EXPERIMENT NO:1

AIM: Implement basic commands in R,R Graphics,Indexeing data,loading data,Additional graphical and numerical summaries .

BASIC COMMANDS IN R:

DESCRIPTION: R uses functions to perform operations. To run a function called funcname , we type funcname(input1, input2) , where the inputs (or arguments) input1 and input2 tell R how to run the function. A function can have any number of inputs. For example, to create a vector of numbers, we use the function c() (for concatenate). Any numbers inside the parentheses are joined together.

CODE:

```
x <- c(1,3,2,5)
x
x = c(1,6,2)
x
y = c(1,4,3)
length(x)
length(y)
x+y
ls()
rm(x,y)
ls()
character(0)
rm(list=ls())
?matrix
x=matrix(data=c(1,2,3,4), nrow=2, ncol=2)
x'
```

OUTPUT:

```
R 4.3.0 · ~/
> x <- c(1,3,2,5)
> x
[1] 1 3 2 5
> x = c(1,6,2)
> x
[1] 1 6 2
> y = c(1,4,3)
> length(x)
[1] 3
> length(y)
[1] 3
> x+y
[1]  2 10  5
> ls()
[1] "x" "y"
> rm(x,y)
> ls()
character(0)
> character(0)
character(0)
> rm(list=ls())
> ?matrix
> x=matrix(data=c(1,2,3,4), nrow=2, ncol=2)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

CODE:

```
x=matrix(c(1,2,3,4) ,2,2)
matrix(c(1,2,3,4) ,2,2,byrow=TRUE)
sqrt(x)
x^2
x=rnorm(50)
y=x+rnorm(50,mean=50,sd=.1)
cor(x,y)
set.seed(1303)
rnorm(50)
set.seed(3)
y=rnorm(100)
mean(y)
var(y)
sqrt(var(y))
sd(y)
```

OUTPUT:

```
> x=matrix(c(1,2,3,4) ,2,2)
> matrix(c(1,2,3,4) ,2,2,byrow=TRUE)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> sqrt(x)
