Week 1 Assignment

Durga Deepak Valluri

2024-09-09

#2.3.1 Basic Commands

```
x <-c(1, 3, 2, 5) #Command to join the numbers 1,3,2,5 together and save them as vector 'x'
x #Returning the vector x
```

```
## [1] 1 3 2 5
```

```
#Another way of saving values to a variable. Used "=" instead of "<-"

x = c(1, 6, 2)
x
```

```
## [1] 1 6 2
```

```
y = c(1, 4, 3)

#length() function to check the length of a variable
length(x)
```

```
## [1] 3
```

```
length(y)
```

```
## [1] 3
```

```
x + y #To add two sets of numbers together
```

```
## [1]  2 10  5
```

```
ls() #To return the list of objects(Data and functions) that were saved so far
```

```
## [1] "x" "y"
```

```
rm(x, y) #To delete any object that we don't want

ls() #listing the objects after removing x and y in the previous step using rm(X, y)
```

```
## character(0)
```

```
rm(list = ls()) #To remove all objects at once
```

```r
#Using matrix function taking 3 arguments: data, number of rows, number of columns
x <- matrix(data = c(1, 2, 3, 4), nrow = 2, ncol = 2)
x
```

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

```r
#The same output as above can be achieved without mentioning the title of the arguments like data, nrow, ncol
x <- matrix(c(1, 2, 3, 4), 2, 2)
x
```

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

```r
#By default, R creates matrices by successively filling in columns. byrow = TRUE option can be used to populate the matrix in order of the rows

#To be noticed that the matrix function in this block is not assigned to any other variable
matrix(c(1, 2, 3, 4), 2, 2, byrow = TRUE)
```

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

```r
#sqrt() function returns the square root of each element of a vector or a matrix
sqrt(x)
```

```
##          [,1]     [,2]
## [1,] 1.000000 1.732051
## [2,] 1.414214 2.000000
```

```r
#squaring the elements of matrix x
x^2
```

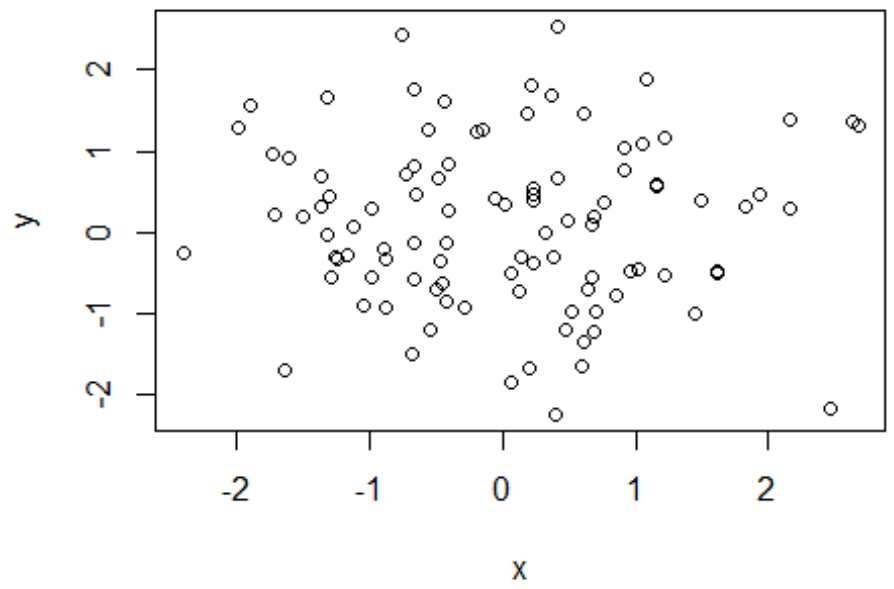
```
##      [,1] [,2]
## [1,]    1    9
## [2,]    4   16
```

```r
#rnorm() function generates a vector of random normal variables,
x <- rnorm(50)

#the first argument in rnorm(), "n" is the sample size. We get a different answer everytime this function is called.
y <- x + rnorm(50, mean = 50, sd = .1)

#cor() function to compute the correlation between two correlated set of
```

```
numbers x and y
cor(x, y)

## [1] 0.9954374

#we use set.seed() for rnorm() to produce the exact same set of random
numbers
set.seed(1303)
rnorm(50)

##  [1] -1.1439763145  1.3421293656  2.1853904757  0.5363925179  0.0631929665
##  [6]  0.5022344825 -0.0004167247  0.5658198405 -0.5725226890 -1.1102250073
## [11] -0.0486871234 -0.6956562176  0.8289174803  0.2066528551 -0.2356745091
## [16] -0.5563104914 -0.3647543571  0.8623550343 -0.6307715354  0.3136021252
## [21] -0.9314953177  0.8238676185  0.5233707021  0.7069214120  0.4202043256
## [26] -0.2690521547 -1.5103172999 -0.6902124766 -0.1434719524 -1.0135274099
## [31]  1.5732737361  0.0127465055  0.8726470499  0.4220661905 -0.0188157917
## [36]  2.6157489689 -0.6931401748 -0.2663217810 -0.7206364412  1.3677342065
## [41]  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) #mean() function to compute the mean of a vector of numbers

## [1] 0.01103557

var(y) #var() function to compute the variance of a vector of numbers

## [1] 0.7328675

sqrt(var(y)) #Applying sqrt() to the var() to get the standard deviation

## [1] 0.8560768

sd(y) #sd() is another way to get the standard deviation of a vector of
numbers

## [1] 0.8560768
```

