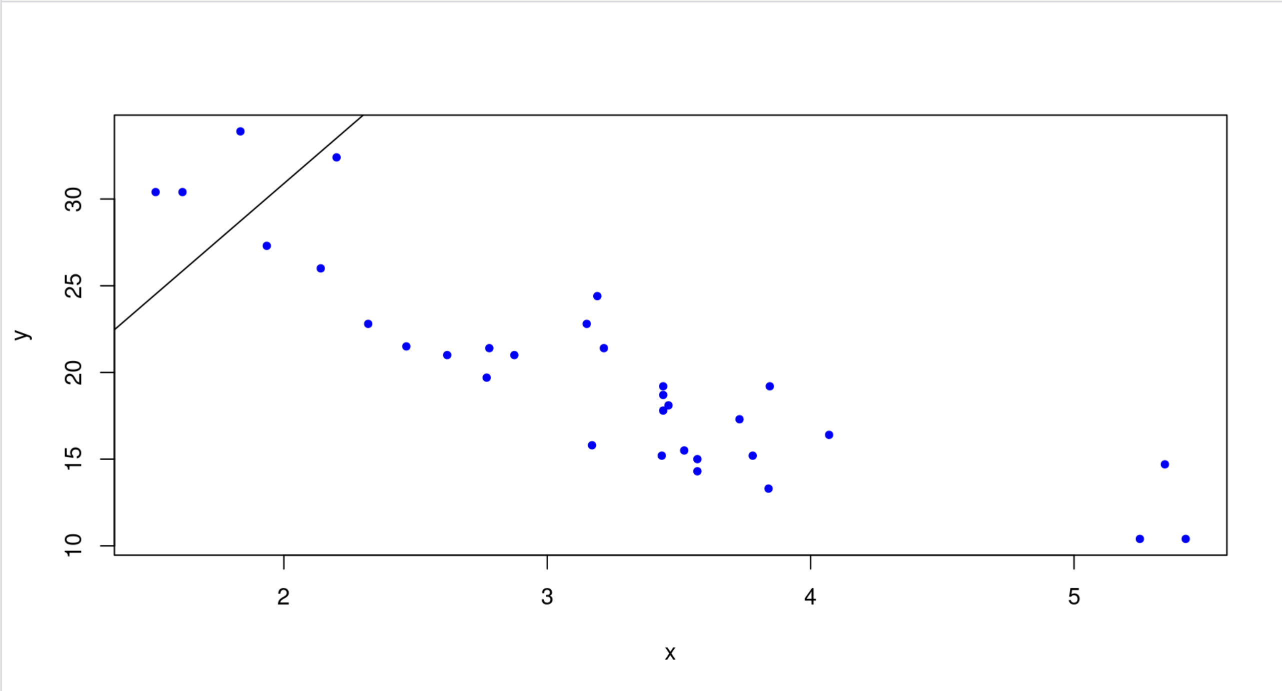
R CODE:

```
rm(list = ls ())
data <- mtcars
GRADIENT.DESCENT <- function (y, x, alpha, conv_threshold, n,
max_iter) {
  plot (x, y, col = "blue", pch = 20)
  m <- runif(1, 0, 1)
  c <- runif(1, 0, 1)
  yhat <- m * x + c
  MSE <- sum((y - yhat) ^ 2) / n
  converged = F
  iterations = 0
  while(converged == F) {
    m_new <- m - alpha * ((1 / n) * (sum((yhat - y) * x)))
    c_new <- c - alpha * ((1 / n) * (sum (yhat - y)))
    m <- m_new
    c <- c_new
    yhat <- m * x + c
    MSE_new <- sum((y - yhat) ^ 2) / n
    if(MSE - MSE_new <= conv_threshold){
        abline(c, m)
        converged = T
         return(paste("Optimal intercept:", c, "Optimal slope:", m,
"No of iterations:", iterations, "MSE:", MSE_new))
    }
    iterations = iterations + 1
    if(iterations >= max_iter) {
      abline(c , m)
      converged = T
        return(paste("Optimal intercept:", c, "Optimal slope:", m,
"No of iterations:", iterations, "MSE:", MSE_new))
    }
  }
}

GRADIENT.DESCENT(data$mpg, data$wt, 0.25, 0.001, length(data$mpg),
2500)
slr <- lm(mpg ~ wt, data = mtcars)
slr$coef
mpg_p <- predict (slr)
sqerr <- (data$mpg - mpg_p)^2
```

```
MSE.SLR <- sum(sqerr)/length(data$mpg)
slope:", m,
length (dataSmpg),
```

OUTPUT:

```
> rm(list = ls ())
> data <- mtcars
> GRADIENT.DESCENT <- function (y, x, alpha, conv_threshold, n, max_iter) {
+   plot (x, y, col = "blue", pch = 20)
+   m <- runif(1, 0, 1)
+   c <- runif(1, 0, 1)
+   yhat <- m * x + c
+   MSE <- sum((y - yhat) ^ 2) / n
+   converged = F
+   iterations = 0
+   while(converged == F) {
+     m_new <- m - alpha * ((1 / n) * (sum((yhat - y) * x)))
+     c_new <- c - alpha * ((1 / n) * (sum (yhat - y)))
+     m <- m_new
+     c <- c_new
+     yhat <- m * x + c
+     MSE_new <- sum((y - yhat) ^ 2) / n
+     if(MSE - MSE_new <= conv_threshold){
+         abline(c, m)
+         converged = T
+         return(paste("Optimal intercept:", c, "Optimal slope:", m, "No of iterations:", iterations, "MSE:", MSE_new))
+     }
+     iterations = iterations + 1
+     if(iterations >= max_iter) {
+       abline(c , m)
+       converged = T
+       return(paste("Optimal intercept:", c, "Optimal slope:", m, "No of iterations:", iterations, "MSE:", MSE_new))
+     }
+   }
+ }
>
> GRADIENT.DESCENT(data$mpg, data$wt, 0.25, 0.001, length(data$mpg), 2500)
[1] "Optimal intercept: 4.69138829676791 Optimal slope: 13.1012890281482 No of iterations: 0 MSE: 1039.87236235623"
> slr <- lm(mpg ~ wt, data = mtcars)
> slr$coef
(Intercept)          wt
  37.285126   -5.344472
> mpg_p <- predict (slr)
> sqerr <- (data$mpg - mpg_p)^2
> MSE.SLR <- sum(sqerr)/length(data$mpg)
> slope:", m,
+ length (dataSmpg),
```


y
10
15
20
25
30
x
2
3
4
5

Date: 08/Feb/2024	SIMPLE LINEAR REGRESSION
EXPERIMENT – 04	

AIM: To perform Simple Linear Regression and get the output with graphs

SOFTWARE REQUIRED: RStudio

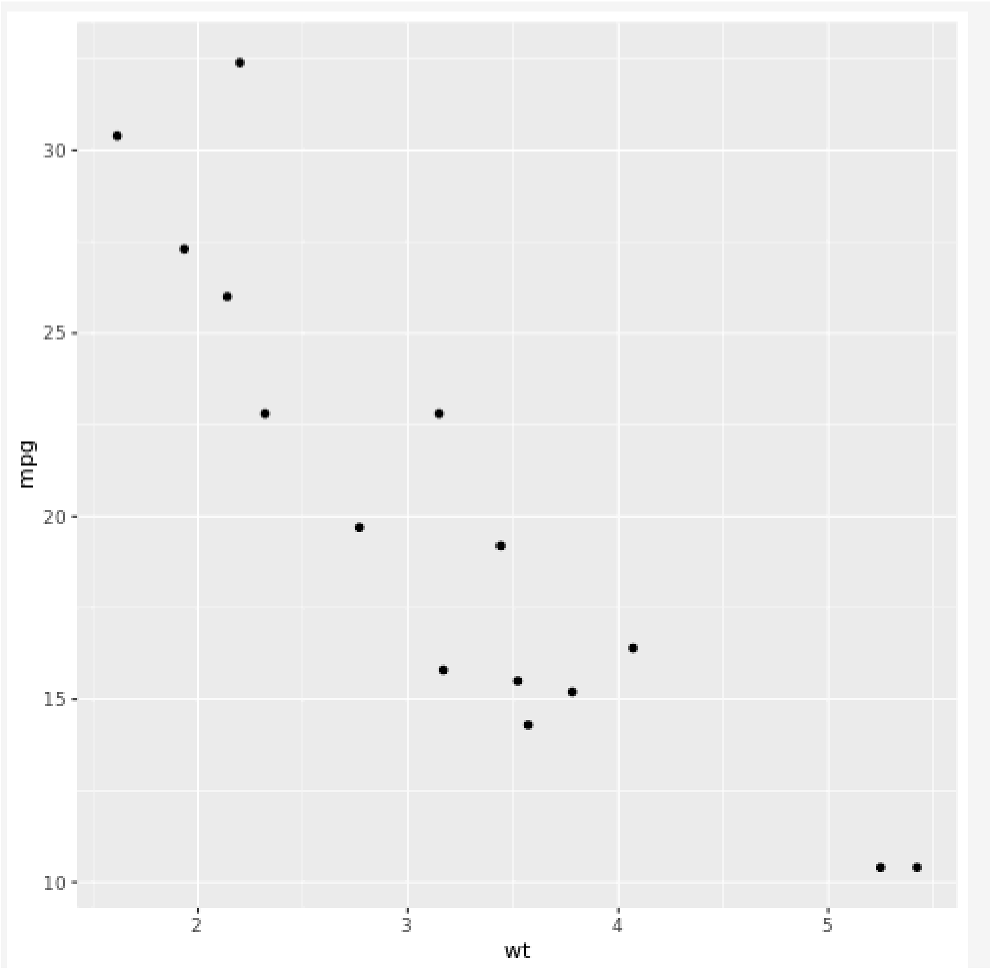
R CODE:

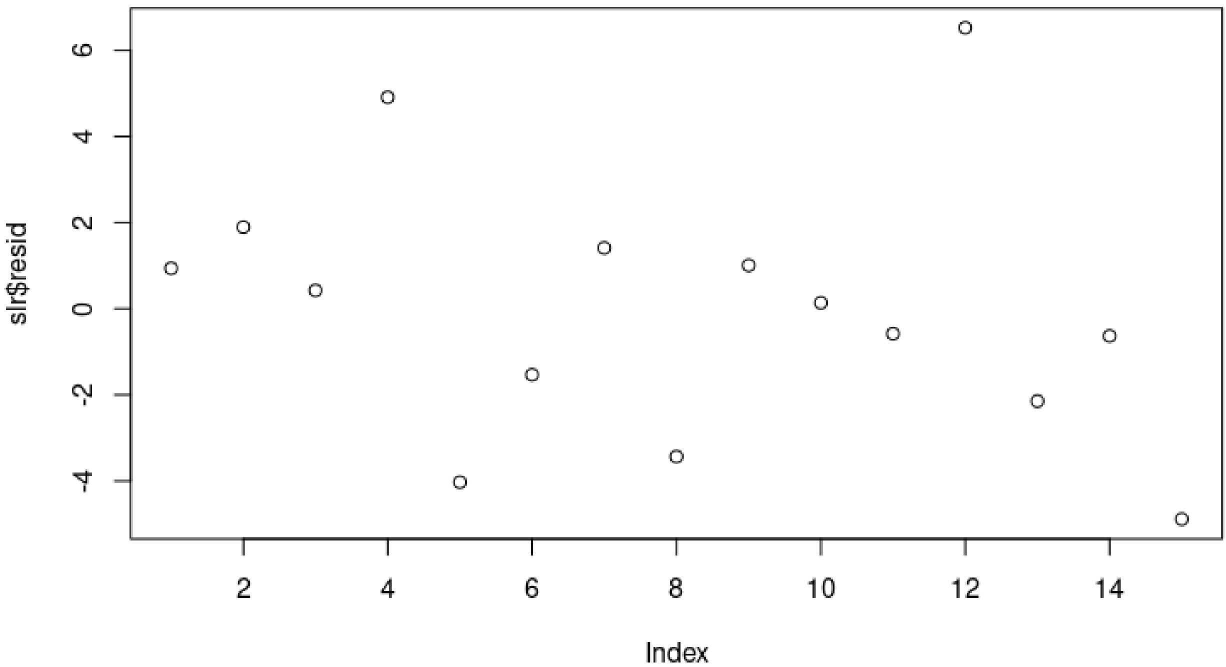
```
rm(list=ls())
data <- mtcars
library(dplyr)
data <- sample_n(data,15)
library("ggplot2")
ggplot(data, aes(x=wt,y=mpg))+geom_point()
cor.test(data$wt,data$mpg)
slr = lm(mpg~wt, data)
summary (slr)
plot(slr$resid)
qqnorm(slr$resid)
mlr = lm(mpg~wt+gear,data)
summary(mlr)
plot(mlr$resid)
qqnorm(mlr$resid)
```

OUTPUT:

```
> data <- sample_n(data,15)
>
> # install packages ("ggplot2")
> library("ggplot2")
> ggplot(data, aes(x=wt,y=mpg))+geom_point()
> cor.test(data$wt,data$mpg)

        Pearson's product-moment correlation

data:  data$wt and data$mpg
t = -6.2959, df = 13, p-value = 2.762e-05
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.9553546 -0.6400169
sample estimates:
      cor
-0.867774

> slr = lm(mpg~wt, data)
> summary (slr)

Call:
lm(formula = mpg ~ wt, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7251 -3.3019  0.2764  1.6628  6.3502

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.2055     2.9188  12.747 1.01e-08 ***
wt           -5.2620     0.8358  -6.296 2.76e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.515 on 13 degrees of freedom
Multiple R-squared:  0.753,     Adjusted R-squared:  0.734
F-statistic: 39.64 on 1 and 13 DF,  p-value: 2.762e-05

> plot(slr$resid)
> qqnorm(slr$resid)
> mlr = lm(mpg~wt+gear,data)
> summary(mlr)
```

