Practical No.8
Implementation of Classifications Algorithms like Navie Bayes Theorem, K-Nearest Neighbour.

Implementing data analysis in R typically involves several key steps using various packages. Below is a simplified example using the popular `dplyr` and `ggplot2` packages for data manipulation and visualization. Assume you have a dataset called `my_data.csv`.

1. Loading Data:

```R
# Install and load necessary packages
install.packages(c("dplyr", "ggplot2"))
library(dplyr)
library(ggplot2)

# Load data
my_data <- read.csv("my_data.csv")
```

2. Data Exploration:

```R
# View the structure of the dataset
str(my_data)

# Summary statistics
summary(my_data)
```

3. Data Cleaning:

```R
# Remove missing values
my_data <- na.omit(my_data)

# Filter or transform data as needed
my_data <- my_data %>% filter(column_name > 0)
```

4. Data Transformation:

```R
# Create new variables
my_data <- my_data %>% mutate(new_variable =
some_operation(existing_variable))
```

5. Data Analysis:

```R
# Use dplyr functions for analysis
result <- my_data %>%
  group_by(category_variable) %>%
  summarise(avg_value = mean(numeric_variable))
```

6. Data Visualization:

```R
# Create plots with ggplot2
```

```
ggplot(my_data, aes(x = x_variable, y = y_variable, color = category_variable)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Scatter Plot with Linear Fit") +
  theme_minimal()
```

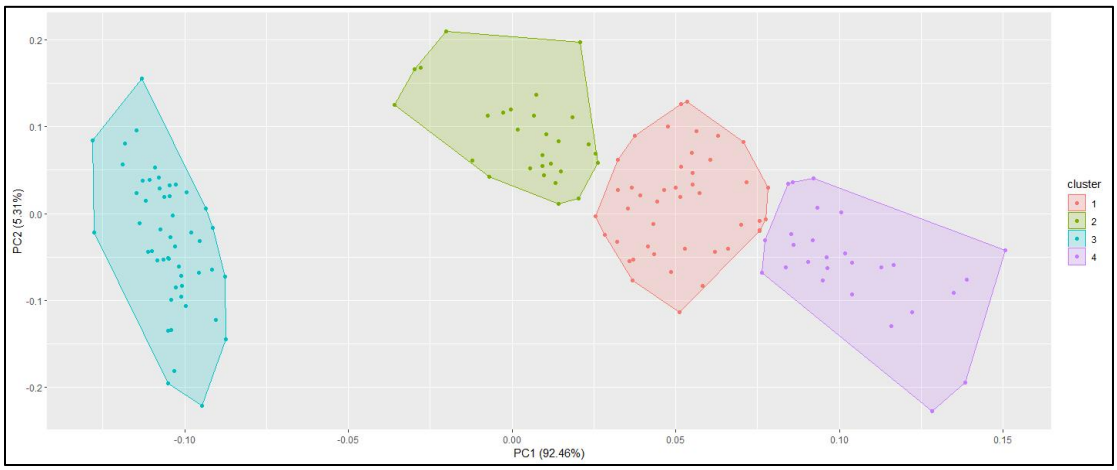
These are just basic examples. Depending on your specific analysis goals, you might perform more complex operations, statistical tests, or create more advanced visualizations. R offers a wide range of packages and functions tailored to different data analysis needs.