         [,1]     [,2]
[1,] 1.000000 1.732051
[2,] 1.414214 2.000000
> x^2
     [,1] [,2]
[1,]    1    9
[2,]    4   16
> x=rnorm(50)
> y=x+rnorm(50,mean=50,sd=.1)
> cor(x,y)
[1] 0.9934096
> set.seed(1303)
> rnorm(50)
 [1] -1.1439763145  1.3421293656  2.1853904757  0.5363925179  0.0631929665  0.5022344825 -0.0004167247  0.5658198405 -0.5725226890
[10] -1.1102250073 -0.0486871234 -0.6956562176  0.8289174803  0.2066528551 -0.2356745091 -0.5563104914 -0.3647543571  0.8623550343
[19] -0.6307715354  0.3136021252 -0.9314953177  0.8238676185  0.5233707021  0.7069214120  0.4202043256 -0.2690521547 -1.5103172999
[28] -0.6902124766 -0.1434719524 -1.0135274099  1.5732737361  0.0127465055  0.8726470499  0.4220661905 -0.0188157917  2.6157489689
[37] -0.6931401748 -0.2663217810 -0.7206364412  1.3677342065  0.2640073322  0.6321868074 -1.3306509858  0.0268888182  1.0406363208
[46]  1.3120237985 -0.0300020767 -0.2500257125  0.0234144857  1.6598706557
> set.seed(3)
> y=rnorm(100)
> mean(y)
[1] 0.01103557
> var(y)
[1] 0.7328675
> sqrt(var(y))
[1] 0.8560768
> sd(y)
[1] 0.8560768
> sd(y)
[1] 0.8560768
```

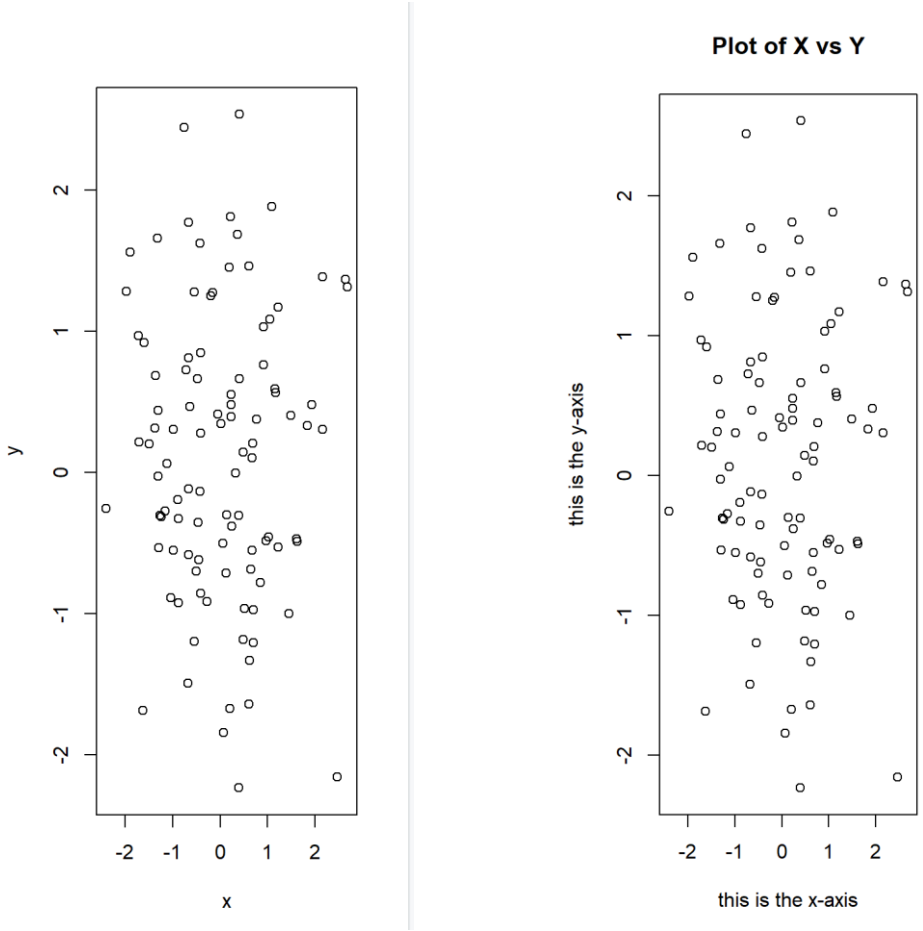
GRAPHICS IN R:

DESCRIPTION: The plot() function is the primary way to plot data in R . For instance, plot(x,y) produces a scatterplot of the numbers in x versus the numbers in y . There are many additional options that can be passed in to the plot() function.

CODE:

```
x=rnorm(100)
y=rnorm(100)
plot(x,y)
plot(x,y,xlab="this is the x-axis",ylab="this is the y-axis", main="Plot of X vs Y")
pdf("Figure.pdf")
plot(x,y,col="green")
dev.off()
x=seq(1,10)
x
x=1:10
x
```

OUTPUT:

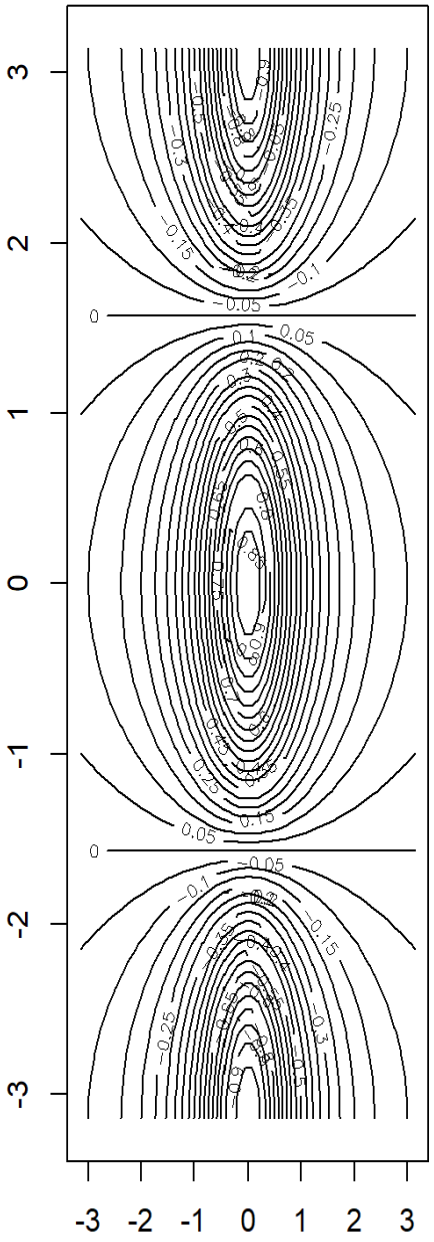


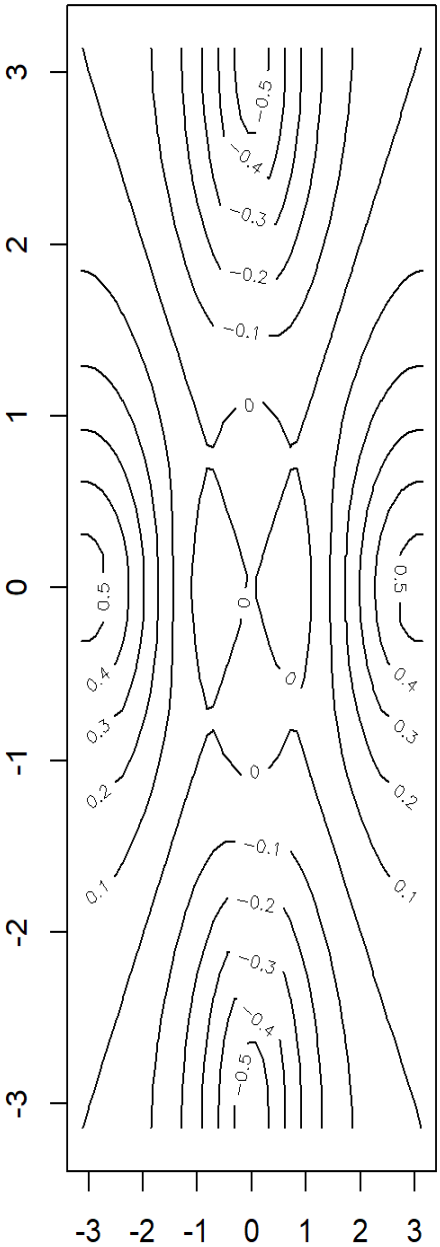