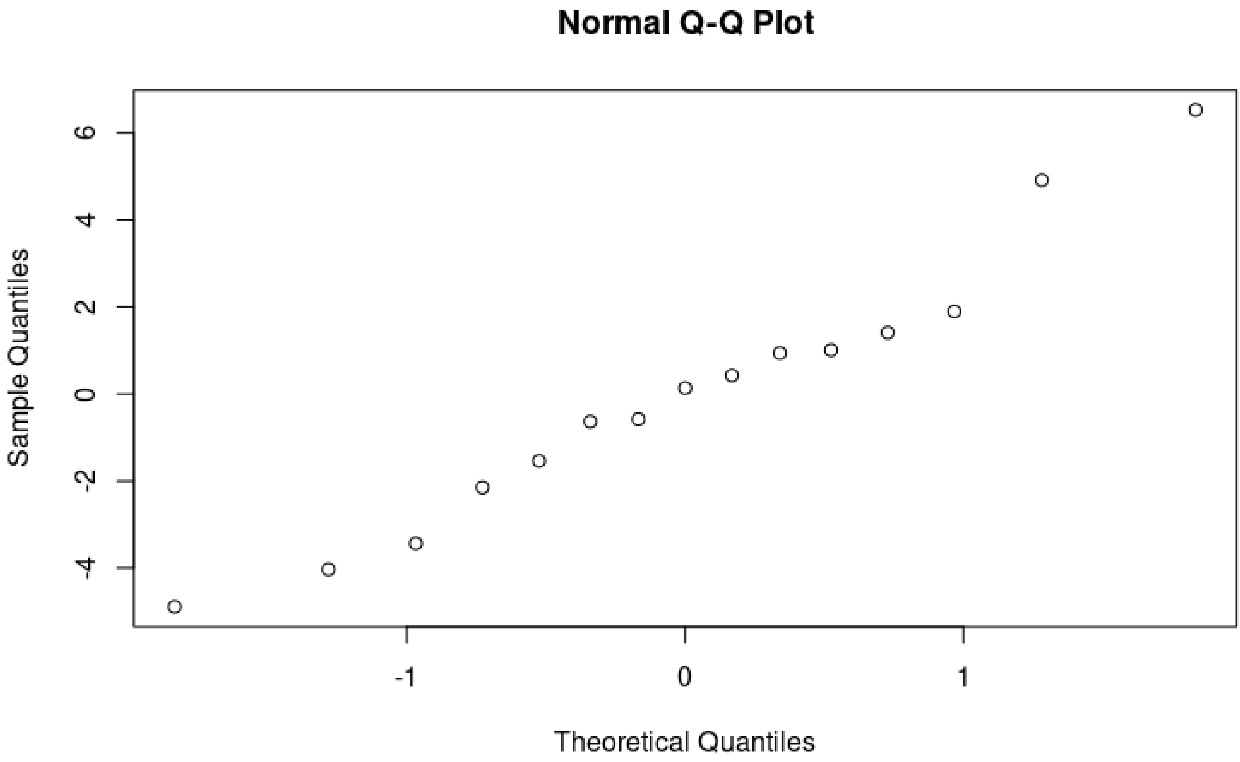
```
Call:
lm(formula = mpg ~ wt + gear, data = data)

Residuals:
   Min     1Q Median     3Q    Max 
-4.669 -3.050  0.306  1.599  5.921 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  41.6091     8.4669   4.914 0.000357 ***
wt           -5.6601     1.1182  -5.062 0.000279 ***
gear         -0.8111     1.4583  -0.556 0.588312    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.612 on 12 degrees of freedom
Multiple R-squared:  0.7592,	Adjusted R-squared:  0.7191 
F-statistic: 18.92 on 2 and 12 DF,  p-value: 0.0001948

> plot(mlr$resid)
> qqnorm(mlr$resid)
> 
```

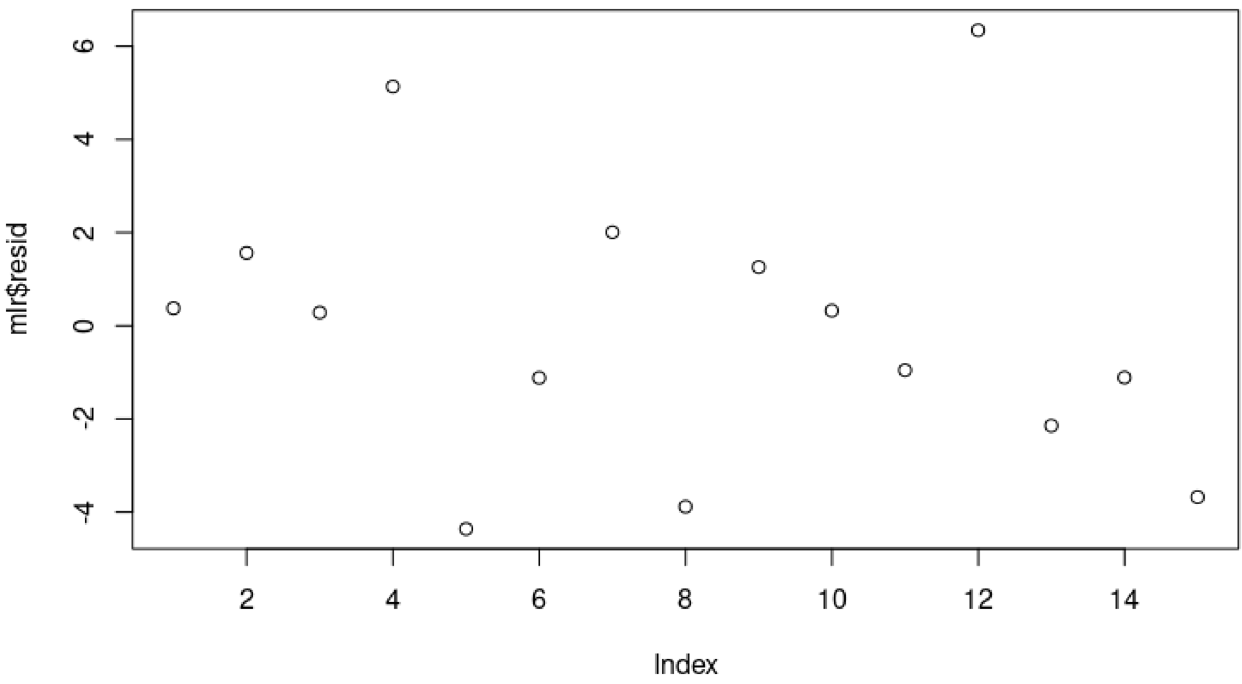




BCSE352E–Essentials of Data Analytics – Lab [Winter Semester 2023–24]

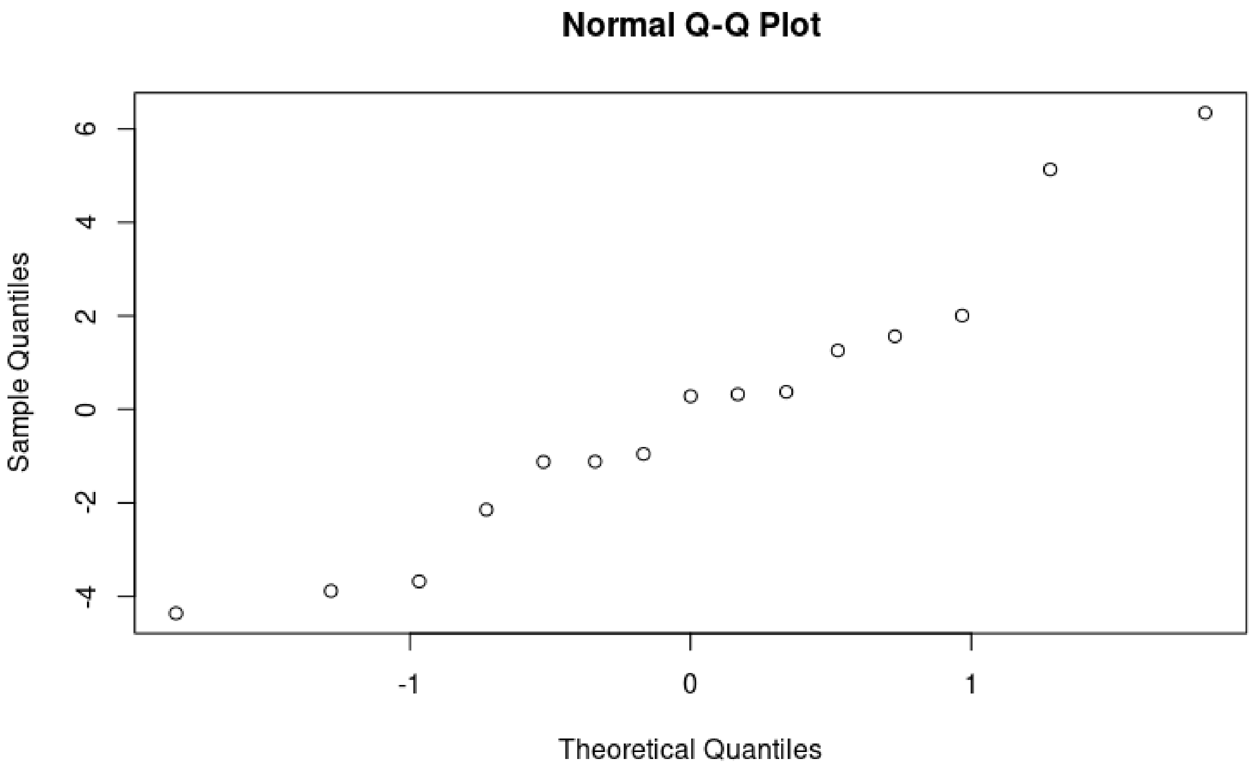
Name of the Student: **Sujay Ghosh**
Register Number: **21BLC1607**





BCSE352E–Essentials of Data Analytics – Lab [Winter Semester 2023–24]

Name of the Student: **Sujay Ghosh**
Register Number: **21BLC1607**



mlr	list [12] (S3: lm)	List of length 12
coefficients	double [3]	41.004 -5.380 -0.894
residuals	double [15]	0.373 1.563 0.281 5.132 -4.364 -1.123 ...
effects	double [15]	-74.59 23.06 -1.82 4.62 -4.64 -1.03 ...
rank	integer [1]	3
fitted.values	double [15]	21.03 17.64 27.02 9.57 17.66 18.92 ...
assign	integer [3]	0 1 2
qr	list [5] (S3: qr)	List of length 5
df.residual	integer [1]	12
xlevels	list [0]	List of length 0
call	language	lm(formula = mpg ~ wt + gear, data = data)
terms	formula	mpg ~ wt + gear
model	list [15 x 3] (S3: data.frame)	A data.frame with 15 rows and 3 columns

slr	list [12] (S3: lm)	List of length 12
coefficients	double [2]	36.57 -5.01
residuals	double [15]	0.938 1.895 0.424 4.912 -4.030 -1.534 ...
effects	double [15]	-74.594 23.064 0.728 4.064 -4.369 -1.738 ...
rank	integer [1]	2
fitted.values	double [15]	20.46 17.30 26.88 9.79 17.33 19.33 ...
assign	integer [2]	0 1
qr	list [5] (S3: qr)	List of length 5
df.residual	integer [1]	13
xlevels	list [0]	List of length 0
call	language	lm(formula = mpg ~ wt, data = data)
terms	formula	mpg ~ wt
model	list [15 x 2] (S3: data.frame)	A data.frame with 15 rows and 2 columns

Date: 22/Feb/2024	SUPPORT VECTOR MACHINE
EXPERIMENT – 05	

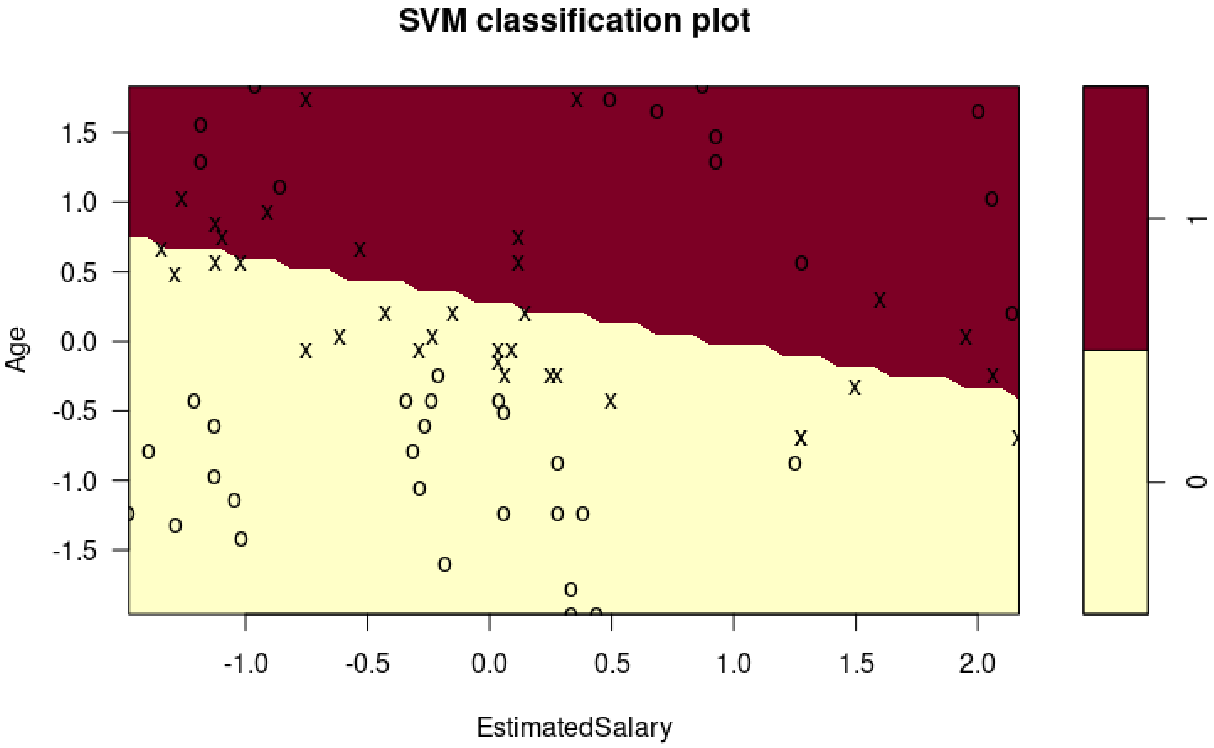
AIM: To perform support vector machine

SOFTWARE REQUIRED: RStudio

R CODE:

```
rm(list=ls())
data =read.csv('social.csv' )
library (dplyr)
data=sample_n(data, 80)
data=data [3:5]
dataşPurchased= factor(data$Purchased, levels=c(0,1))
data_TRAIN<-sample_n(data,0.9*length(data$Purchased))
data_TEST<-setdiff(data, data_TRAIN)
data_TRAIN[-3] <- scale (data_TRAIN[-3])
data_TEST[-3] <- scale(data_TEST[-3])
library(e1071)
SVMclassifier =svm(formula=Purchased ~ ., data= data_TRAIN,type=
'C-classification', kernel ='linear')
plot(SVMclassifier, data_TRAIN)
y_p =predict(SVMclassifier, newdata =data_TEST[-3])
install.packages('caret')
library (caret)
library (ggplot2)
install.packages('lattice')
library (lattice)
confusionMatrix(table (y_p, data_TEST$Purchased))
```

OUTPUT:



Data	
data	80 obs. of 3 variables
data_TEST	8 obs. of 3 variables
data_TRAIN	72 obs. of 3 variables
SVMclassifier	List of 31
Values	
data$Purchased	Factor w/ 2 levels "0","1": 1 2 1 2 1 1 2 2 1 2 ...
y_p	Factor w/ 2 levels "0","1": 1 1 2 1 2 2 1 1

```
Confusion Matrix and Statistics


y_p 0 1
  0 4 1
  1 0 3

               Accuracy : 0.875
                 95% CI : (0.4735, 0.9968)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.03516

                  Kappa : 0.75

 Mcnemar's Test P-Value : 1.00000

            Sensitivity : 1.000
            Specificity : 0.750
         Pos Pred Value : 0.800
         Neg Pred Value : 1.000
             Prevalence : 0.500
         Detection Rate : 0.500
   Detection Prevalence : 0.625
      Balanced Accuracy : 0.875

       'Positive' Class : 0
```

	Age	EstimatedSalary	Purchased
1	33	28000	0
2	60	34000	1
3	42	64000	0
4	37	79000	1
5	39	59000	0
6	26	72000	0
7	45	22000	1
8	32	150000	1
9	34	43000	0
10	51	23000	1
11	31	18000	0
12	37	62000	0
13	33	60000	0
14	18	82000	0
15	40	47000	0
16	35	61000	0
17	43	129000	1

```
User ID,Gender,Age,EstimatedSalary,Purchased
15624510,Male,19,19000,0
15810944,Male,35,20000,0
15668575,Female,26,43000,0
15603246,Female,27,57000,0
15804002,Male,19,76000,0
15728773,Male,27,58000,0
15598044,Female,27,84000,0
15694829,Female,32,150000,1
15600575,Male,25,33000,0
15727311,Female,35,65000,0
15570769,Female,26,80000,0
15606274,Female,26,52000,0
15746139,Male,20,86000,0
15704987,Male,32,18000,0
15628972,Male,18,82000,0
15697686,Male,29,80000,0
15733883,Male,47,25000,1
15617482,Male,45,26000,1
15704583,Male,46,28000,1
15621083,Female,48,29000,1
15649487,Male,45,22000,1
15736760,Female,47,49000,1
15714658,Male,48,41000,1
15599081,Female,45,22000,1
15705113,Male,46,23000,1
15631159,Male,47,20000,1
15792818,Male,49,28000,1
15633531,Female,47,30000,1
```

Date: 29/Feb/2024	ANALYSIS OF VARIANCE(ANNOVA)
EXPERIMENT – 06	

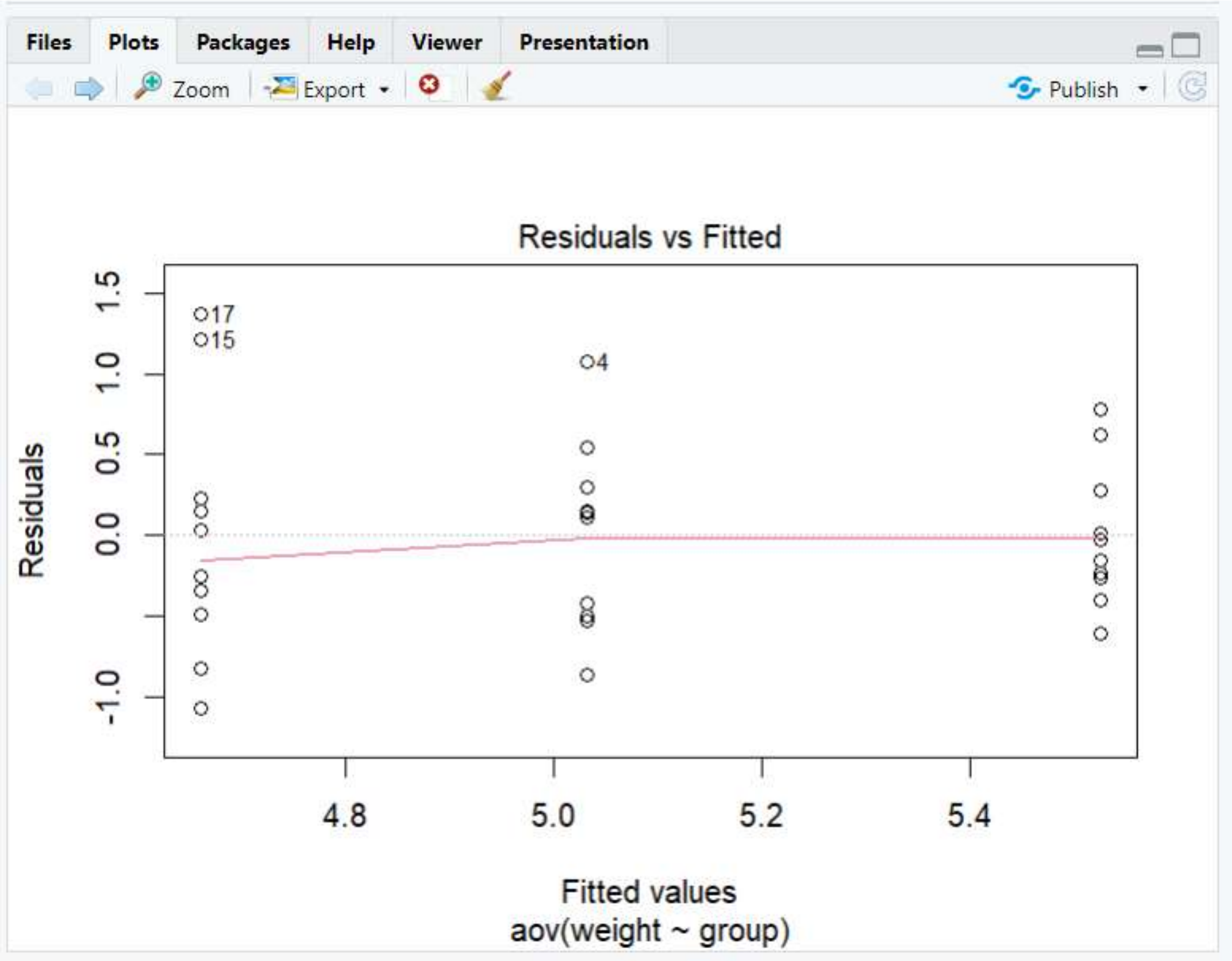
AIM: To perform analysis of variance(annova)

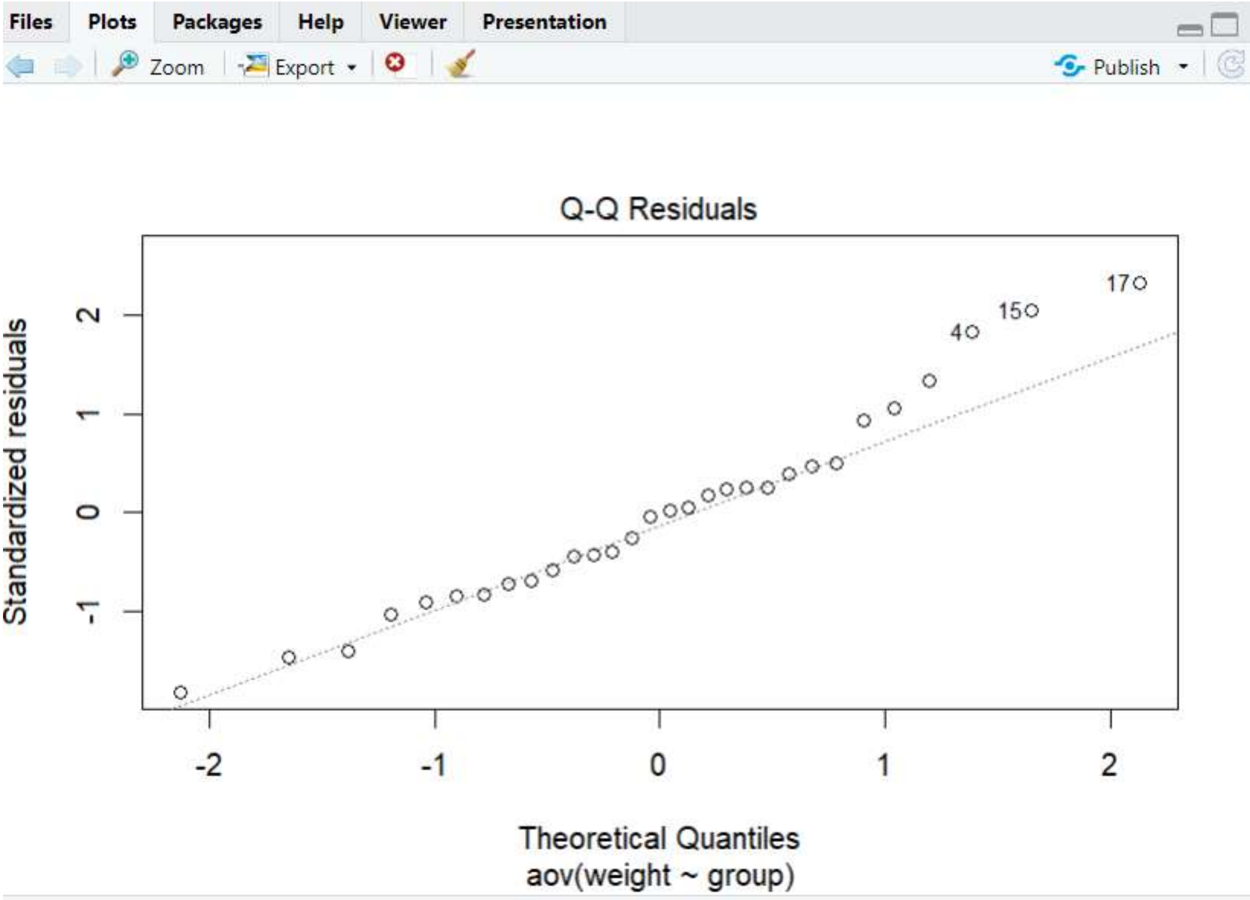
SOFTWARE REQUIRED: RStudio

R CODE:

```
rm(list=ls())

data<-data.frame(sale.count=c
(40,60,70,30,50,30,30,10,70,60,50,60,30,20,20),type=c("Can-
A","Can-A","Can-A","Can-A","Can-A","Can-B","Can-B","Can-B","Can-
B","Can-B","Can-C","Can-C","Can-C","Can-C","Can-C"))
library(dplyr)
group_by(data,type)                                            %>%
summarise(count=n(),mean=mean(sale.count,na.rm=TRUE))
result<-aov(sale.count~type,data=data)
summary(result)
data<-PlantGrowth
group_by(data,group)                                           %>%
summarise(count=n(),mean=mean(weight,na.rm=TRUE))
result<-aov(weight~group,data=data)
summary(result)
TukeyHSD(result)
plot(result,1)
plot(result,2)
kruskal.test(weight~group,data=data)
data<- read.csv("iris.csv",row.names=1)
```

OUTPUT:


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Residuals vs Fitted
Residuals
Fitted values
aov(weight ~ group)
17
15
4
1.5
1.0
0.5
0.0
-1.0
4.8
5.0
5.2
5.4




```
> fit<- kmeans(df,3)
>
> fit$size
[1] 47 53 50
> fit$withinss
[1] 47.45019 44.08754 47.35062
> fit$tot.withinss  # Within Cluster Sum of Squares (WCSS)
[1] 138.8884
```

```
> group_by(data,type) %>% summarise(count=n(),mean=mean(sale.c
ount,na.rm=TRUE))
# A tibble: 3 × 3
  type  count  mean
  <chr> <int> <dbl>
1 Can-A     5    50
2 Can-B     5    40
3 Can-C     5    36
> 
```

```
>
> group_by(data,group) %>% summarise(count=n(),mean=mean(weig
t,na.rm=TRUE))
# A tibble: 3 × 3
  group count  mean
  <fct> <int> <dbl>
1 ctrl     10  5.03
2 trt1     10  4.66
3 trt2     10  5.53
> 
```

```
> result<-aov(weight~group,data=data)
> summary(result)
            Df Sum Sq Mean Sq F value Pr(>F)  
group        2  3.766  1.8832   4.846 0.0159 *
Residuals   27 10.492  0.3886                 
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> 
```

```
> TukeyHSD(result)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = weight ~ group, data = data)

$group
            diff        lwr       upr     p adj
trt1-ctrl -0.371 -1.0622161 0.3202161 0.3908711
trt2-ctrl  0.494 -0.1972161 1.1852161 0.1979960
trt2-trt1  0.865  0.1737839 1.5562161 0.0120064
```

```
> kruskal.test(weight~group,data=data)

        Kruskal-Wallis rank sum test

data:  weight by group
Kruskal-Wallis chi-squared = 7.9882, df = 2, p-value
= 0.01842

> 
```

```
> result<-aov(sale.count~type,data=data)
> summary(result)
            Df Sum Sq Mean Sq F value Pr(>F)
type         2    520   260.0   0.661  0.534
Residuals   12   4720   393.3
> 
```

	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
7	5.17	ctrl
8	4.53	ctrl
9	5.33	ctrl
10	5.14	ctrl
11	4.81	trt1
12	4.17	trt1
13	4.41	trt1
14	3.59	trt1
15	5.87	trt1

	sale.count	type
1	40	Can-A
2	60	Can-A
3	70	Can-A
4	30	Can-A
5	50	Can-A
6	30	Can-B
7	30	Can-B
8	10	Can-B
9	70	Can-B
10	60	Can-B
11	50	Can-C
12	60	Can-C
13	30	Can-C
14	20	Can-C
15	20	Can-C