#2.3.2 Graphics

```
#To generate a scatter plot for random numbers.
#The plot() function is the primary way to plot Data in R
x <- rnorm(100)
y <- rnorm(100)
plot(x, y)
```




```r
plot(x, y, xlab = "this is the x-axis",
     ylab = "this is the y-axis",
     main = "Plot of X vs Y")
```

```r
#We can save the output of an R plot. In this case, the pdf() function was
used to save the plot as a pdf.
pdf("Figure.pdf")
plot(x, y, col = "green")
dev.off()
```

```
## png
##   2
```

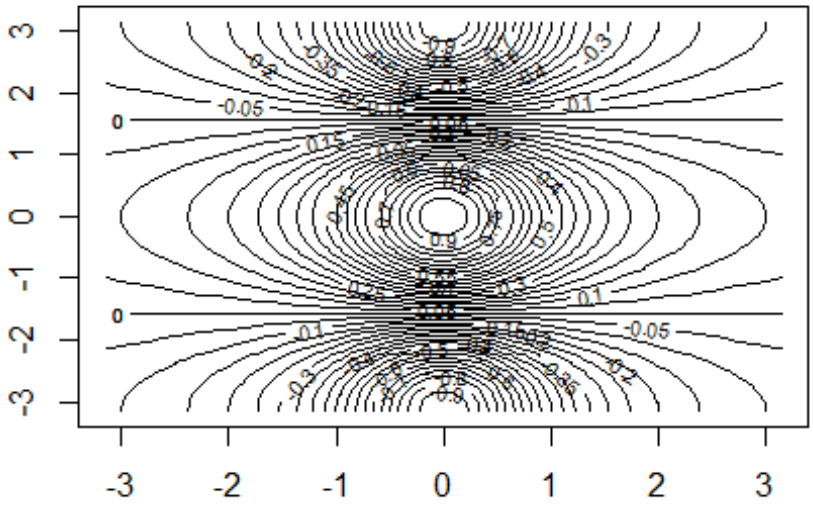
```r
#seq() function to create a sequence of numbers. Here this command makes a
vector of integers between 1 and 10

x <- seq(1, 10)
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

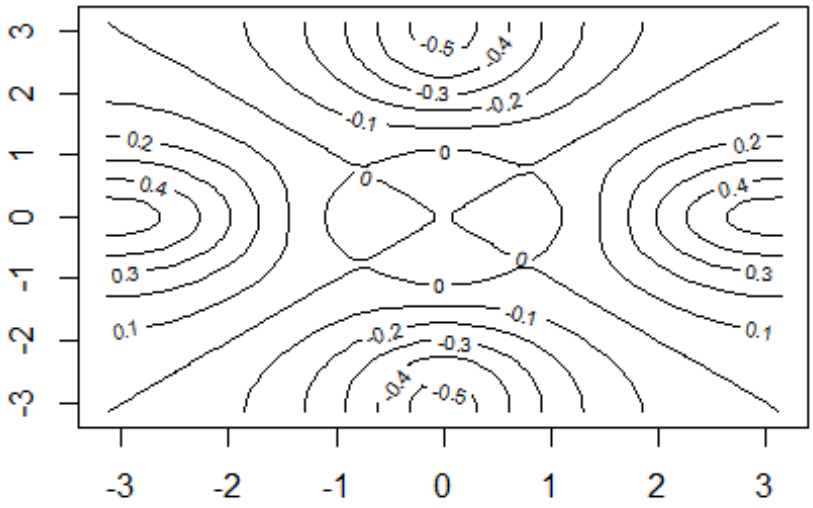
```r
x <- 1:10
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

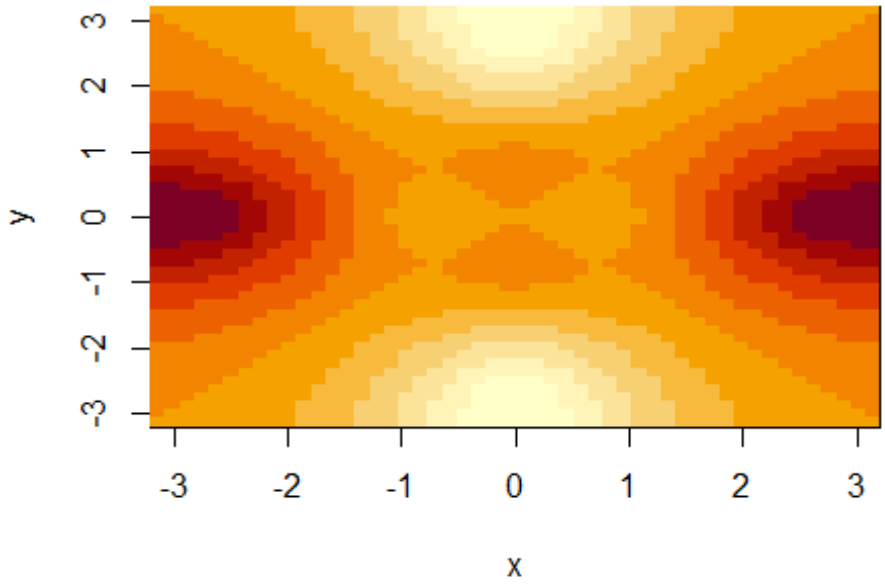
```r
#Another way to use the seq() function
x <- seq(-pi, pi, length = 50)

#contour() function to produce a contour plot in order to represent three-
dimensional data, like a topographical map

#first argument: A vector of the x values, which is the first dimension

#second argument: A vector of the y values, which is the second dimension

#Third argument: A matric whose elements correspond to the z value, the third
dimension, for each pair of (x, y) coordinates

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)
```