```
> x=seq(1,10)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x=1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
```

CODE:

```
x=seq(-pi,pi,length =50)
y=x
f=outer(x,y,function(x,y)cos(y)/(1+x^2))
contour(x,y,f)
contour(x,y,f,nlevels=45,add=T)
fa=(f-t(f))/2
contour(x,y,fa,nlevels=15)
```

OUTPUT:

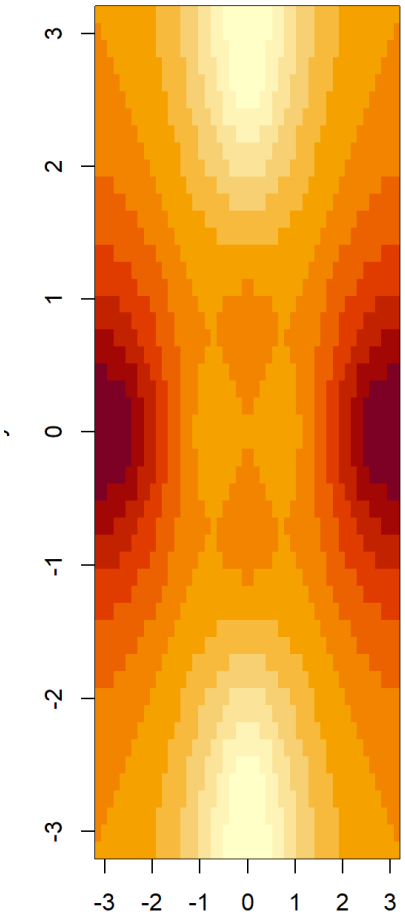


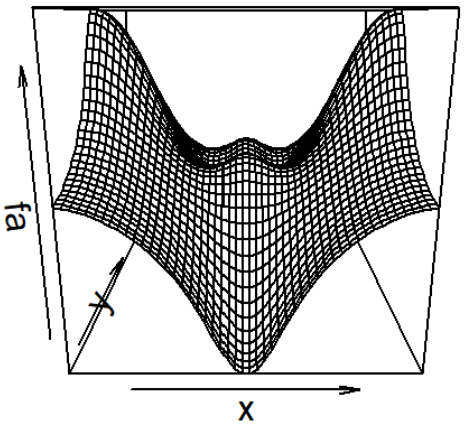


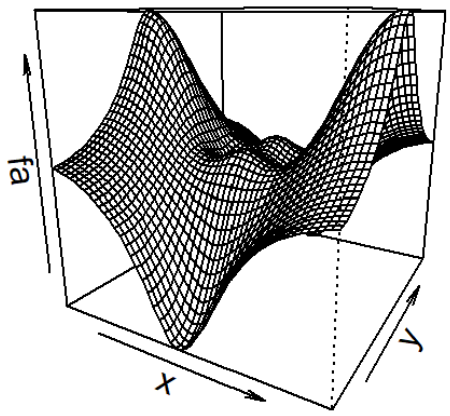
CODE:

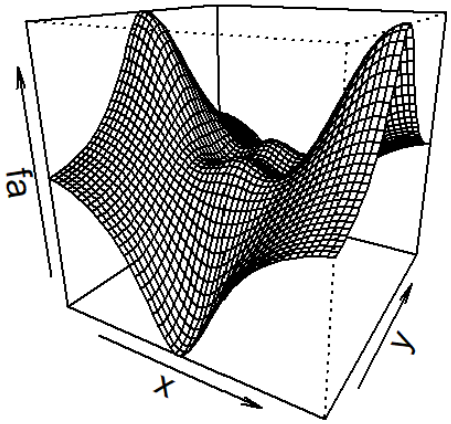
```
image(x,y,fa)
persp(x,y,fa)
persp(x,y,fa,theta =30)
persp(x,y,fa,theta=30,phi=20)
```

OUTPUT:



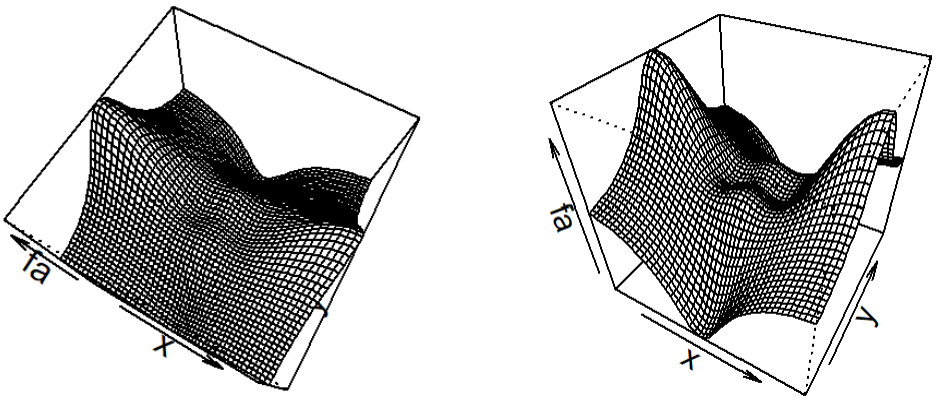






CODE:
persp(x,y,fa,theta=30,phi=70)
persp(x,y,fa,theta=30,phi=40)

OUTPUT:



INDEXING DATA:

DESCRIPTION: We often wish to examine part of a set of data. Suppose that our data is stored in the matrix A .

CODE:
A=matrix(1:16,4,4)
A
A[2,3]
A[c(1,3),c(2,4)]
A[1:3,2:4]
A[1:2,]

OUTPUT:

```
R 4.3.0 · ~/
> A=matrix(1:16,4,4)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
> A[2,3]
[1] 10
> A[c(1,3),c(2,4)]
     [,1] [,2]
[1,]    5   13
[2,]    7   15
> A[1:3,2:4]
     [,1] [,2] [,3]
[1,]    5    9   13
[2,]    6   10   14
[3,]    7   11   15
> A[1:2,]
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
```

CODE:

```
A[,1:2]
A[1,]
A[-c(1,3),]
A[-c(1,3),-c(1,3,4)]
dim(A)
```

OUTPUT:

```
> A[,1:2]
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
> A[1,]
[1]  1  5  9 13
> A[-c(1,3),]
     [,1] [,2] [,3] [,4]
[1,]    2    6   10   14
[2,]    4    8   12   16
> A[-c(1,3),-c(1,3,4)]
[1] 6 8
> dim(A)
[1] 4 4
>
```

LOADING DATA:

DESCRIPTION: For most analyses, the first step involves importing a data set into R . The read.table() function is one of the primary ways to do this. The help file read.table() contains details about how to use this function. We can use the function write.table() to export data.

CODE:

```
auto<- read_excel("C:/Users/Hp/OneDrive/Desktop/ML/auto_mpg.csv.xlsx")
str(auto)
auto[1:4,]
auto=na.omit(auto)
dim(auto)
names(auto)
```

OUTPUT:

```
> auto<- read_excel("C:/Users/Hp/OneDrive/Desktop/ML/auto_mpg.csv.xlsx")
> str(auto)
tibble [398 × 9] (S3: tbl_df/tbl/data.frame)
 $ mpg         : num [1:398] 18 15 18 16 17 15 14 14 14 15 ...
 $ cylinders   : num [1:398] 8 8 8 8 8 8 8 8 8 8 ...
 $ displacement: num [1:398] 307 350 318 304 302 429 454 440 455 390 ...
 $ horsepower  : chr [1:398] "130" "165" "150" "150" ...
 $ weight      : num [1:398] 3504 3693 3436 3433 3449 ...
 $ acceleration: num [1:398] 12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
 $ model year  : num [1:398] 70 70 70 70 70 70 70 70 70 70 ...
 $ origin      : num [1:398] 1 1 1 1 1 1 1 1 1 1 ...
 $ car name    : chr [1:398] "chevrolet chevelle malibu" "buick skylark 320" "plymouth satellite" "amc rebel sst" ...
> auto[1:4,]
# A tibble: 4 × 9
    mpg cylinders displacement horsepower weight acceleration `model year` origin `car name`
  <dbl>     <dbl>        <dbl> <chr>       <dbl>        <dbl>        <dbl>  <dbl> <chr>
1    18         8          307 130          3504         12            70      1 chevrolet chevelle malibu
2    15         8          350 165          3693         11.5          70      1 buick skylark 320
3    18         8          318 150          3436         11            70      1 plymouth satellite
4    16         8          304 150          3433         12            70      1 amc rebel sst
> auto=na.omit(auto)
> dim(auto)
[1] 398   9
> names(auto)
[1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"       "acceleration" "model year"   "origin"       "car name"
```

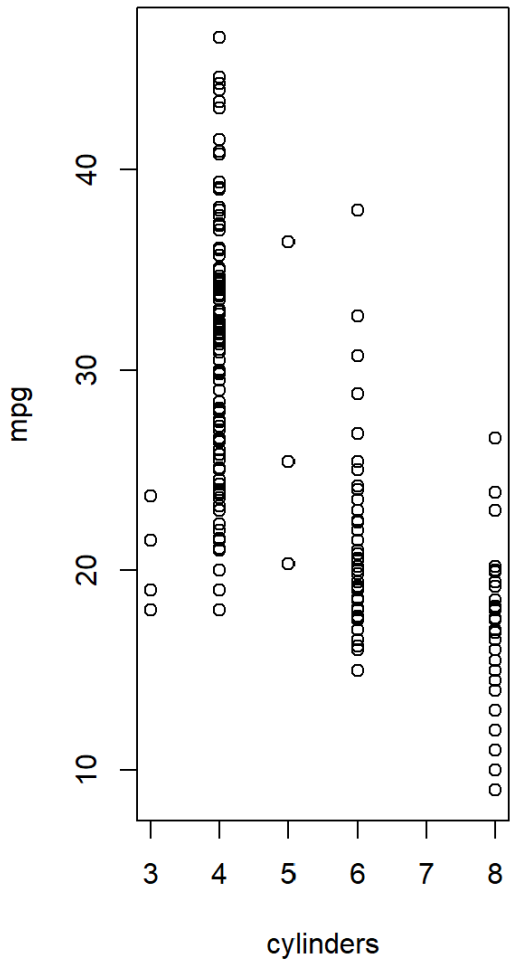
ADDITIONAL GRAPHICAL AND NUMERICAL SUMMARIES :

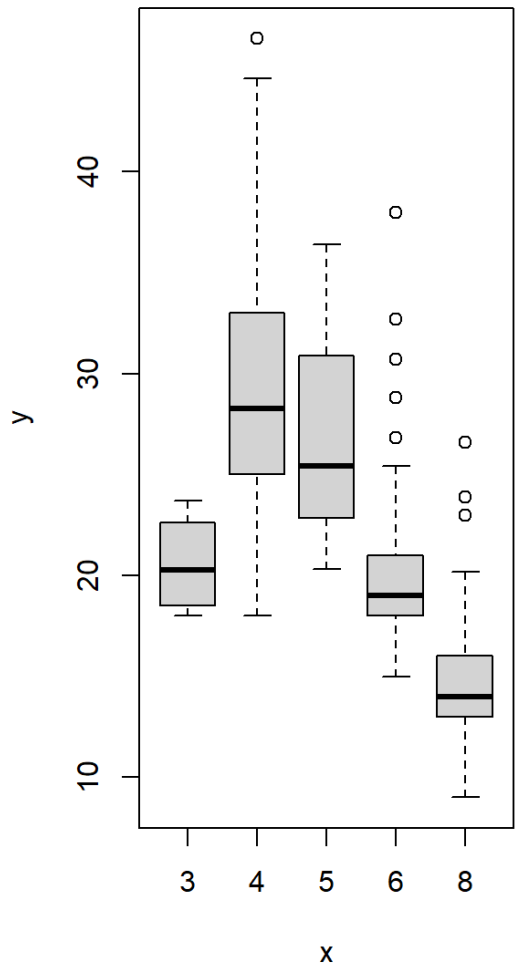
DESCRIPTION: We can use the plot() function to produce scatterplots of the quantitative variables. However, simply typing the variable names will produce an error message, because R does not know to look in the Auto data set for those variables.

CODE:

```
attach(auto)
plot(cylinders, mpg)
cylinders=as.factor(cylinders)
plot(cylinders,mpg)
```

OUTPUT:

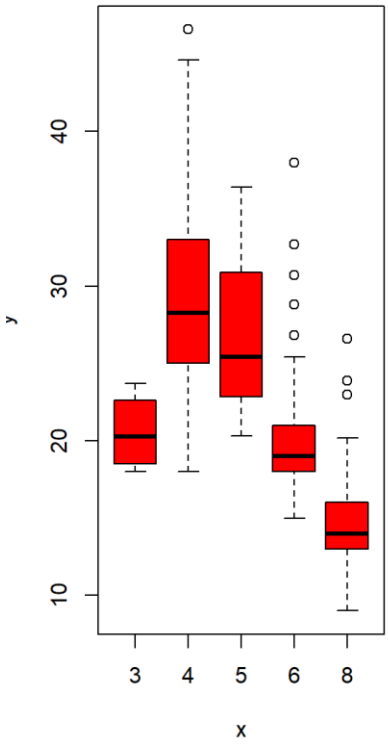


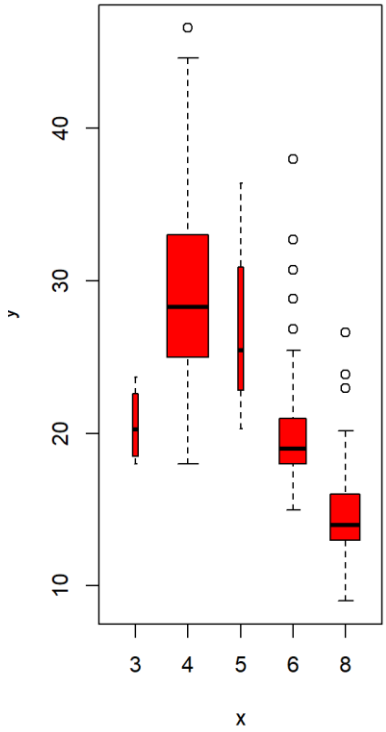


CODE:

```
plot(cylinders,mpg,col="red")
plot(cylinders,mpg,col="red",varwidth=T)
```

OUTPUT:

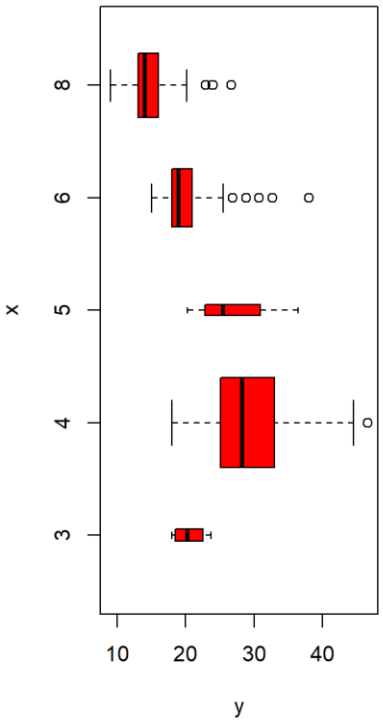


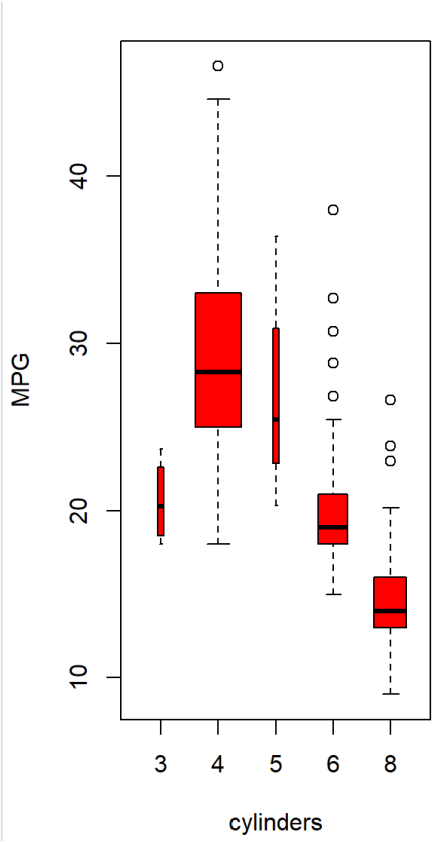


CODE:

```
plot(cylinders,mpg,col="red",varwidth=T,horizontal =T)
plot(cylinders,mpg,col="red",varwidth=T,xlab="cylinders",ylab="MPG")
```

OUTPUT:

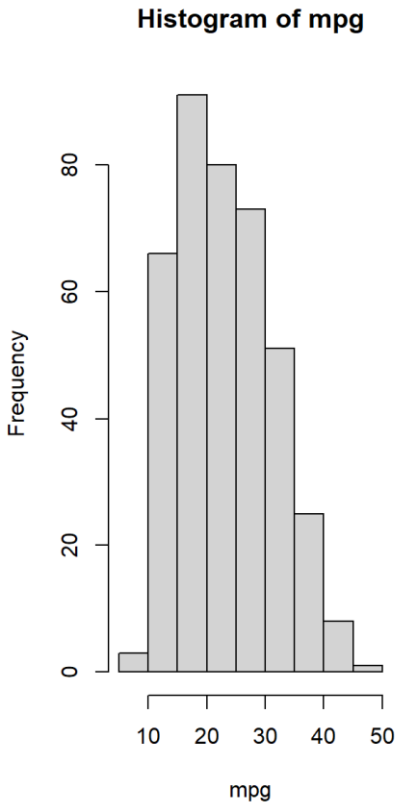


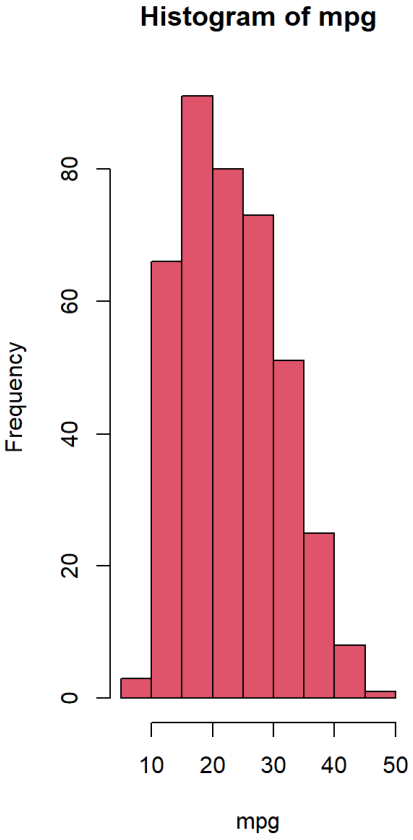


CODE:

```
hist(mpg)
hist(mpg,col=2)
```

OUTPUT:

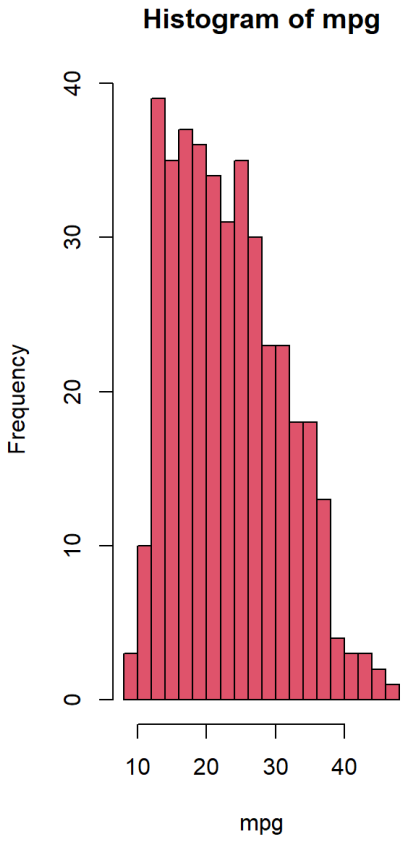


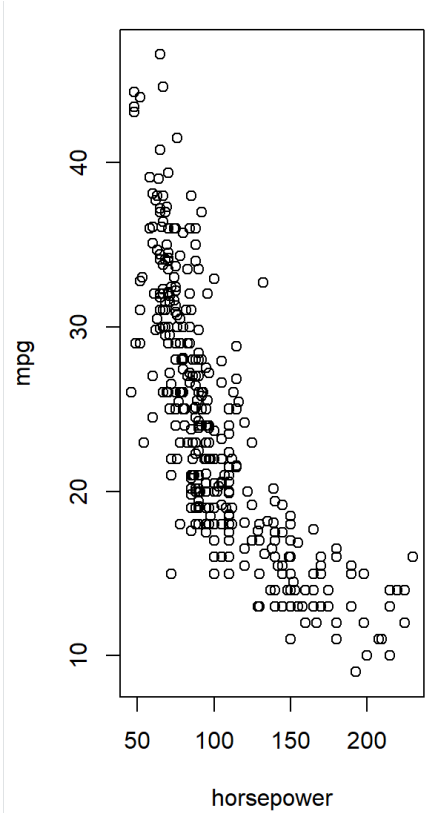


CODE:

```
hist(mpg,col=2,breaks=15)
plot(horsepower,mpg)
```

OUTPUT:

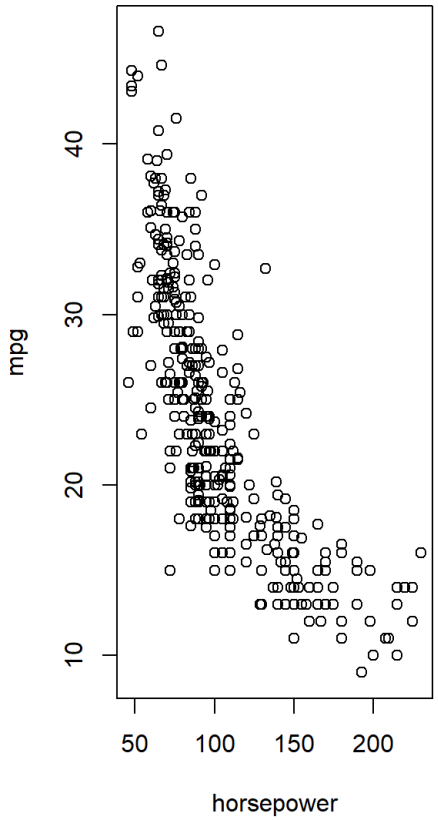




CODE:

```
identify(horsepower,mpg,car.name)
summary(auto)
summary(auto$mpg)
```

OUTPUT:




```
> summary(auto)
      mpg          cylinders      displacement    horsepower          weight      acceleration     model year        origin
 Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Length:398         Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
 1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.2   Class :character   1st Qu.:2224   1st Qu.:13.82   1st Qu.:73.00   1st Qu.:1.000
 Median :23.00   Median :4.000   Median :148.5   Mode  :character   Median :2804   Median :15.50   Median :76.00   Median :1.000
 Mean   :23.51   Mean   :5.455   Mean   :193.4                      Mean   :2970   Mean   :15.57   Mean   :76.01   Mean   :1.573
 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0                      3rd Qu.:3608   3rd Qu.:17.18   3rd Qu.:79.00   3rd Qu.:2.000
 Max.   :46.60   Max.   :8.000   Max.   :455.0                      Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
   car name
 Length:398
 Class :character
 Mode  :character


> summary(auto$mpg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9.00   17.50   23.00   23.51   29.00   46.60
```