Date: 14/Mar/2024	TIME-SERIES FORECASTING
EXPERIMENT – 07	

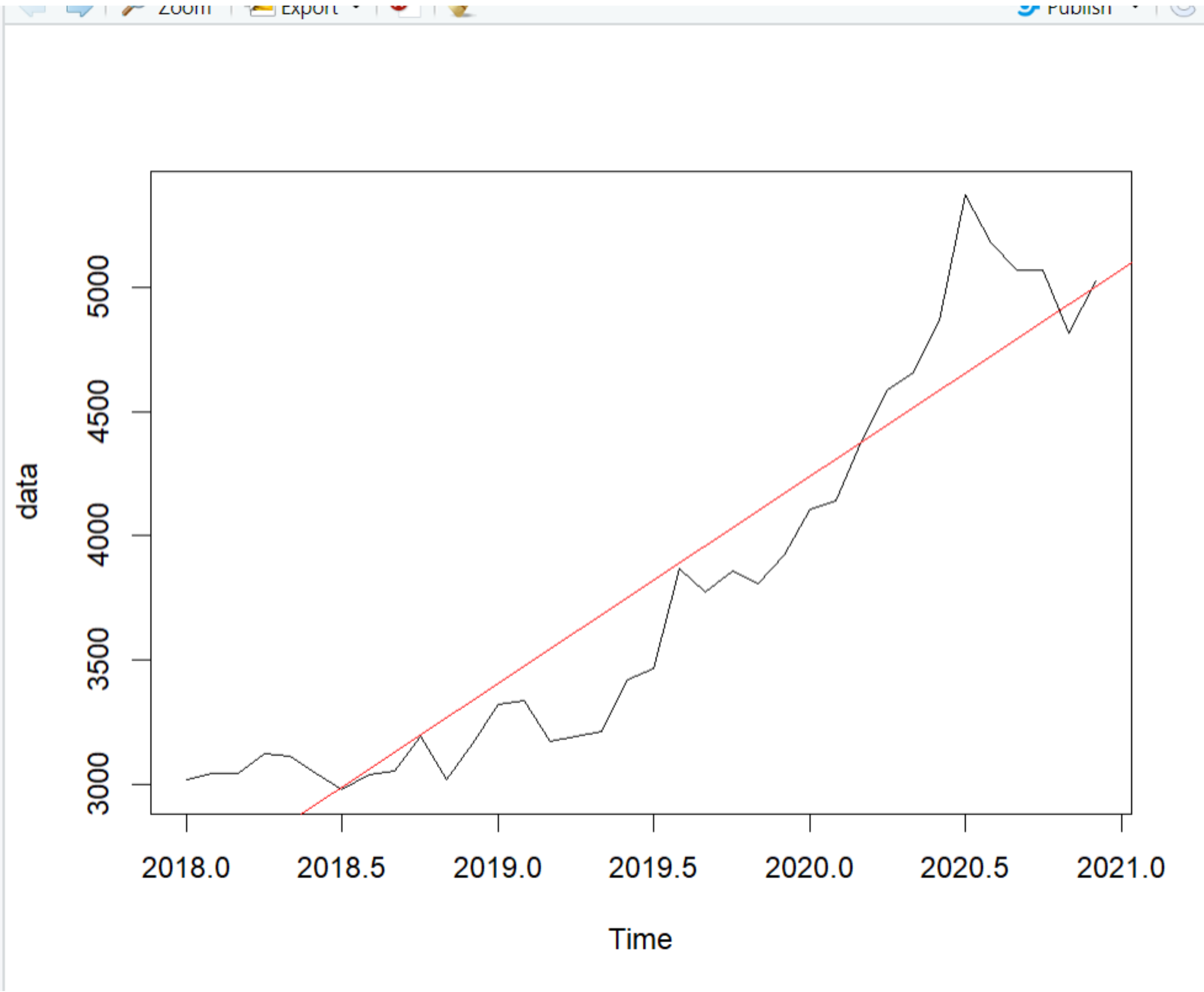
AIM: To perform time-series forecasting and obtain the plots.

SOFTWARE REQUIRED: RStudio

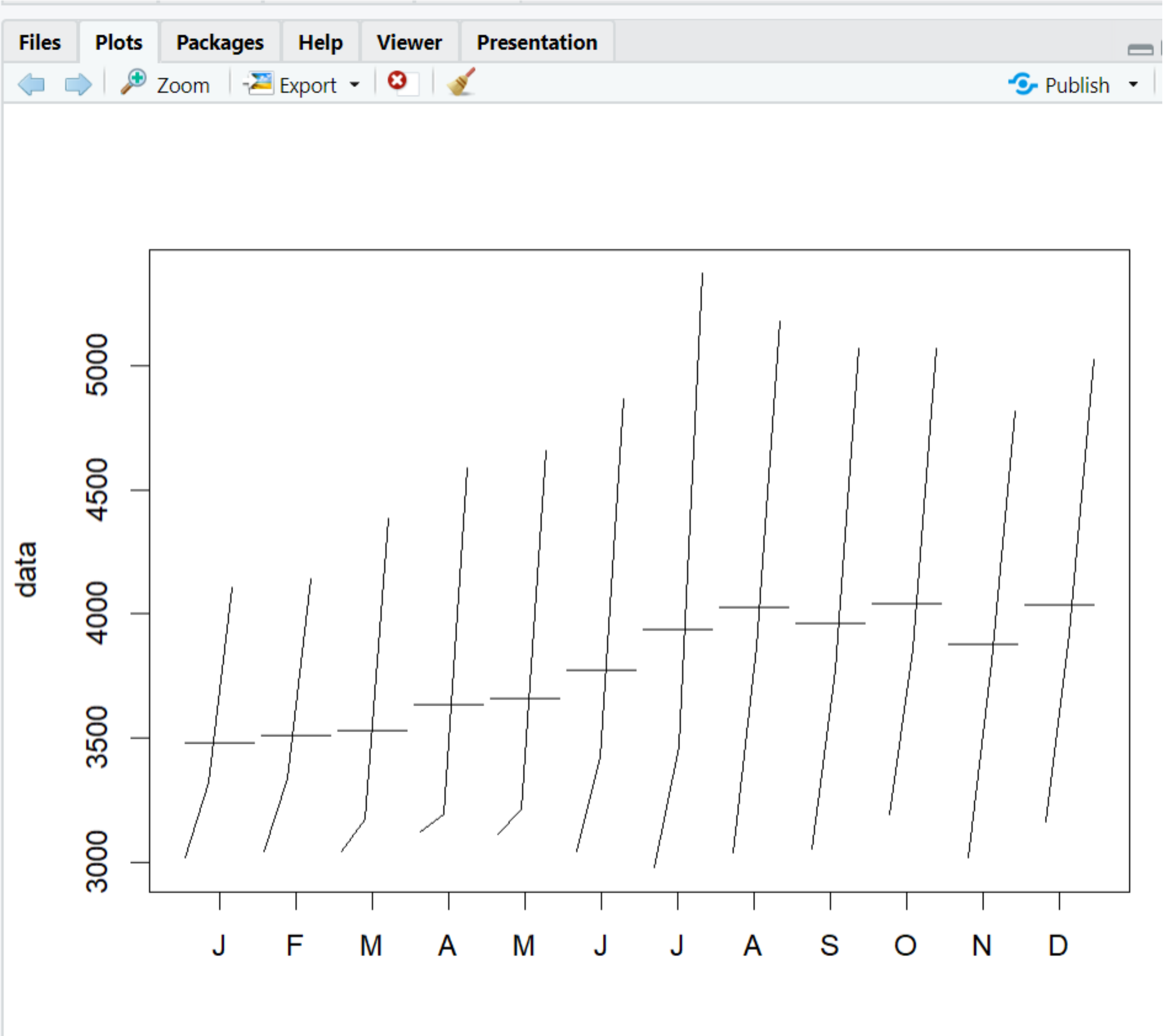
R CODE:

```
rm(list=ls())
vec=c(3016, 3044, 3041, 3121, 3111, 3043, 2977, 3036, 3051, 3191,
3016, 3164, 3321, 3338, 3170, 3194, 3212, 3420, 3465, 3866, 3774,
3858, 3807, 3922, 4105, 4141, 4383, 4587, 4656, 4864, 5373, 5179,
5068, 5071, 4814, 5024)
data<- ts(vec, start=c(2018,1), end=c(2020,12), frequency=12)
start(data)
end(data)
frequency (data)
cycle(data)
summary(data)
plot(data)
abline(reg=lm(data~time(data)), col="red")
monthplot(data)
plot(aggregate(data,FUN=mean))
boxplot(data~cycle(data))
library (forecast)
seasonplot(data)
acf(data)
pacf(data, lag=length(data), pl=TRUE)
fit<- arima(data, order=c(3,2,2))
accuracy(fit)
newdata<- forecast(fit, 4)
plot(newdata)
fit<- auto.arima(data)
newdata<- forecast(fit, 4)
plot(newdata)
```

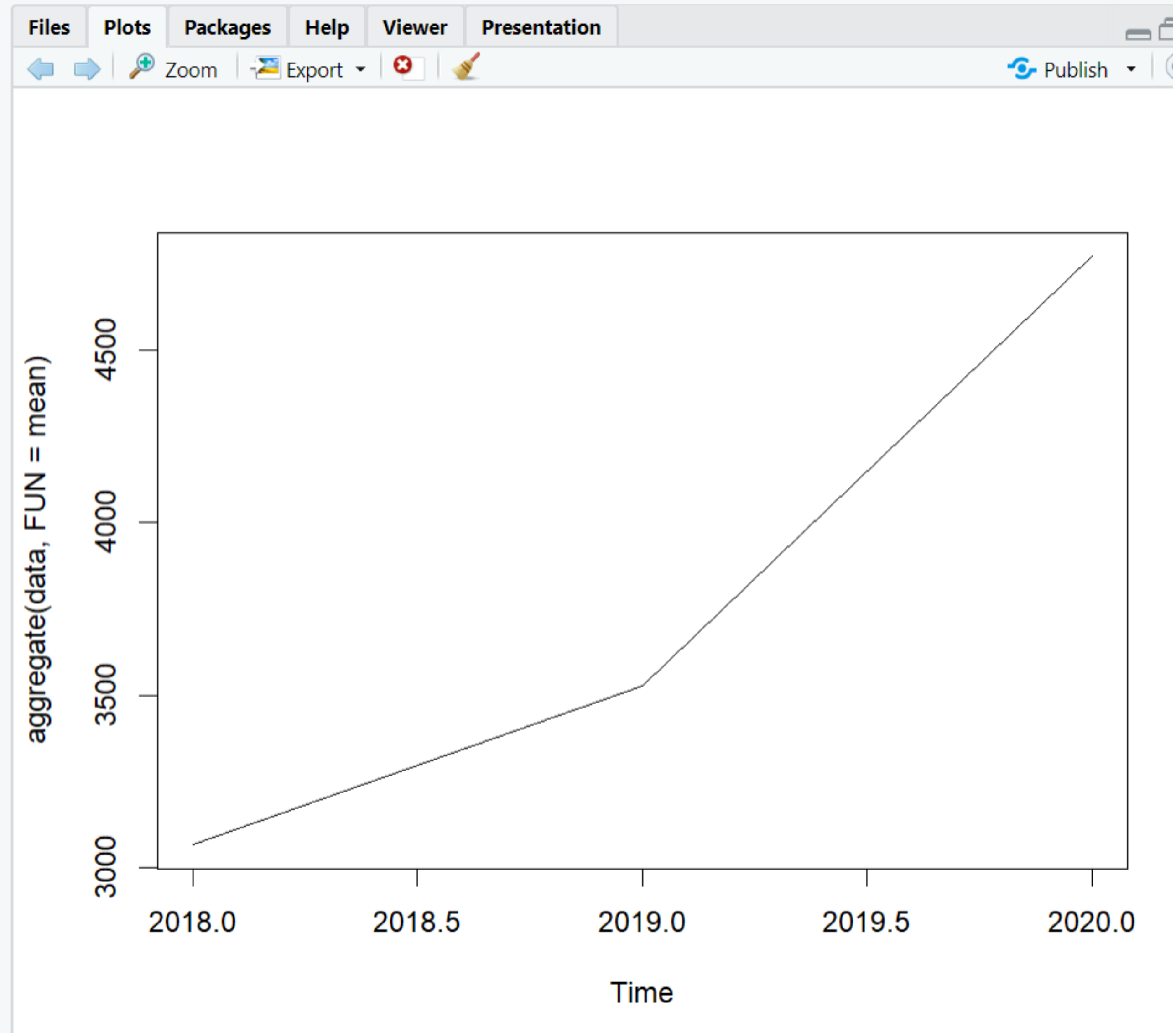
OUTPUT:



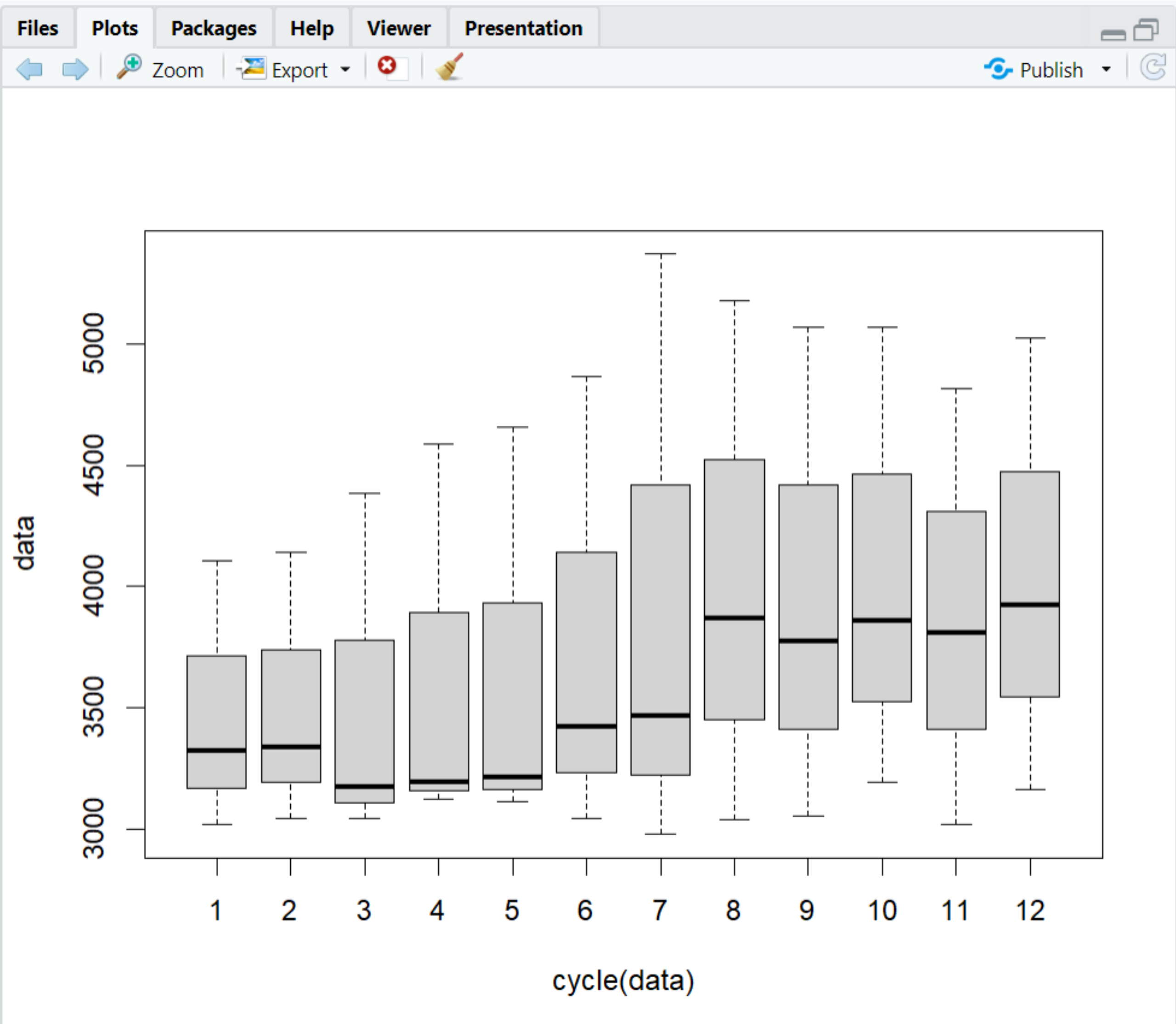
Zoom
Export
Publish
data
5000
4500
4000
3500
3000
2018.0
2018.5
2019.0
2019.5
2020.0
2020.5
2021.0
Time