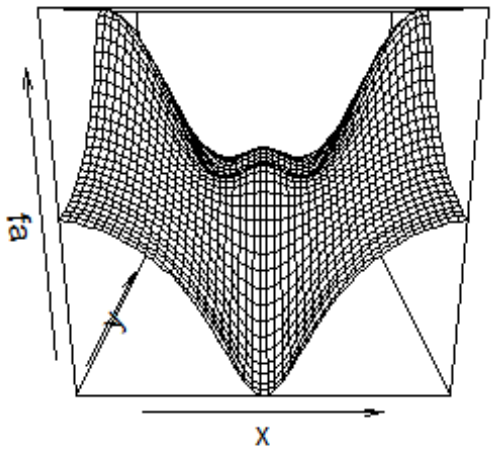
```
fa <- (f - t(f)) / 2
contour(x, y, fa, nlevels = 15)
```



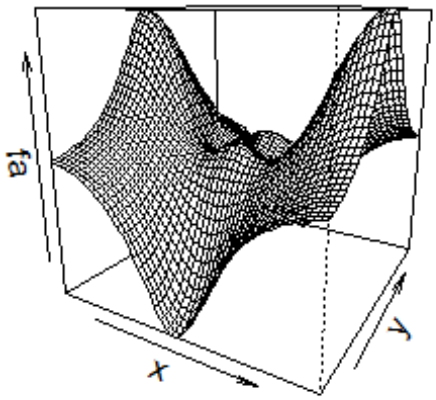
```r
#image() function creates a color-coded plot, called heatmap, whose colors
depend on the z value
image(x, y, fa)
```



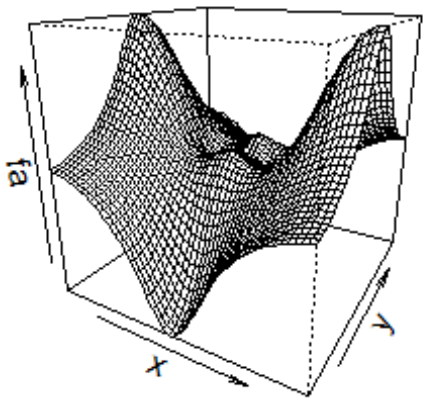

```r
#persp() function also produces a three-dimensional plot. The angle of
viewing the plot depends upon its arguments theta and phi
persp(x, y, fa)
```



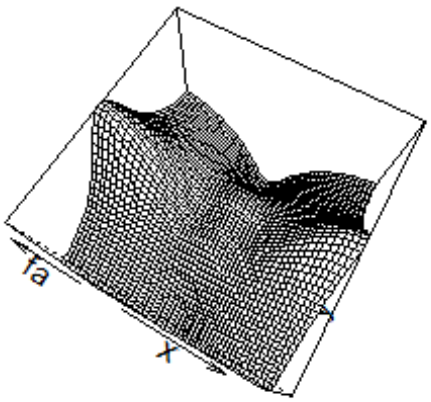

```
persp(x, y, fa, theta = 30)
```



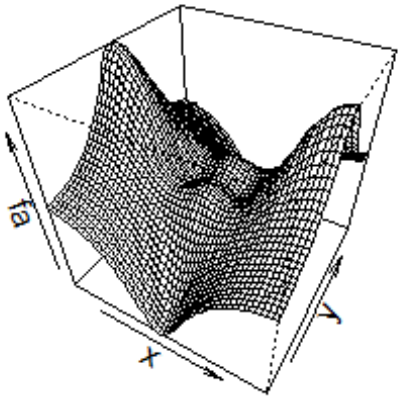

```
persp(x, y, fa, theta = 30, phi = 20)
```




```
persp(x, y, fa, theta = 30, phi = 70)
```




```
persp(x, y, fa, theta = 30, phi = 40)
```



#2.3.3 Indexing Data

```
#Creating a 4 x 4 matrix with numbers 1 to 16 as elements
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
```

```
#This will select the element from the A matrix corresponding to second row
and third column
A[2, 3]
```

```
## [1] 10
```

```
#Multiple rows and columns at a time can be selected as well

#To select elements from rows 1 and 3 in columns 2 and 4
A[c(1, 3), c(2, 4)]
```

```
##      [,1] [,2]
## [1,]    5   13
## [2,]    7   15
```

```
#To select elements from rows 1 to 3 in columnds 2 to 4
A[1:3, 2:4]
```

```
##      [,1] [,2] [,3]
## [1,]    5    9   13
## [2,]    6   10   14
## [3,]    7   11   15
```

```
#To select and display the elements row-wise from 1 to 2
A[1:2, ]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
```

```
#To select and display the elements column-wise from 1 to 2
A[, 1:2]
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
```

```
#Displaying single row, [1] of matrix A
A[1, ]
```

```
## [1]  1  5  9 13
```

```
#Using negative sign in the index to keep all rows or columns except those
indicated in the index
A[-c(1, 3), ]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    6   10   14
## [2,]    4    8   12   16
```

```
#dim() function to output th enumber of rows, followed by the number of
columns in a given matrix.
dim(A)
```

```
## [1] 4 4
```