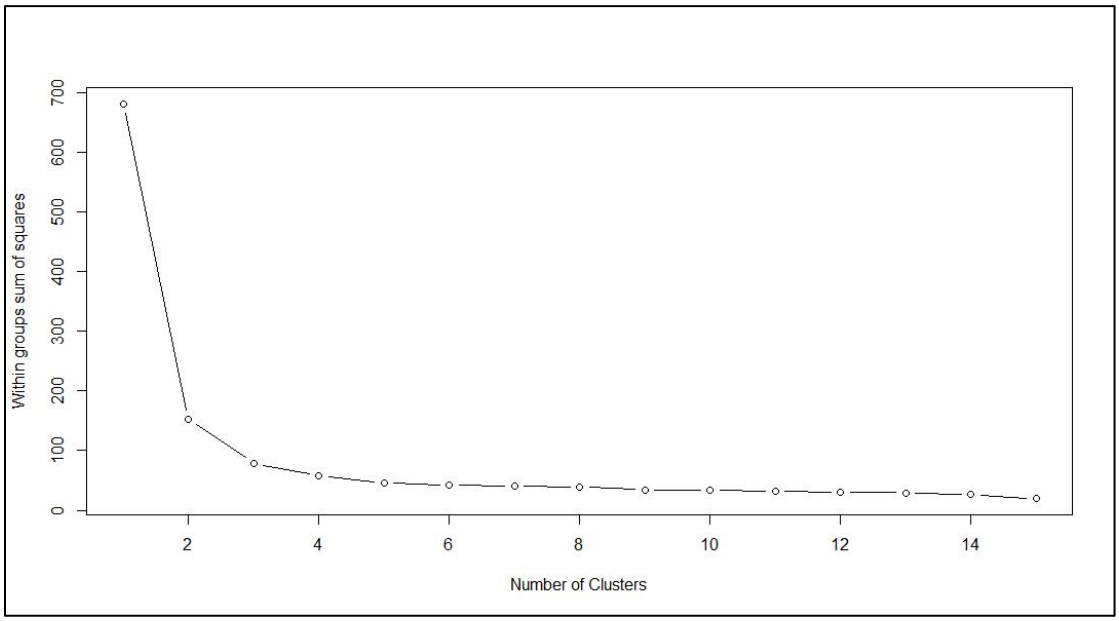
```
> library(e1071)
>
> NBdataset<-read.table("new_dataset.csv",header = TRUE,sep = ",")
>
> classifier<-naiveBayes(NBdataset[,1:4],NBdataset[,5])
>
> table(predict(classifier,NBdataset[,5]),NBdataset[,5],
+               dnn= list('predicted','actual'))
         actual
predicted no yes
      no   0   0
      yes  5   9
>
> classifier$tables
$Outlook
              Outlook
NBdataset[, 5]  Overcast     Rainy     Sunny
           no  0.0000000 0.6000000 0.4000000
           yes 0.4444444 0.2222222 0.3333333

$Temp
              Temp
NBdataset[, 5]      Cool       Hot      Mild
           no  0.2000000 0.4000000 0.4000000
           yes 0.3333333 0.2222222 0.4444444

$Humidity
              Humidity
NBdataset[, 5]      High    Normal
           no  0.8000000 0.2000000
           yes 0.3333333 0.6666667

$Windy
              Windy
NBdataset[, 5]         f         t
           no  0.4000000 0.6000000
           yes 0.6666667 0.3333333

>
> NBdataset[15,-5] <- as.factor(c(Outlook="Sunny",Temperature="Cool",Humidity="High",Wind="Strong"))
>
> print(NBdataset[15,-5])
   Outlook Temp Humidity  Windy
```

```
15   Sunny Cool     High Strong
> result<-predict(classifier,NBdataset[15,-5])
> print(result)
[1] yes
Levels: no yes
>
```

```
> library(ggplot2)
> library(ggfortify)
>
> View(iris)
> mydata <- select(iris,c(1,2,3,4))
> View(iris)
>
> wssplot <- function(data, nc=15, seed=1234){
+   wss <- (nrow(data)-1)*sum(apply(data,2,var))
+   for (i in 2:nc){
+     set.seed(seed)
+     wss[i] <- sum(kmeans(data, centers=i)$withinss)}
+   plot(1:nc, wss, type="b", xlab="Number of Clusters",
+        ylab="Within groups sum of squares")
+   wss
+ }
>
> wssplot(mydata)
 [1] 681.37060 152.34795  78.85144  57.26562  46.46117  41.70442  40.66047  39.03110
>
> KM = kmeans(mydata,4)
>
> autoplot(KM,mydata,frame=TRUE)
>
> KM$centers
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.264444    2.884444     4.886667    1.666667
2     5.532143    2.635714     3.960714    1.228571
3     5.006000    3.428000     1.462000    0.246000
4     7.014815    3.096296     5.918519    2.155556
>
```

Digram:

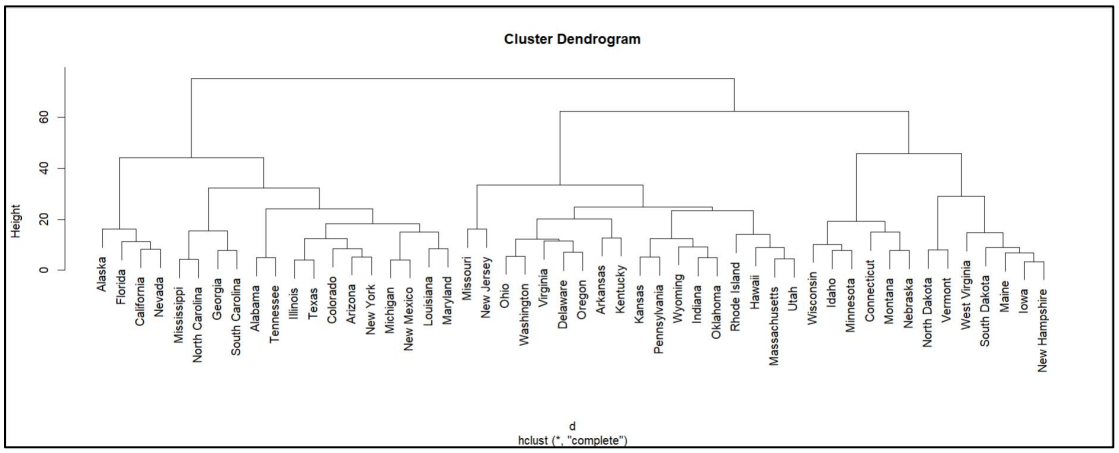




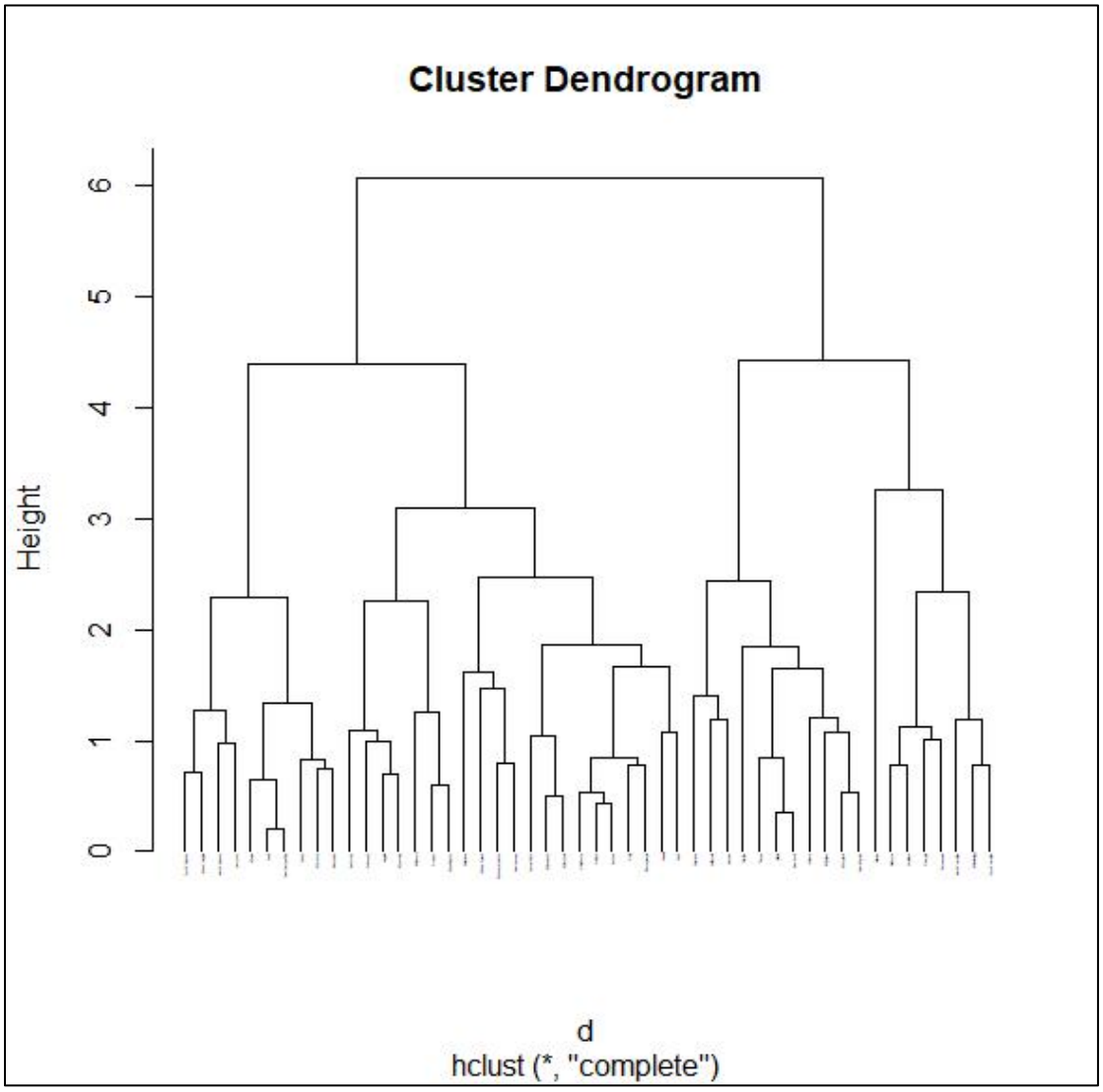
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa

```
> df<-na.omit(df)
>
> d<-scale(df)
>
> head(d)
               Murder   Assault   UrbanPop
Alabama    1.24256408 0.7828393 -0.5209066
Alaska     0.50786248 1.1068225 -1.2117642
Arizona    0.07163341 1.4788032  0.9989801
Arkansas   0.23234938 0.2308680 -1.0735927
California 0.27826823 1.2628144  1.7589234
Colorado   0.02571456 0.3988593  0.8608085
                 Rape
Alabama    -0.003416473
Alaska      2.484202941
Arizona     1.042878388
Arkansas   -0.184916602
California  2.067820292
Colorado    1.864967207
>
```

```
> d<-dist(d,method="euclidean")
> hc<-hclust(d,method = "complete")
>
> plot(hc)
> d<-dist(d,method="euclidean")
> hc<-hclust(d,method = "complete")
>
```