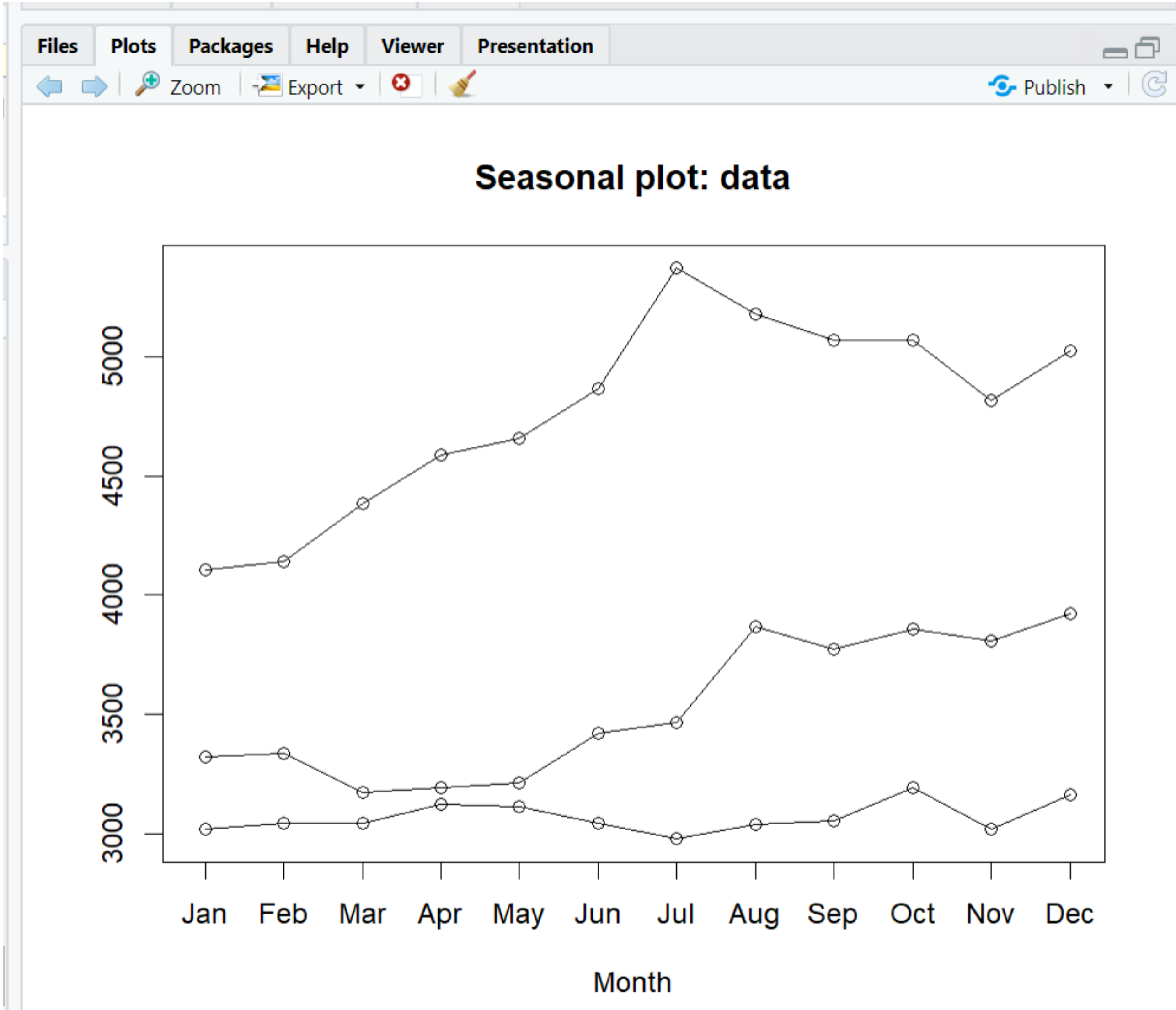


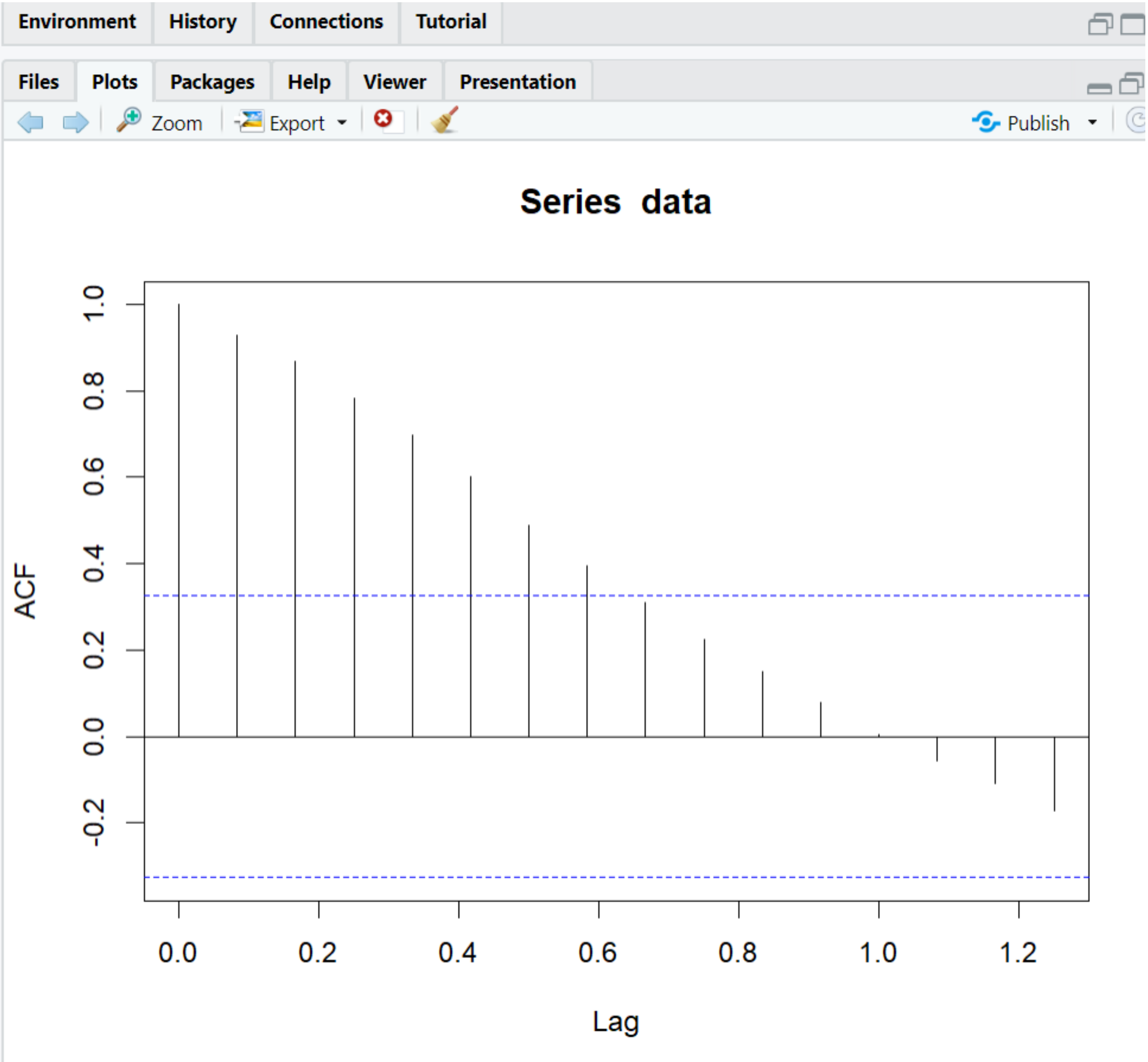
Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
data
3000
3500
4000
4500
5000
J
F
M
A
M
J
J
A
S
O
N
D

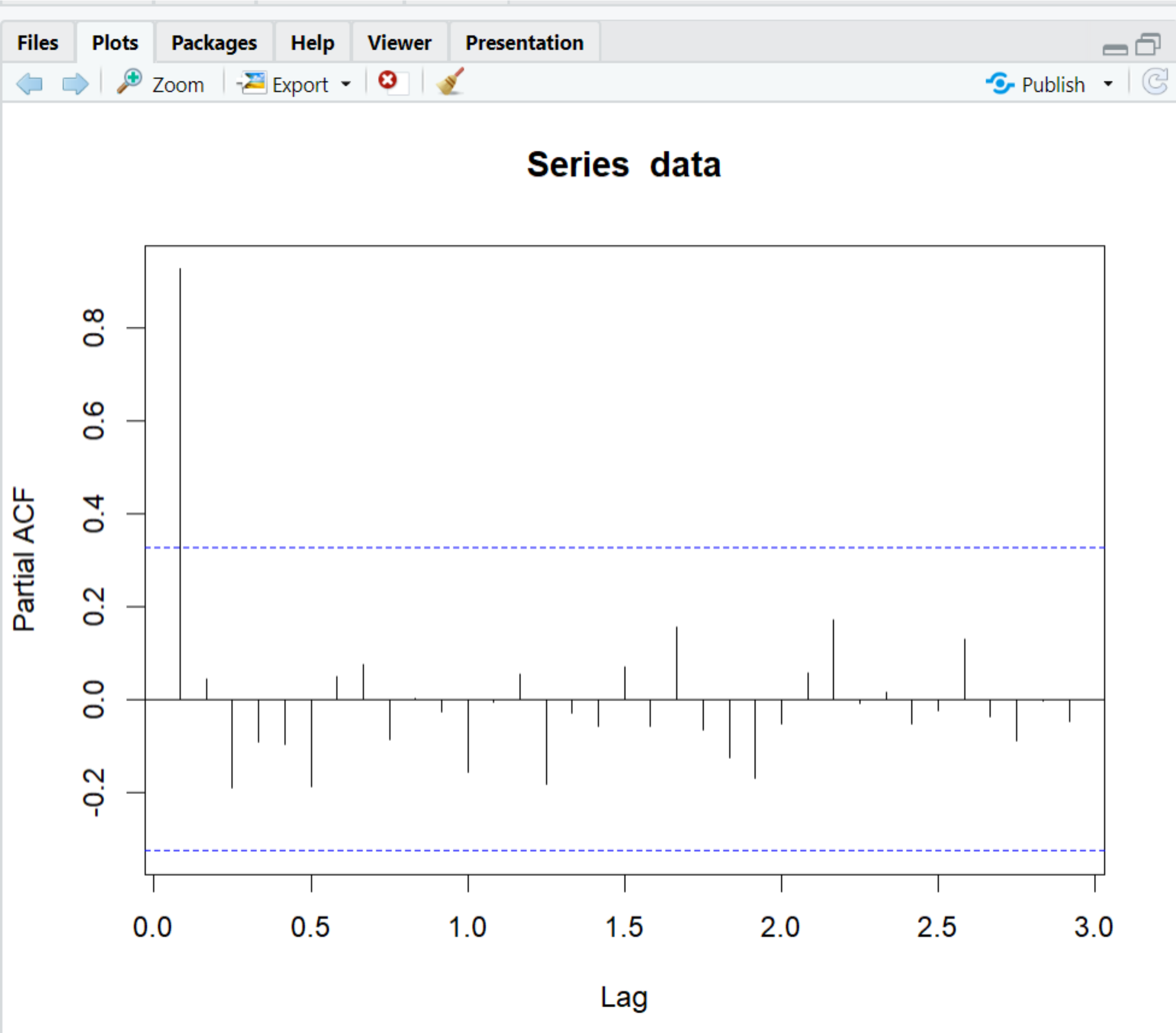


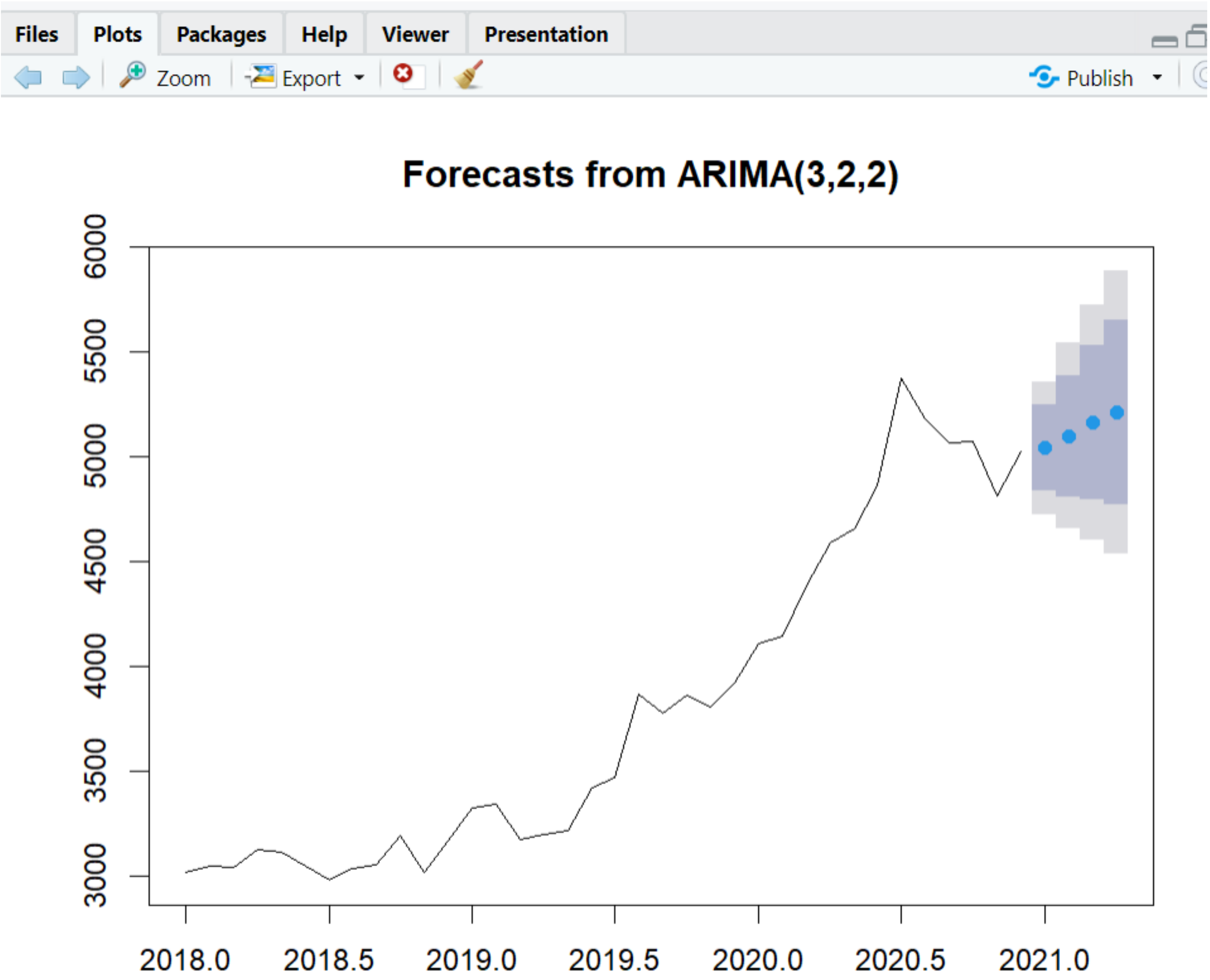
Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
aggregate(data, FUN = mean)
3000
3500
4000
4500
2018.0
2018.5
2019.0
2019.5
2020.0
Time