#2.3.4 Loading Data

```
#read.table() function to load data from a text file which has been
downloaded from the textbook website to the working directory
Auto <- read.table("C:/Users/vddee/Documents/Auto.data")


View(Auto) #View() function to view it in a spreadsheet like window
head(Auto) #head() function to view the first few rows of the data
```

```
##     V1        V2           V3         V4     V5           V6   V7     V8
## 1  mpg cylinders displacement horsepower weight acceleration year origin
## 2 18.0         8        307.0      130.0  3504.         12.0   70      1
## 3 15.0         8        350.0      165.0  3693.         11.5   70      1
## 4 18.0         8        318.0      150.0  3436.         11.0   70      1
## 5 16.0         8        304.0      150.0  3433.         12.0   70      1
## 6 17.0         8        302.0      140.0  3449.         10.5   70      1
##                         V9
## 1                     name
## 2 chevrolet chevelle malibu
## 3         buick skylark 320
## 4        plymouth satellite
## 5             amc rebel sst
## 6               ford torino
```

```
#This is to deal with the missing values in datasets. Using the argument
header = T, tells R that the first line of the file contains the variable
names and using the option na.strings tells R that any time it sees a
particular character (like "?") or a set of characters, it should be treated
as a missing element.

Auto <- read.table("C:/Users/vddee/Documents/Auto.data", header = T,
na.strings = "?", stringsAsFactors = T)
View(Auto)

#Here dim() function tells us that the data has 397 observations and 9
variables, or columns
dim(Auto)
```

```
## [1] 397   9
```

```
Auto[1:4, ]
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
##                         name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
```

```
#To remove the rows that contain missing observations
Auto <- na.omit(Auto)
dim(Auto)
```

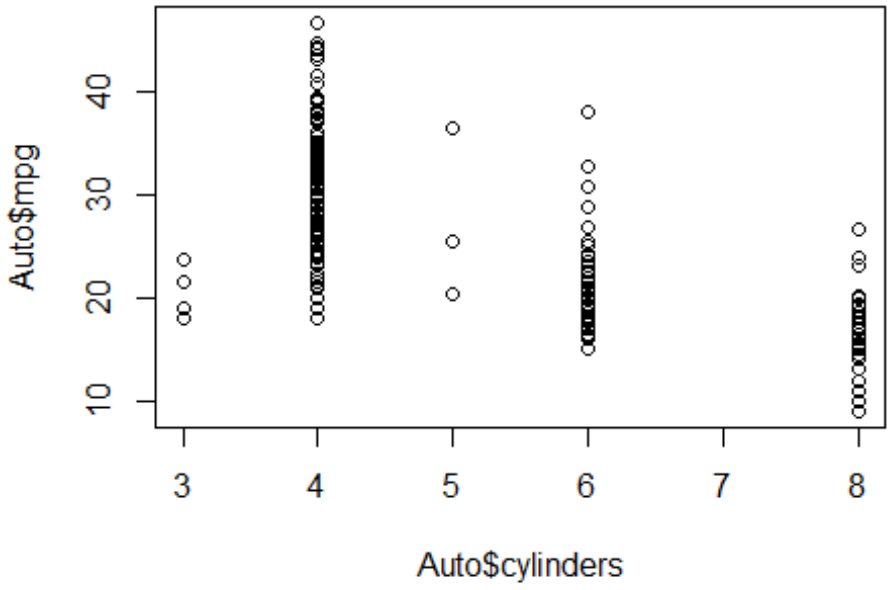
```
## [1] 392   9
```

```
#To check variable names after the data has been loaded correctly
names(Auto)
```

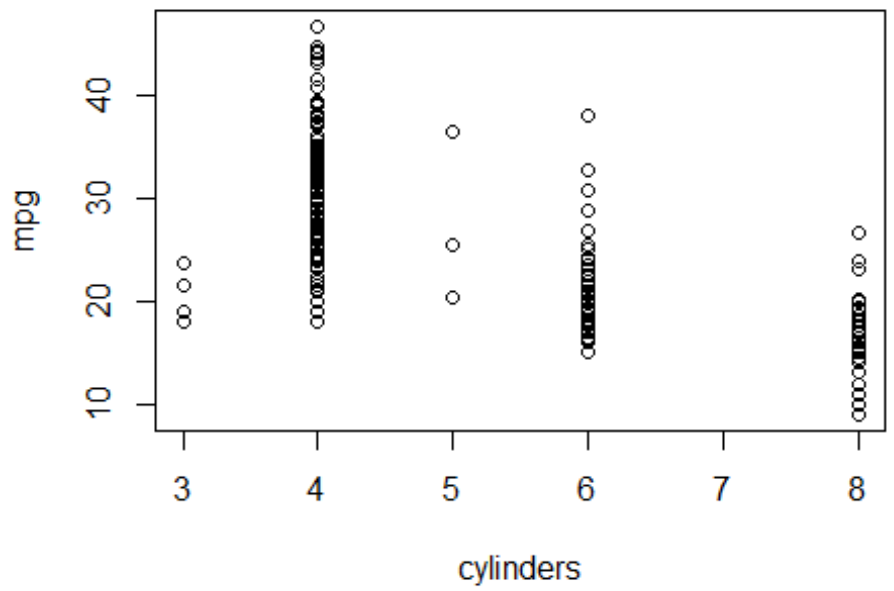
```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

#2.3.5 Additional Graphical and NUmerical Summaries

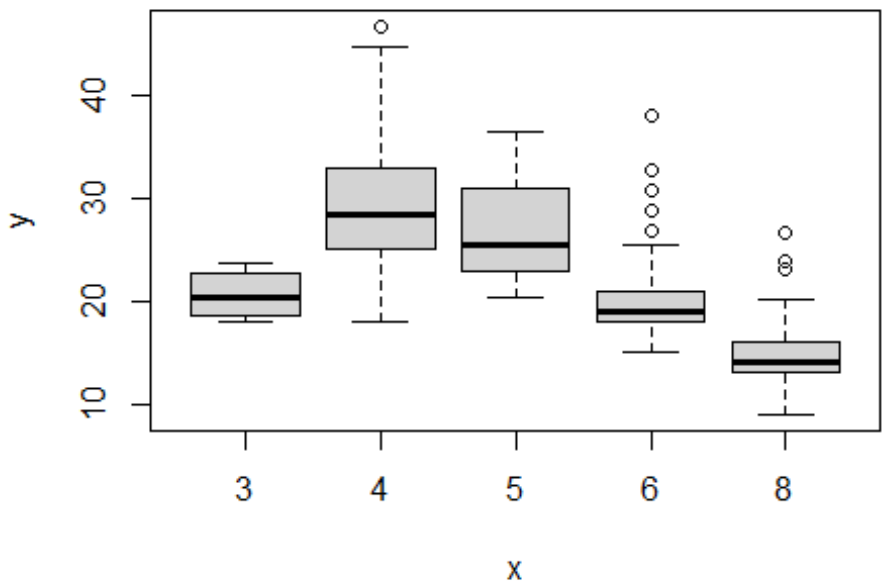
```
#As per the textbook, this is a wrong format as we have to mention the
dataset and then the column as shown in the next step. But could be a new
update in the R-version that is making this format work as well
#plot(cylinders, mpg)