plot(hc,cex=0.5,hang=-1)



KNN alogorithm

```
rm(list = ls())
install.packages("class",dependencies = TRUE)
library(class)
install.packages("caret")
library(caret)

diabetics=read.csv('pima-indians-diabetes.csv')
class(diabetics$mass)
```

```
[1] "numeric"
>
```

```
str(diabetics)
```

```
> str(diabetics)
'data.frame':   768 obs. of  9 variables:
 $ preg : int  6 1 8 1 0 5 3 10 2 8 ...
 $ plas : int  148 85 183 89 137 116 78 115 197 125 ...
 $ pres : int  72 66 64 66 40 74 50 0 70 96 ...
 $ skin : int  35 29 0 23 35 0 32 0 45 0 ...
 $ test : int  0 0 0 94 168 0 88 0 543 0 ...
 $ mass : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ pedi : num  0.627 0.351 0.672 0.167 2.288 ...
 $ age  : int  50 31 32 21 33 30 26 29 53 54 ...
 $ class: int  1 0 1 0 1 0 1 0 1 1 ...
```

```
diabetics[,'class']=factor(diabetics[,'class'])
```

```
str(diabetics)
```

```
> str(diabetics)
'data.frame':   768 obs. of  9 variables:
 $ preg : int  6 1 8 1 0 5 3 10 2 8 ...
 $ plas : int  148 85 183 89 137 116 78 115 197 125 ...
 $ pres : int  72 66 64 66 40 74 50 0 70 96 ...
 $ skin : int  35 29 0 23 35 0 32 0 45 0 ...
 $ test : int  0 0 0 94 168 0 88 0 543 0 ...
 $ mass : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ pedi : num  0.627 0.351 0.672 0.167 2.288 ...
 $ age  : int  50 31 32 21 33 30 26 29 53 54 ...
 $ class: Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 1 2 2 ...
>
```

```
test = diabetics[501:768,]
train=diabetics[1:500,]

pred_test=knn(train[,-9],test[,-9],train$class,k=2)
```

```
> pred_test
  [1] 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0
 [45] 0 1 1 1 0 1 1 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 0 0
 [89] 1 0 1 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 0 1 0 0 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 1 1 1 0 0 0
[133] 0 1 1 1 0 0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 1
[177] 1 0 1 1 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0
[221] 0 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1
[265] 0 0 1 0
Levels: 0 1
>
```