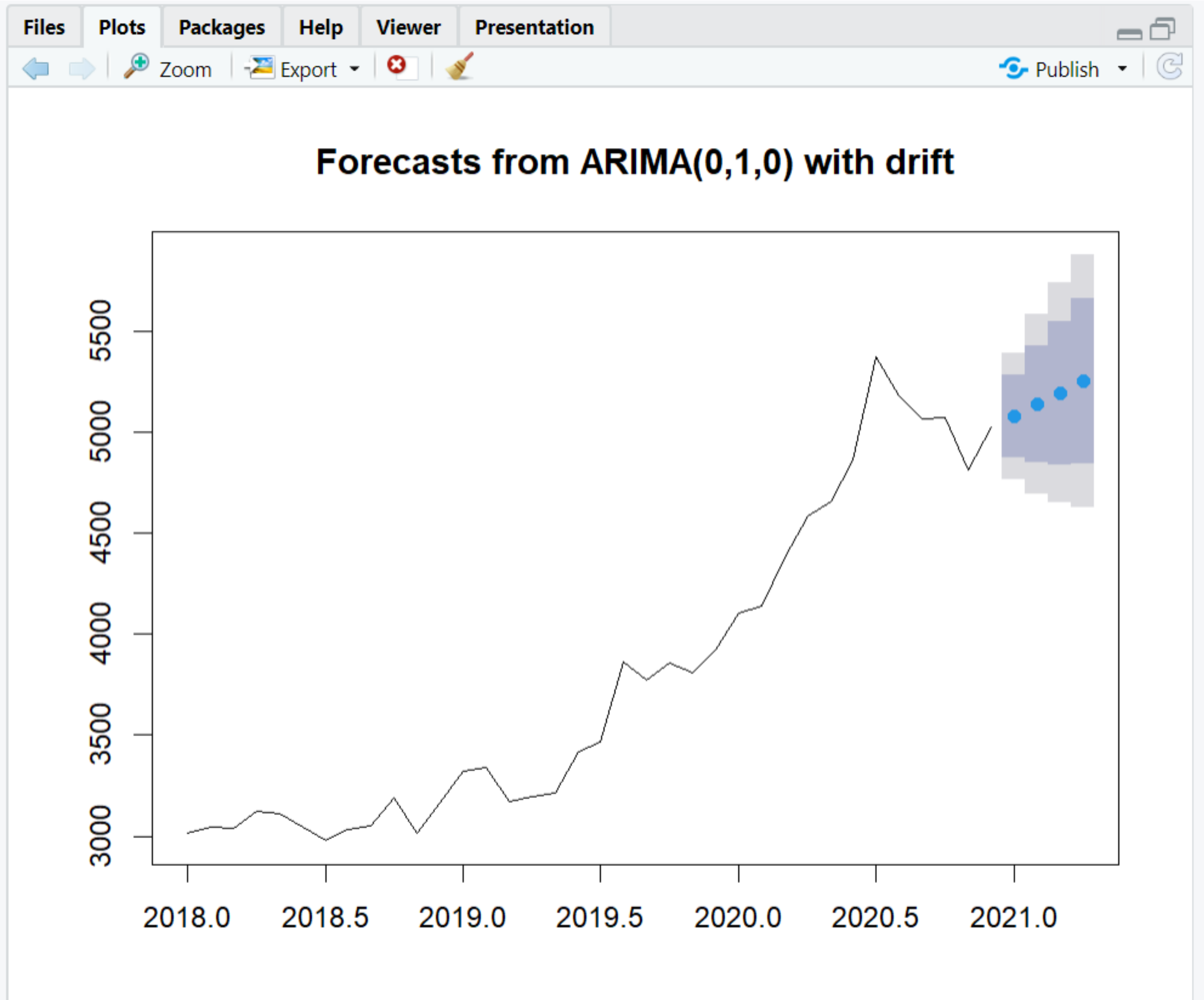

Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
data
3000
3500
4000
4500
5000
1
2
3
4
5
6
7
8
9
10
11
12
cycle(data)


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Seasonal plot: data
3000
3500
4000
4500
5000
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month


Environment
History
Connections
Tutorial
Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Series data
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
1.2
-0.2
0.0
0.2
0.4
0.6
0.8
1.0


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Series data
Partial ACF
-0.2
0.0
0.2
0.4
0.6
0.8
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Lag


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Forecasts from ARIMA(3,2,2)
3000
3500
4000
4500
5000
5500
6000
2018.0
2018.5
2019.0
2019.5
2020.0
2020.5
2021.0




```
> data<- ts(vec, start=c(2018,1), end=c(2020,12), frequency=12)
> #This line converts the vector vec into a time series object (ts) with a monthly frequency.
 It specifies the start and end dates of the time series.
> start(data)
[1] 2018    1
> end(data)
[1] 2020   12
> frequency (data)
[1] 12
> cycle(data)
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2018   1   2   3   4   5   6   7   8   9  10  11  12
2019   1   2   3   4   5   6   7   8   9  10  11  12
2020   1   2   3   4   5   6   7   8   9  10  11  12
> #These lines display information about the time series object data, including its start dat
e, end date, frequency (number of observations per unit of time), and the cycle length.
> summary(data)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   2977    3118    3442    3790    4434    5373 
```

```
ified maximum lag and plot option.
 #ARIMA
 fit<- arima(data, order=c(3,2,2))
 accuracy(fit)
                  ME     RMSE      MAE       MPE     MAPE      MASE         ACF1
raining set 20.49163 154.8341 114.0773 0.4894397 2.866398 0.9078461 -0.04015319
 newdata<- forecast(fit, 4)
 plot(newdata)
 #These lines fit an ARIMA (AutoRegressive Integrated Moving Average) model to the time serie
 data, compute accuracy measures, generate forecasts for the next 4 periods, and plot the for
casted values.
 #Auto ARIMA
 fit<- auto.arima(data)
 newdata<- forecast(fit, 4)
 plot(newdata)
```

Date: 221/Mar/2024	REGRESSION AND FORECASTING ON WEATHER DATA
EXPERIMENT – 08	

AIM: To perform regression and forecasting on weather data

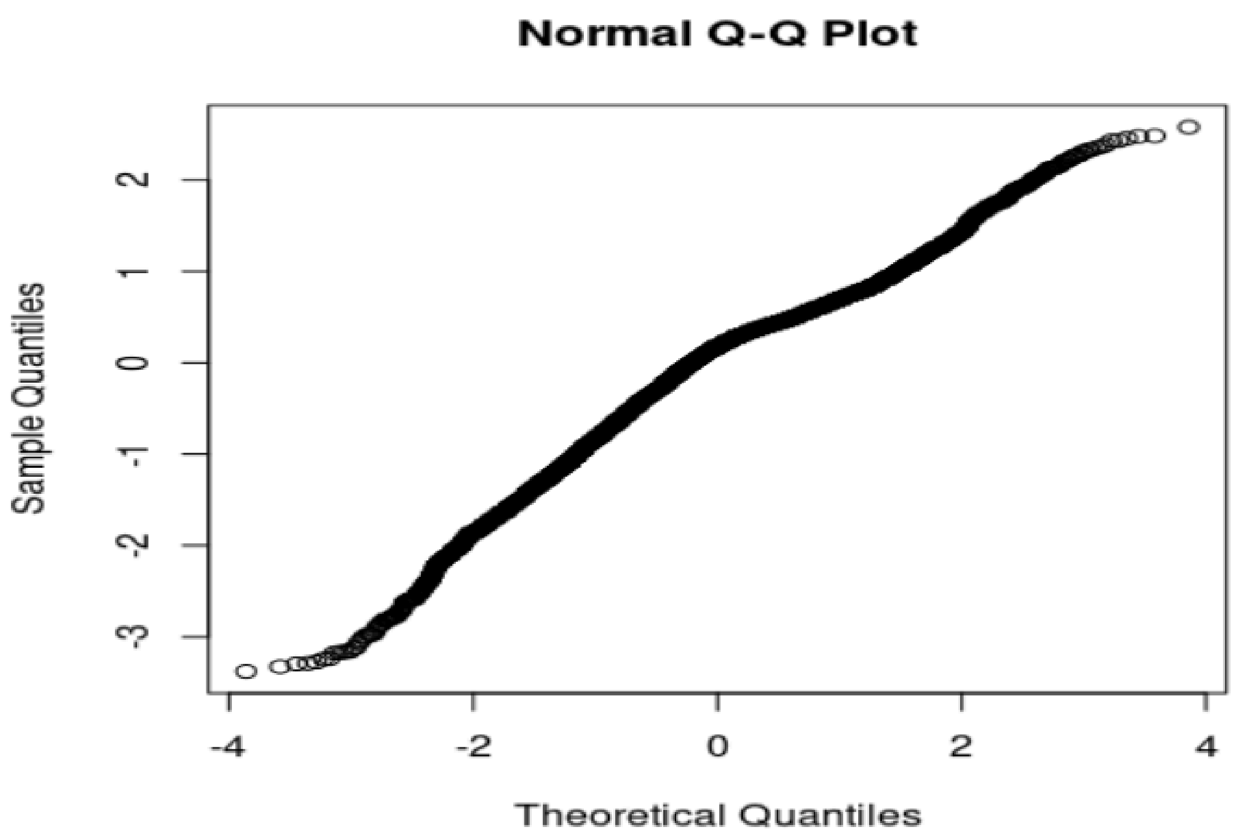
SOFTWARE REQUIRED: RStudio

R CODE:

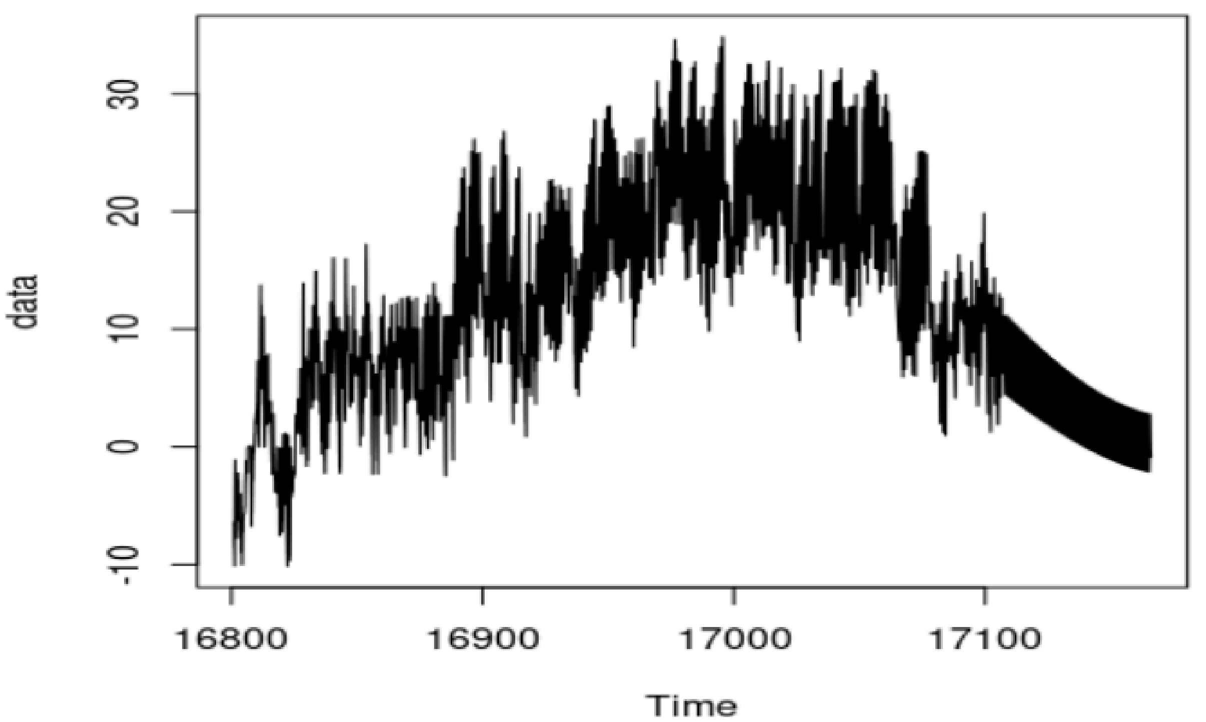
```
rm(list=ls())
a <- read.csv('weatherHistory2016.csv')
mlr=

lm(Temperature..C.~Apparent.Temperature..C.+Humidity+Wind.Speed..k
m.h.,a)
summary(mlr)
qqnorm(mlr$resid)
data    <-    ts(a$Temperature..C.,    start=as.Date("2016-01-01"),
end=as.Date("2016-12-31"), frequency=24)
frequency(data)
summary(data)
plot(data)
plot(aggregate(data,FUN=mean))
boxplot(data~cycle(data))
library(forecast)
acf(data)
fit<- auto.arima(data)
accuracy(fit)
newdata<- forecast(fit, 240)
plot(newdata)
```

OUTPUT:

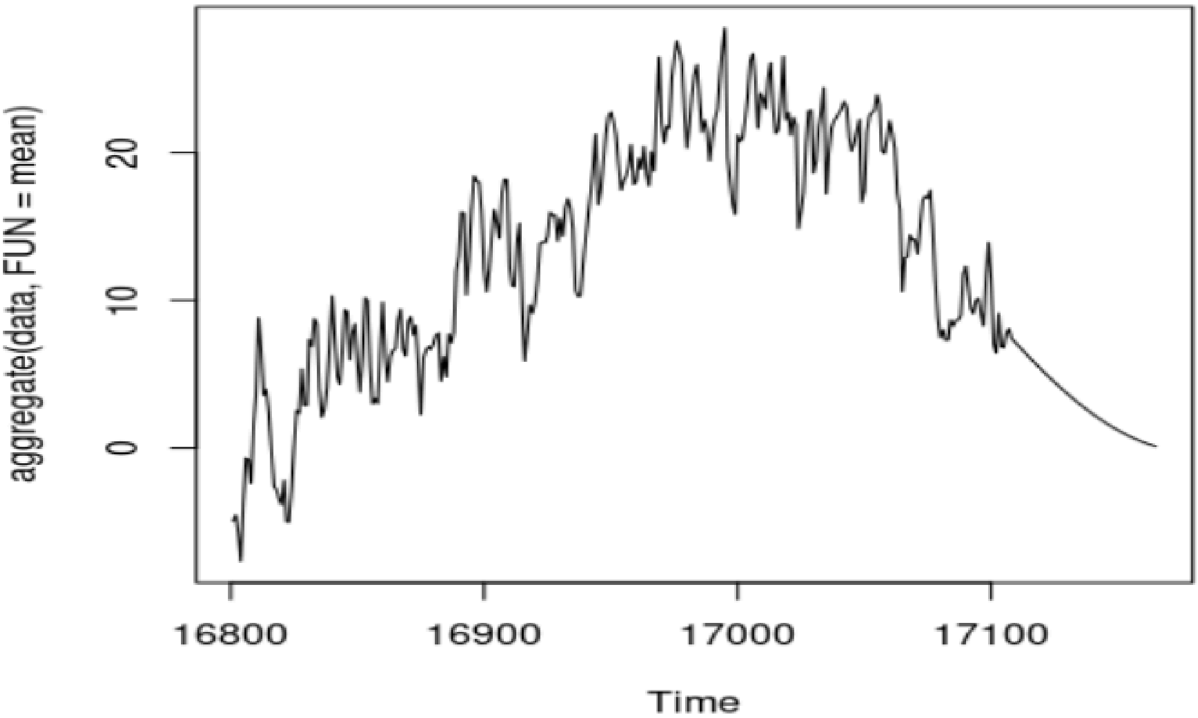
```
Console   Terminal ×   Background Jobs ×
R 4.3.3 . /cloud/project/
> rm(list=ls())
> a <- read.csv('weatherHistory2016.csv')
> mlr=
+   lm(Temperature..C.~Apparent.Temperature..C.+Humidity+Wind.Speed..km.h.,a)
> summary(mlr)

Call:
lm(formula = Temperature..C. ~ Apparent.Temperature..C. + Humidity + 
    Wind.Speed..km.h., data = a)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.3766 -0.4811  0.1755  0.5217  2.5771 

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)    
(Intercept)               3.216921   0.066288   48.53   <2e-16 ***
Apparent.Temperature..C.  0.863081   0.001148  751.63   <2e-16 ***
Humidity                 -1.521075   0.065469  -23.23   <2e-16 ***
Wind.Speed..km.h.         0.053840   0.001501   35.86   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8203 on 8780 degrees of freedom
Multiple R-squared:  0.9918,	Adjusted R-squared:  0.9917 
F-statistic: 3.519e+05 on 3 and 8780 DF,  p-value: < 2.2e-16

> qqnorm(mlr$resid)
> data <- ts(a$Temperature..C., start=as.Date("2016-01-01"), end=as.Date("2016-12-31"), frequency=24)
> frequency(data)
[1] 24
> summary(data)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-10.133   4.839  11.111  12.016  18.811  34.811 
> plot(data)
> plot(aggregate(data,FUN=mean))
> boxplot(data~cycle(data))
> library(forecast)
> acf(data)
> fit<- auto.arima(data)
> accuracy(fit)
                      ME      RMSE       MAE MPE MAPE      MASE         ACF1
Training set 0.001361921 0.8586029 0.5773431 NaN  Inf 0.2691404 5.272405e-05
> newdata<- forecast(fit, 240)
> plot(newdata)
> 
```

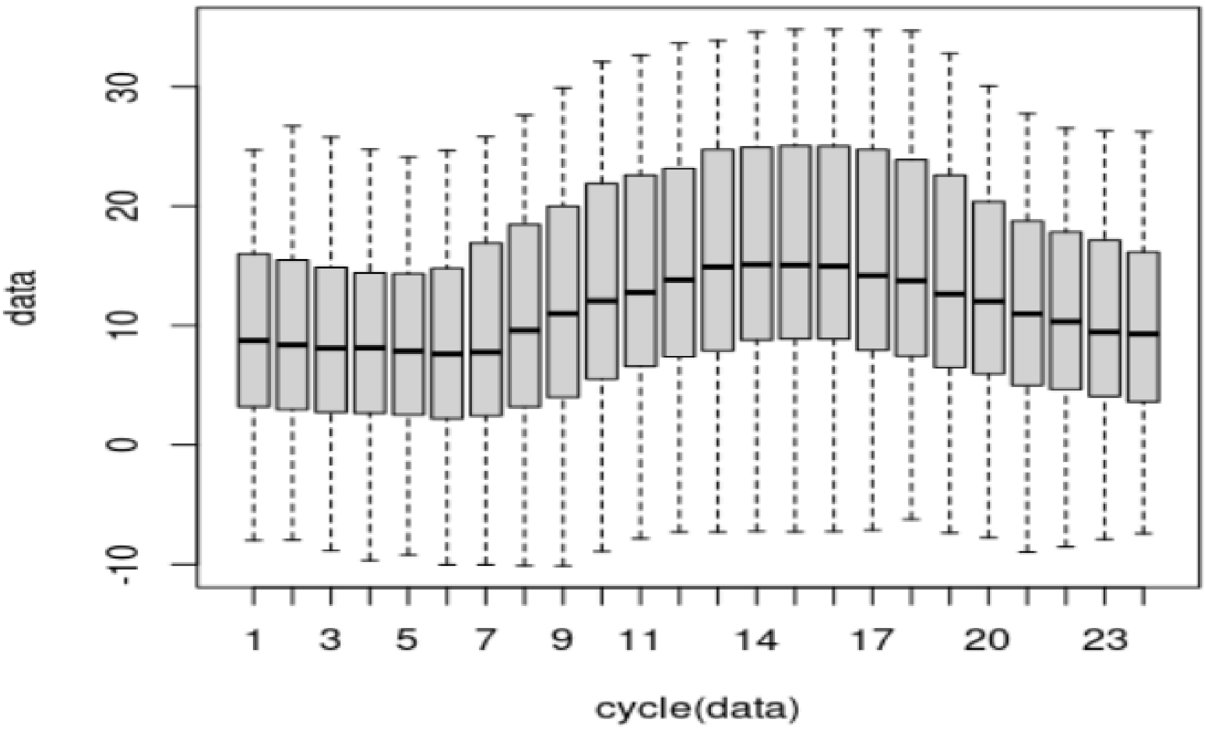


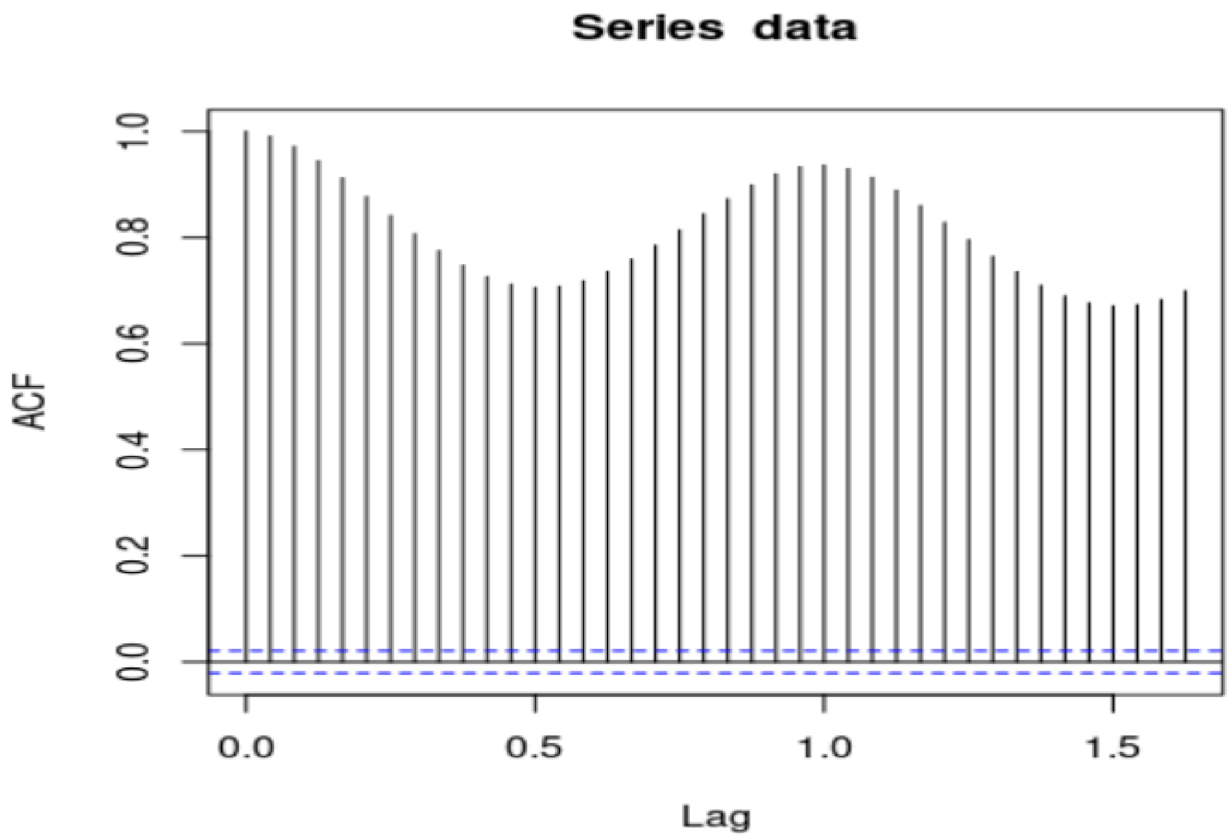
Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
-4
-2
0
2
4
-3
-2
-1
0
1
2

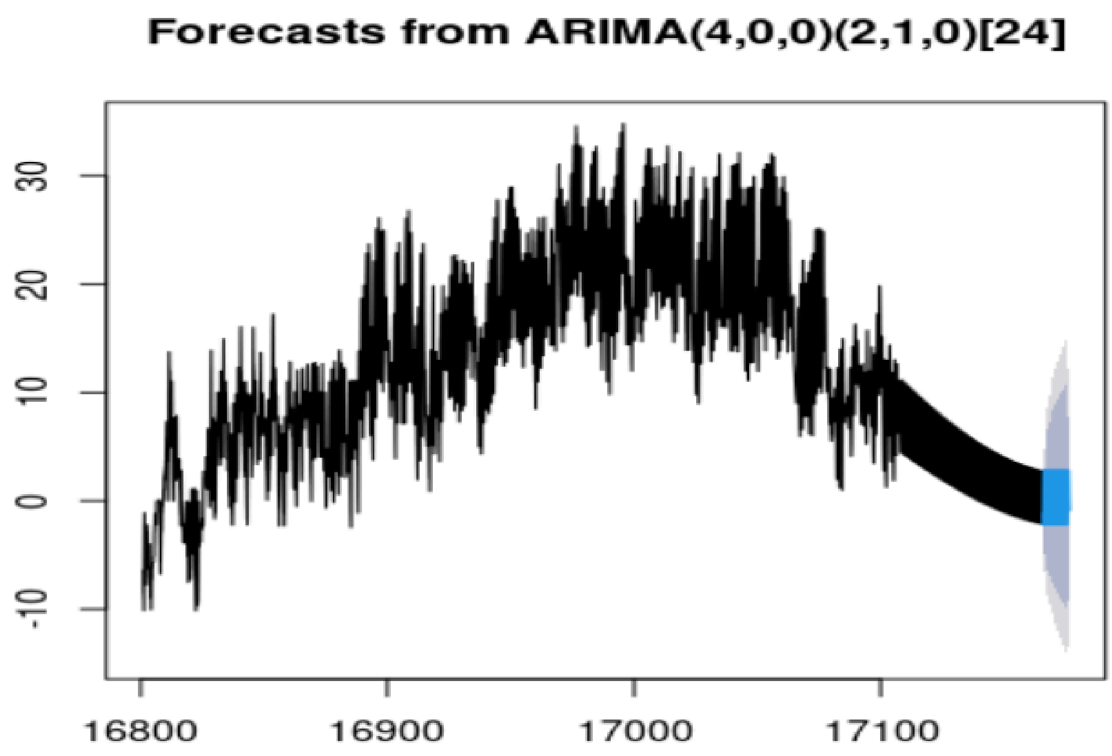


data
Time
16800
16900
17000
17100
-10
0
10
20
30


aggregate(data, FUN = mean)
0
10
20
16800
16900
17000
17100
Time


data
-10
0
10
20
30
1
3
5
7
9
11
14
17
20
23
cycle(data)