#To refer to a variable the format should be dataset$variablename as shown
plot(Auto$cylinders, Auto$mpg)
```



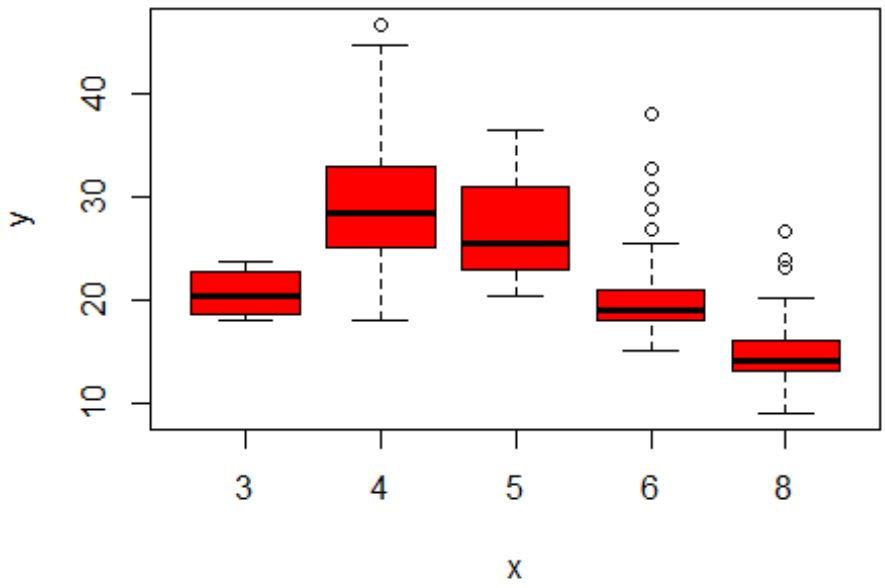

```
#alternatively, attach() function can be used in order to
attach(Auto)
plot(cylinders, mpg)
```



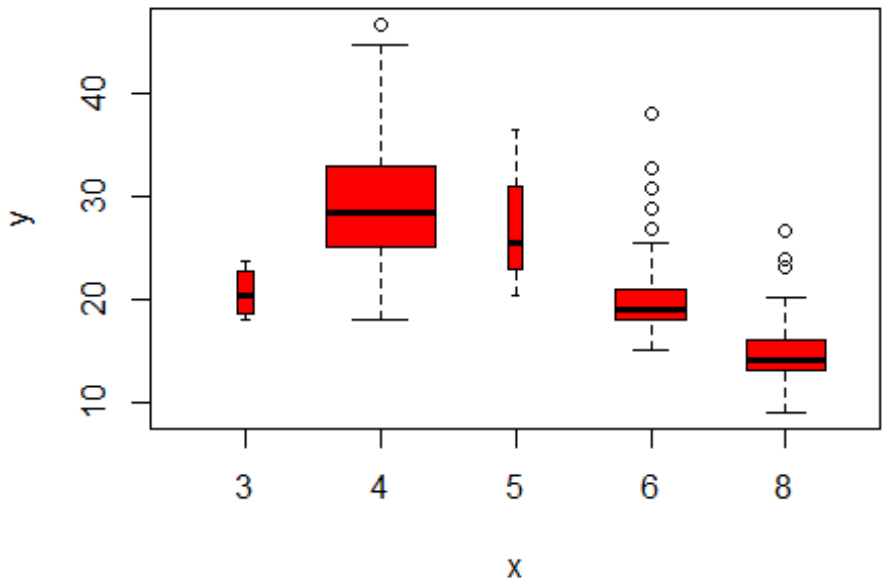

```r
#the as.factor() function converts quantitative variables into qualitative variables
cylinders <- as.factor(cylinders)