pred_test

```
pred_test   0   1
        0 129  33
        1  53  53
```

confusion=table(pred_test,test$class)
Confusion

```
pred_test   0   1
        0 129  33
        1  53  53
```

sum(diag(confusion))/nrow(test)

```
> sum(diag(confusion))/nrow(test)
[1] 0.6791045
```

confusionMatrix(pred_test,test$class)

```
> confusionMatrix(pred_test,test$class)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 129  33
         1  53  53

               Accuracy : 0.6791
                 95% CI : (0.6196, 0.7346)
    No Information Rate : 0.6791
    P-Value [Acc > NIR] : 0.52917

                  Kappa : 0.3063

 Mcnemar's Test P-Value : 0.04048

            Sensitivity : 0.7088
            Specificity : 0.6163
         Pos Pred Value : 0.7963
         Neg Pred Value : 0.5000
             Prevalence : 0.6791
         Detection Rate : 0.4813
   Detection Prevalence : 0.6045
      Balanced Accuracy : 0.6625

       'Positive' Class : 0
```

I. Linear Regression in R

▷ Regression analysis is a very widely used statistical tool to establish a relationship model between two variables.

▷ One of these variable is called predictor variable whose value is gathered through experiments.

▷ The other variable is called response variable whose value is derived from the predictor variable.

▷ Linear regression is used to predict the value of an outcome variable Y based on one or more input predictor variables X.

▷ Mathematically a linear relationship represents a straight line when plotted as a graph.

▷ The general mathematical equation for a linear regression is −

▷ =b0 + b1 *

▷ Following is the description of the parameters used −

○y is the response variable.

○x is the predictor variable.

○b1 – slope

○b0 - intercept

○Collectively, they are called regression coefficients.

▷ For example, we want to predict weight (y) from height (x), the linear regression model can be represented by the following equation

▷ Weight= b0 + b1 * height

○b1 is called slope because it defines the slope of the line or how x translates into a y i.e by how much y is affected by change in x

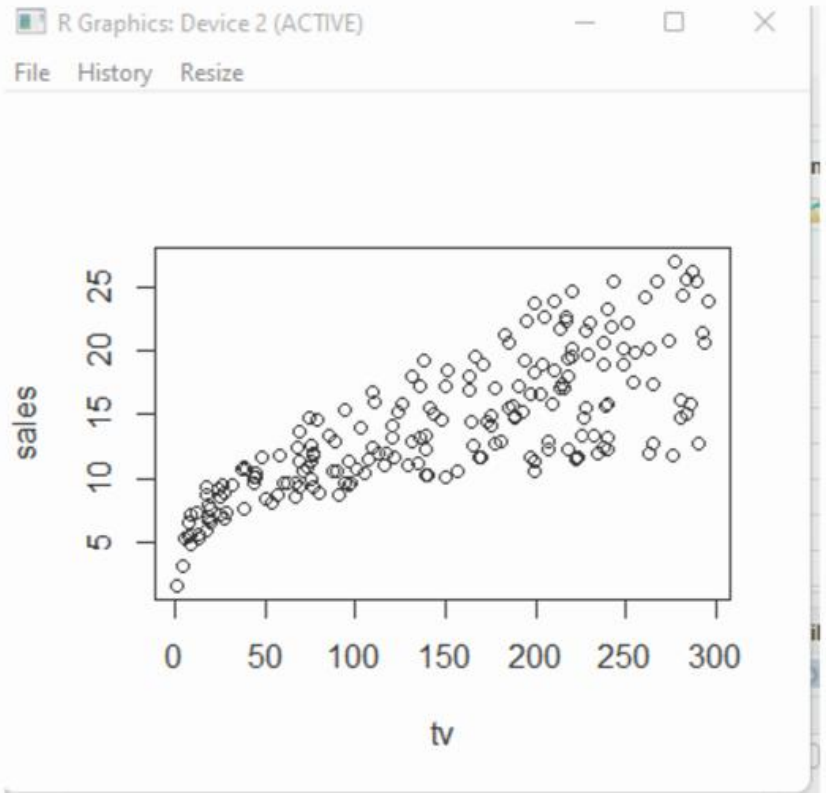
▷ The goal is to find best estimates for the coefficients to minimize the error in predicting y
from x