Series data
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
Lag


Forecasts from ARIMA(4,0,0)(2,1,0)[24]
30
20
10
0
-10
16800
16900
17000
17100

`

Date: 28/Mar/2024	LOGISTIC REGRESSION
EXPERIMENT – 09	

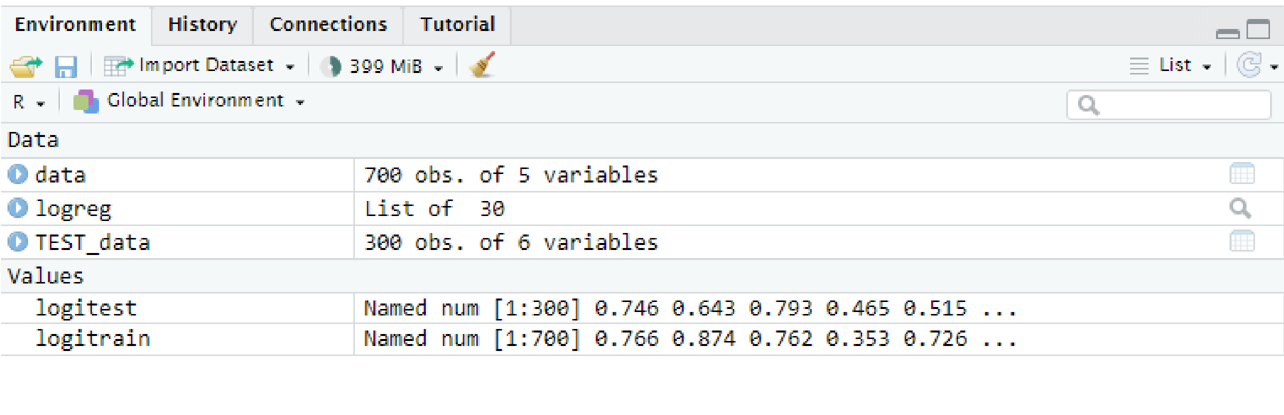
AIM: To perform Logistic Regression

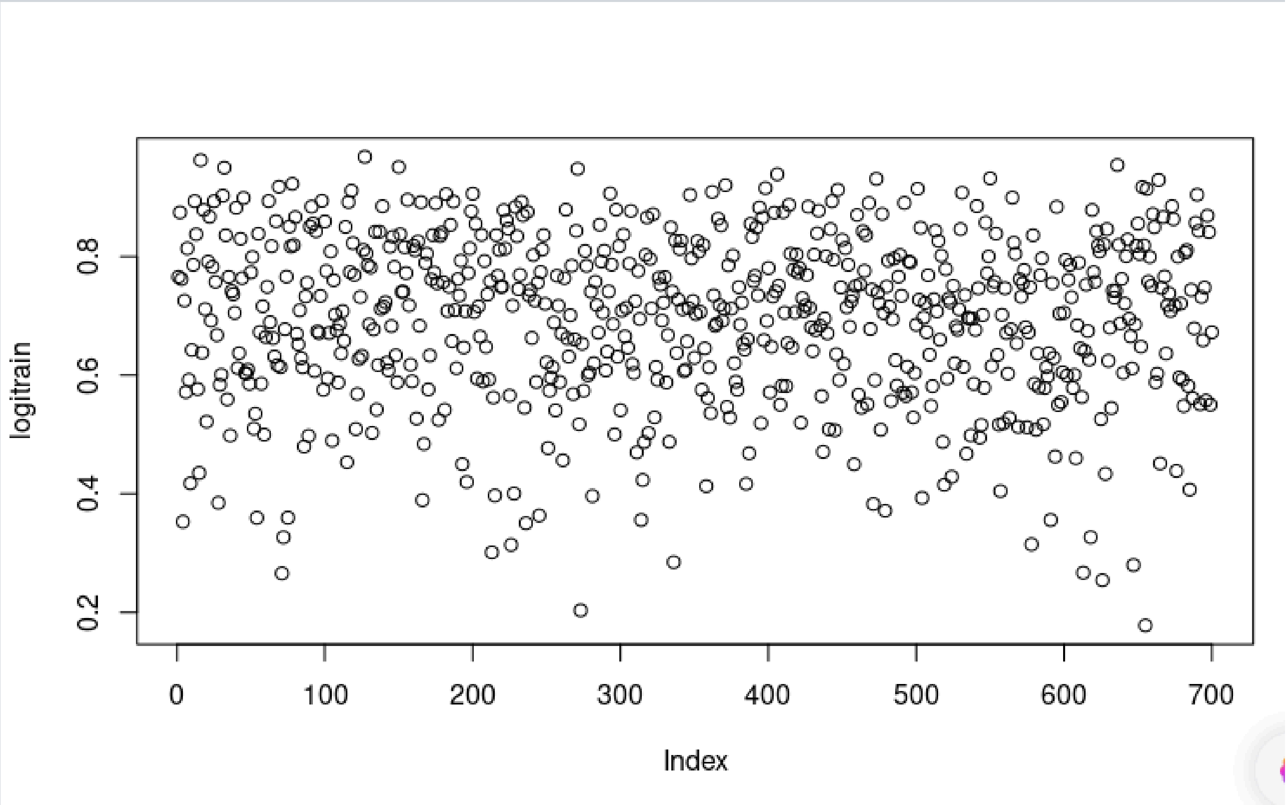
SOFTWARE REQUIRED: RStudio

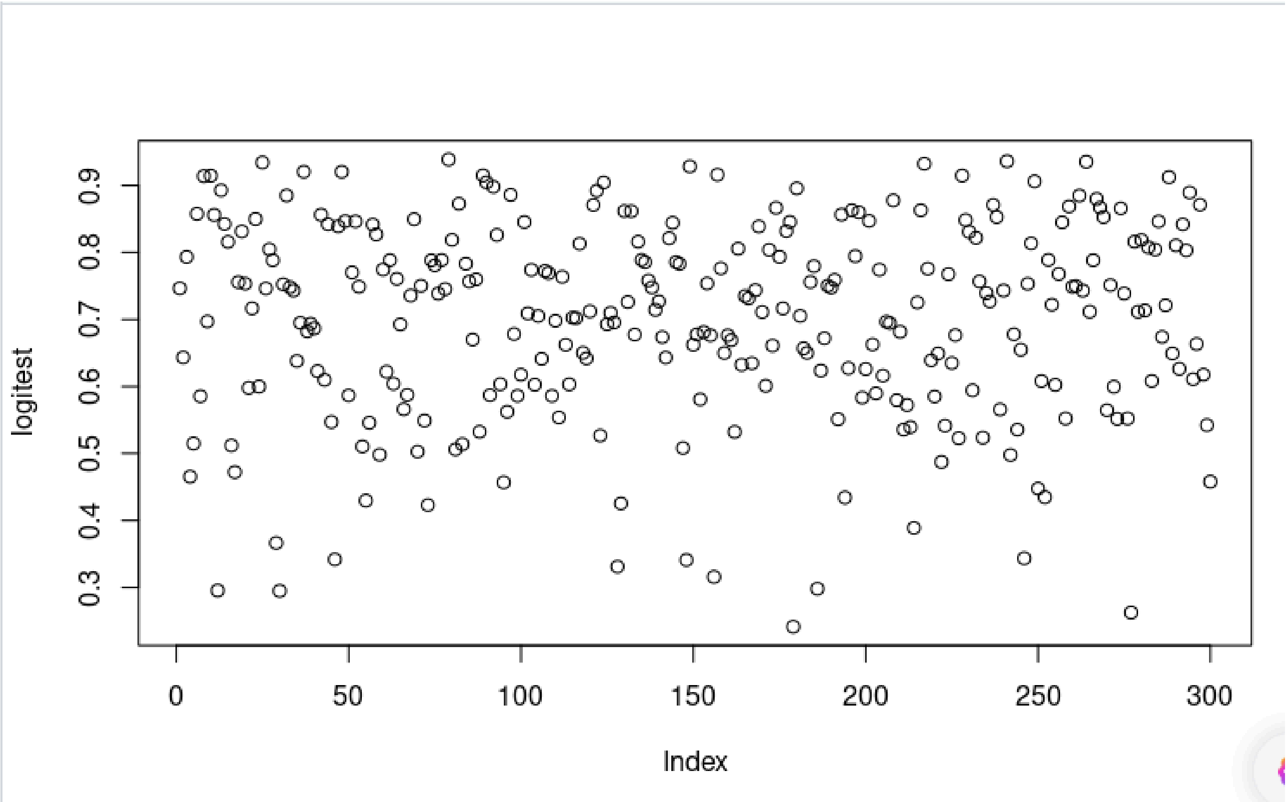
R CODE:

```
rm(list=ls())
data<-read.csv("CreditWorthiness_TRAIN.csv",stringsAsFactors = T)
logreg<-glm(formula = data$creditScore ~.,
family='binomial',data=data)
summary(logreg)
logitrain<-predict(logreg,type='response')
plot(logitrain)
tapply(logitrain,data$creditScore,mean)
TEST_data<-read.csv("CreditWorthiness_TEST.csv",stringsAsFactors =
T)
logitest <- predict(logreg,newdata=TEST_data,type='response')
plot(logitest)
tapply(logitest,TEST_data$creditScore,mean)
TEST_data[logitest<=0.7,"LogiTest"]="bad"
TEST_data[logitest>0.7,"LogiTest"]="good"
install.packages("caret")
library(caret)
confusionMatrix(table(TEST_data[,5],TEST_data[,6]),positive='good'
)
```

OUTPUT:




logitrain
0.2
0.4
0.6
0.8
0
100
200
300
400
500
600
700
Index


logitest
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
50
100
150
200
250
300
Index

```
Console   Terminal ×   Background Jobs ×
R 4.3.3 . /cloud/project/
> rm(list=ls())
> data<-read.csv("CreditWorthiness_TRAIN.csv",stringsAsFactors  = T)
> logreg<-glm(formula = data$creditScore ~., family='binomial',data=data)
> summary(logreg)

Call:
glm(formula = data$creditScore ~ ., family = "binomial", data = data)

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)    
(Intercept)               8.320e-01  4.404e-01   1.889 0.058846 .  
Cdur                     -3.240e-02  9.178e-03  -3.530 0.000416 ***
Cpurdomestic needs       -5.132e-01  7.925e-01  -0.648 0.517299    
Cpureducation            -8.836e-01  4.318e-01  -2.046 0.040730 *  
Cpurelectronics           3.648e-01  3.202e-01   1.139 0.254638    
Cpurfurniture            -1.992e-01  3.305e-01  -0.603 0.546822    
Cpurmiscellaneous        -2.014e-01  7.279e-01  -0.277 0.782048    
Cpurnew vehicle           1.160e+00  4.524e-01   2.565 0.010324 *  
Cpurrenovation           -7.147e-01  5.730e-01  -1.247 0.212308    
Cpurretaining             5.438e-01  1.124e+00   0.484 0.628615    
Cpursecond hand vehicle -5.999e-01  3.169e-01  -1.893 0.058367 .  
Camt                     -3.647e-06  4.060e-06  -0.898 0.369109    
age                       2.571e-02  8.527e-03   3.016 0.002565 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 860.23  on 699  degrees of freedom
Residual deviance: 793.13  on 687  degrees of freedom
AIC: 819.13

Number of Fisher Scoring iterations: 4

> logitrain<-predict(logreg,type='response')
> plot(logitrain)
> tapply(logitrain,data$creditScore,mean)
      bad      good 
0.6302780 0.7243343 
> TEST_data<-read.csv("CreditWorthiness_TEST.csv",stringsAsFactors = T)
> logitest <- predict(logreg,newdata=TEST_data,type='response')
> plot(logitest)
> tapply(logitest,TEST_data$creditScore,mean)
      bad      good 
0.6293894 0.7302362 
> TEST_data[logitest<=0.7,"LogiTest"]="bad"
```

```
Console   Terminal ×   Background Jobs ×
R 4.3.3 . /cloud/project/
> TEST_data[logitest>0.7,"LogiTest"]="good"
> install.packages("caret")
Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
(as 'lib' is unspecified)
trying URL 'http://rspm/default/__linux__/focal/latest/src/contrib/caret_6.0-94.tar.gz'
Content type 'application/x-gzip' length 3573770 bytes (3.4 MB)
==================================================
downloaded 3.4 MB

* installing *binary* package 'caret' ...
* DONE (caret)

The downloaded source packages are in
        '/tmp/RtmpgadiwD/downloaded_packages'
> library(caret)
Loading required package: ggplot2
Loading required package: lattice
> confusionMatrix(table(TEST_data[,5],TEST_data[,6]),positive='good')
Confusion Matrix and Statistics


       bad good
  bad   53   34
  good  82  131

               Accuracy : 0.6133
                 95% CI : (0.5557, 0.6687)
    No Information Rate : 0.55
    P-Value [Acc > NIR] : 0.01553

                  Kappa : 0.1928

 Mcnemar's Test P-Value : 1.278e-05

            Sensitivity : 0.7939
            Specificity : 0.3926
         Pos Pred Value : 0.6150
         Neg Pred Value : 0.6092
             Prevalence : 0.5500
         Detection Rate : 0.4367
   Detection Prevalence : 0.7100
      Balanced Accuracy : 0.5933

       'Positive' Class : good

> |
```

Date: 10/April/2024 **EXPERIMENT – 10**	**K- NEAREST NEIGBOR CLASSIFIER**

AIM: To perform K nearest neighbor classifier

SOFTWARE REQUIRED: RStudio

R CODE:

```
rm(list=ls())
data <- read.csv ("CreditWorthiness (1).csv", stringsAsFactors = TRUE) #This line reads the data from the CSV file named "CreditWorthiness.csv" into a data frame called data, with strings converted to factors.
str(data)
summary(data)
plot(data)
data$Cdur <-as.integer(data$Cdur)
data$Cpur <-as.integer(data$Cpur)
data$Camt <- as.integer(data$Camt)
data$age <- as.integer(data$age)
data[, -5] <- scale(data[, -5])
set.seed(123)
train_indices <- sample (nrow (data), 900)
data_TRAIN <- data[train_indices, ]
data_TEST <- data[-train_indices, ]
library (class)
library (caret)
knnpredict <- knn(train = data_TRAIN[, -5], test= data_TEST[,- 5], cl = data_TRAIN$creditScore, k = 5)
confusionMatrix(table(knnpredict, data_TEST$creditScore), positive = 'good')
```

OUTPUT:

```
> str(data)
'data.frame':   1000 obs. of  5 variables:
 $ Cdur       : int  9 15 36 48 24 27 12 12 36 36 ...
 $ Cpur       : Factor w/ 10 levels "Business","domestic needs",..: 1 4 1 1 4 8 4 10 3 4 ...
 $ Camt       : int  13790 15250 19410 144090 31690 51780 21590 9950 18070 23820 ...
 $ age        : int  27 50 61 25 26 48 29 22 37 25 ...
 $ creditScore: Factor w/ 2 levels "bad","good": 2 2 1 1 2 2 2 2 1 2 ...
> #This line displays the structure of the data dataframe, showing the data types and the first
e.
> summary(data)
```

```
e.
> summary(data)
      Cdur                          Cpur            Camt              age          creditScore
 Min.   : 4.0   electronics         :280   Min.   :  2380   Min.   :19.00   bad :300
 1st Qu.:12.0   second hand vehicle:234   1st Qu.: 13535   1st Qu.:27.00   good:700
 Median :18.0   furniture           :181   Median : 23075   Median :33.00
 Mean   :20.9   new vehicle         :103   Mean   : 32593   Mean   :35.55
 3rd Qu.:24.0   Business            : 97   3rd Qu.: 39603   3rd Qu.:42.00
 Max.   :72.0   education           : 50   Max.   :184120   Max.   :75.00
                (Other)             : 55
```

```
> confusionMatrix(table(knnpredict, data_TEST
Confusion Matrix and Statistics


knnpredict bad good
      bad    9   12
      good  21   58

               Accuracy : 0.67
                 95% CI : (0.5688, 0.7608)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.7793

                  Kappa : 0.1406

 Mcnemar's Test P-Value : 0.1637

            Sensitivity : 0.8286
            Specificity : 0.3000
         Pos Pred Value : 0.7342
         Neg Pred Value : 0.4286
             Prevalence : 0.7000
         Detection Rate : 0.5800
   Detection Prevalence : 0.7900
```

```
                 ..

Mcnemar's Test P-Value : 0.1637

           Sensitivity : 0.8286
           Specificity : 0.3000
        Pos Pred Value : 0.7342
        Neg Pred Value : 0.4286
            Prevalence : 0.7000
        Detection Rate : 0.5800
  Detection Prevalence : 0.7900
     Balanced Accuracy : 0.5643

      'Positive' Class : good

>
```