#As it can be observed, the output this time is a boxplot. The boxplots will be produced by the plot() function when the variable plotted on x-axis is qualitative
plot(cylinders, mpg)
```



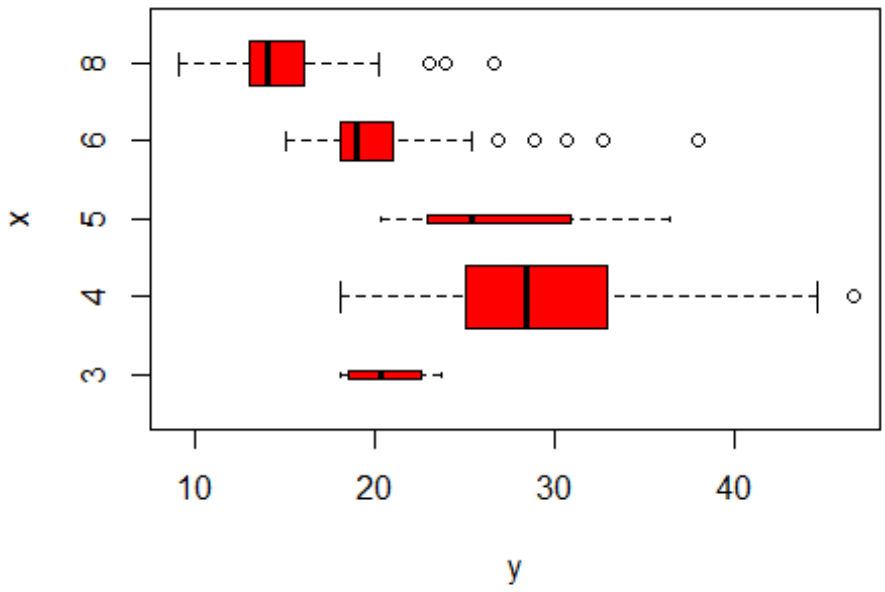

```
plot(cylinders, mpg, col = "red")
```



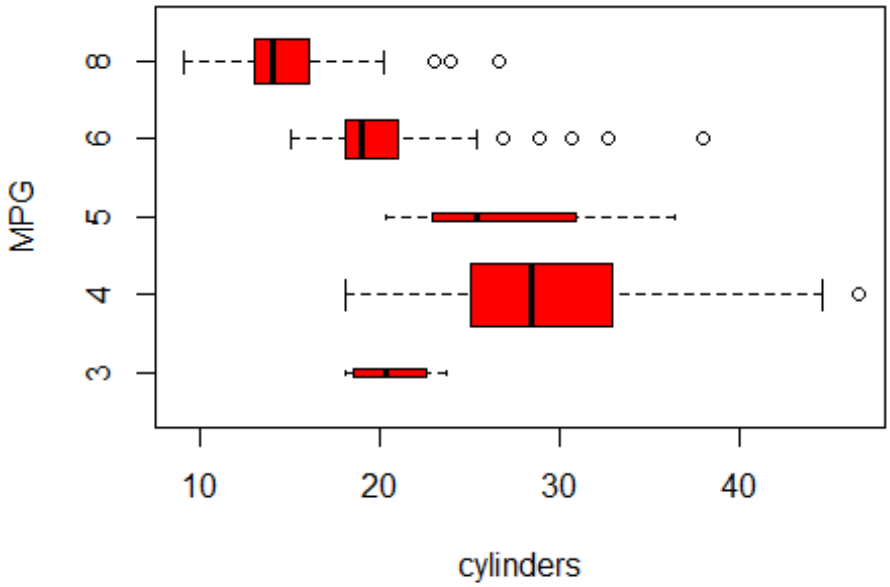

```
plot(cylinders, mpg, col = "red", varwidth = T)
```



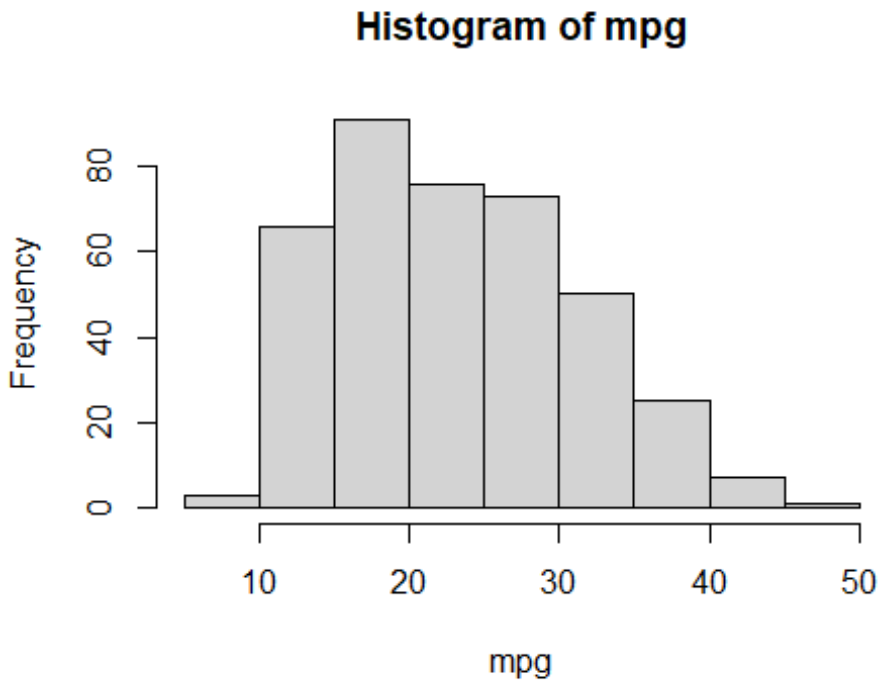