Download marshall adv. csv Dataset to perform operations

plot(tv,sales,pch=16,cex=1,col='blue',main='Tv vs Sales',xlab = 'TV',ylab = 'Sales')

```
200 NA 232.1   8.6       8.7  13.4
> data<-read.csv("advr.csv")
> data
       TV Radio Newspaper Sales
1   230.1  37.8       69.2  22.1
2    44.5  39.3       45.1  10.4
3    17.2  45.9       69.3   9.3
4   151.5  41.3       58.5  18.5
5   180.8  10.8       58.4  12.9
6     8.7  48.9       75.0   7.2
7    57.5  32.8       23.5  11.8
8   120.2  19.6       11.6  13.2
9     8.6   2.1        1.0   4.8
10  199.8   2.6       21.2  10.6
```

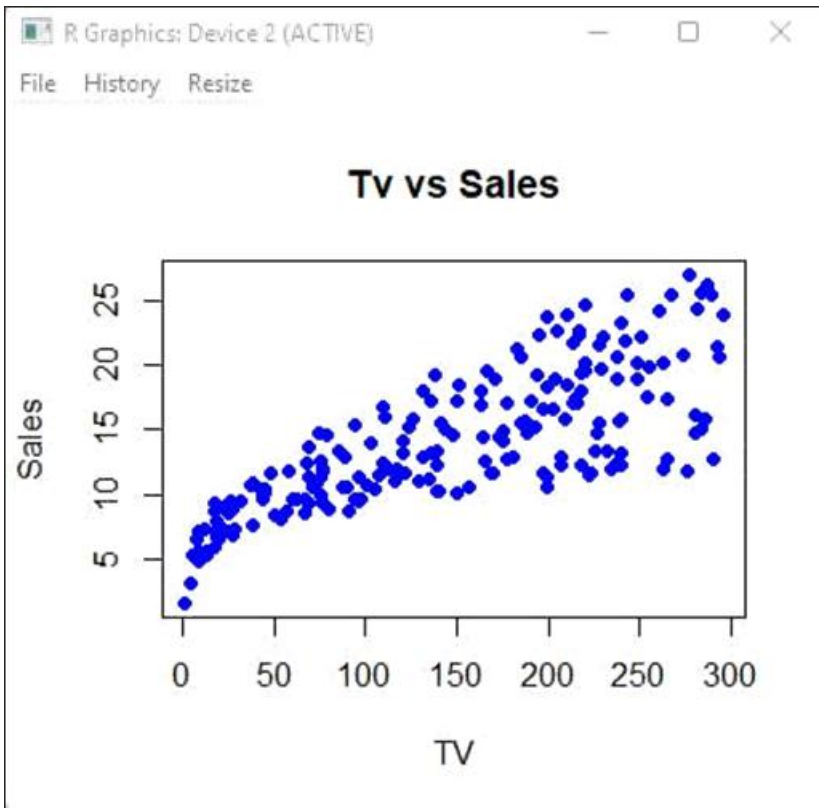
```
tv<-data$TV
tv
sales<data$Sales
sales
```



model<-lm

summary(model)

attributes(model)



coefficients(model)

```
> model<-lm(sales~tv)
> summary(model)

Call:
lm(formula = sales ~ tv)

Residuals:
    Min      1Q  Median      3Q     Max
-8.3860 -1.9545 -0.1913  2.0671  7.2124

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.032594   0.457843   15.36   <2e-16 ***
tv          0.047537   0.002691   17.67   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.259 on 198 degrees of freedom
Multiple R-squared:  0.6119,    Adjusted R-squared:  0.6099
F-statistic: 312.1 on 1 and 198 DF,  p-value: < 2.2e-16

> |
```

abline(model)

```
> attributes(model)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"          "fitted.values" "assign"        "qr"
 [8] "df.residual"   "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"
```

```
> coefficients(model)
(Intercept)          tv
 7.03259355  0.04753664
>
```

```
> coefficients(model)
(Intercept)          tv
 7.03259355  0.04753664
>
```

```
> coef(model)
(Intercept)          tv
 7.03259355  0.04753664
>
```