```r
plot(cylinders, mpg, col = "red", varwidth = T, horizontal = T)
```



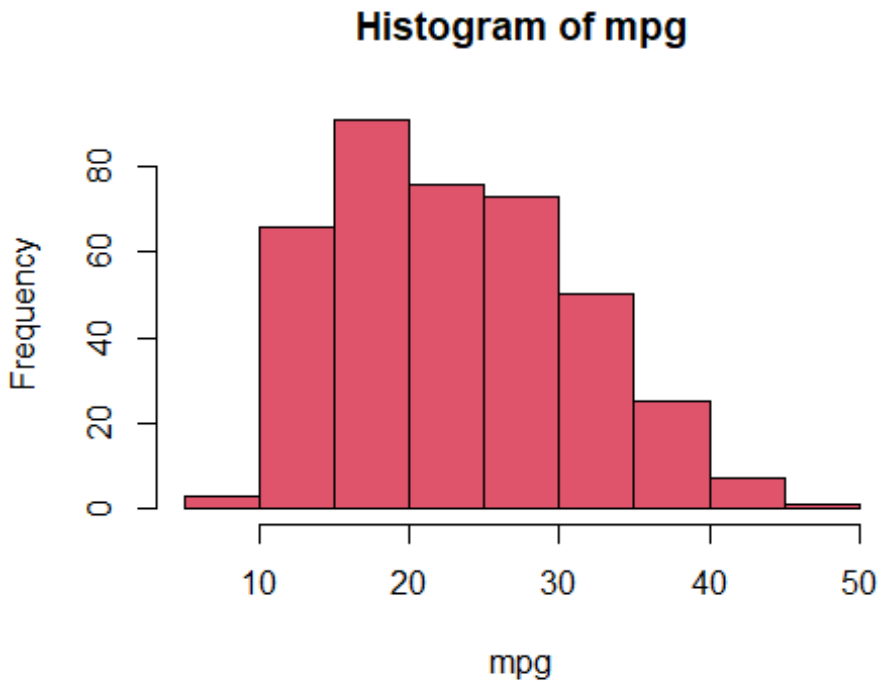
```r
plot(cylinders, mpg, col = "red", varwidth = T, horizontal = T, xlab =
"cylinders", ylab = "MPG")
```



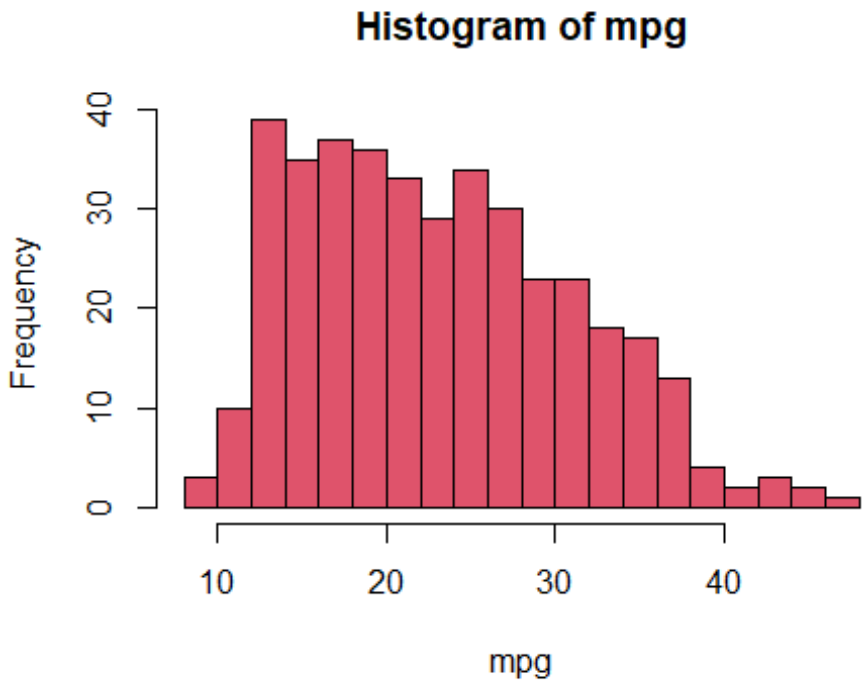

```r
#hist() function to plot a Histogram
hist(mpg)
```



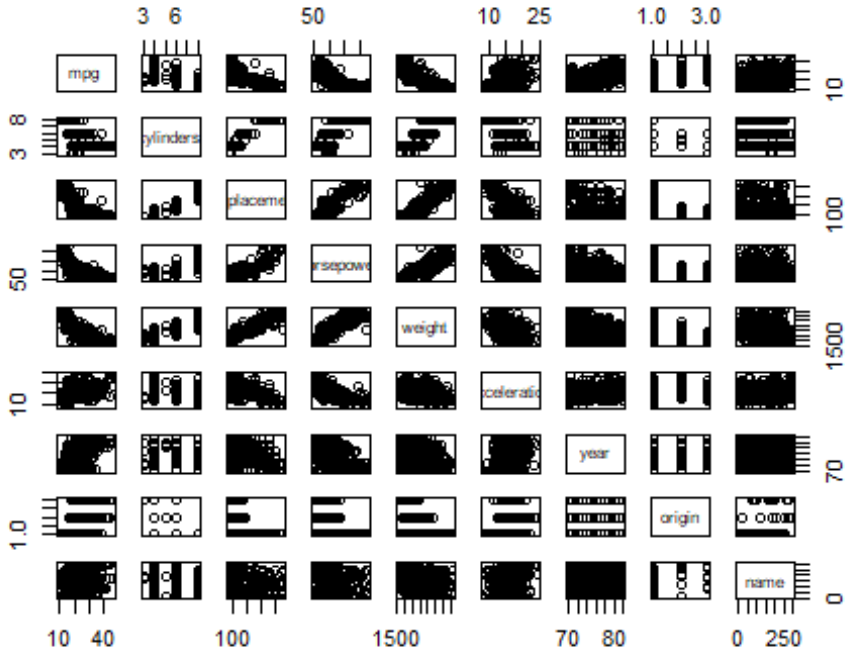

```r
hist(mpg, col = 2) #col =2 has the same effect as col = "red"
```



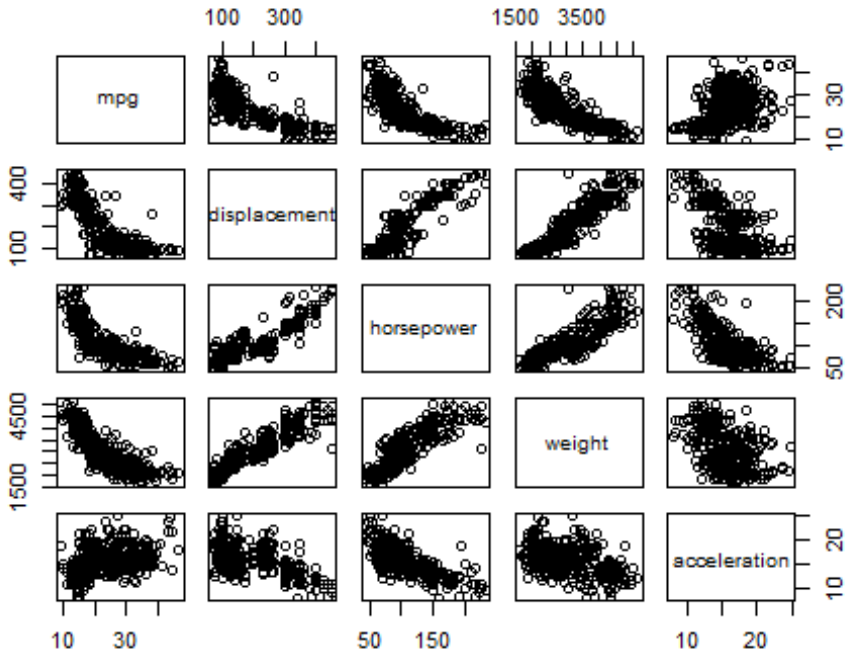

```r
hist(mpg, col = 2, breaks = 15)
```



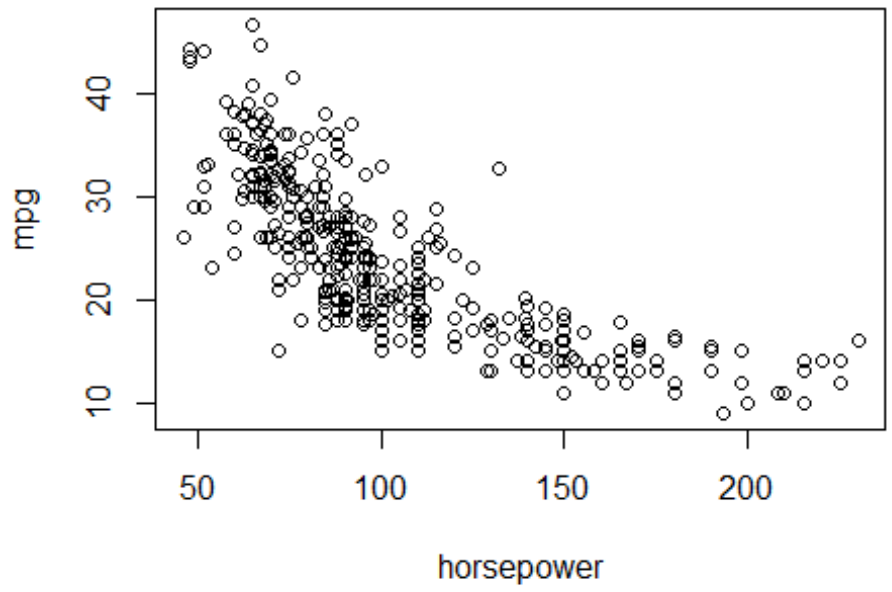

```
#pairs() function to create a Scatterplot Matrix, meaning, a scatterplot for
every pair of variables or just a subset of variables.

pairs(Auto)
```




```r
pairs(~mpg + displacement + horsepower + weight + acceleration, data = Auto)
```



```
#the identify() function in conjunction with the plot() function, provides a
useful interative method for identifying the value of a particular variable
for points on a plot
plot(horsepower, mpg)
identify(horsepower, mpg, name)
```




```
## integer(0)

#summary() function to get a numerical summary of each variable in a
particular data set

summary(Auto)

##       mpg          cylinders      displacement     horsepower
weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.
:1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st
Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median
:2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean
:2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd
Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.
:5140
```

```
##
##   acceleration        year           origin                      name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5
##  Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5
##  Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet        :  4
##  Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4
##                                                  (Other)           :365
```

```
#using summary() function on qualitative variable, will list the number of
observations that fall in each category. A summary of single variable can
also be produced
```

```
summary(mpg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.00   17.00   22.75   23.45   29.00   46.60
```