ISLR Lab 2

Suarez Rodes, Randy

Basic Commands

Vector creation c() "concatenate" function. Use <- or = to declare variables or functions.

```
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)
```

Adding vectors pointwise.

```
length(x)
```

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

```
length(y)
```

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

```
x+y
```

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

ls() and rm() (List of objects and remove)

```
ls()
```

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

```
rm(x,y)
ls()
```

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

Removing all at once

```
rm(list=ls())
```

The matrix function ?matrix to learn more about it (? it is used to get documentation help) For matrix specify data, number of rows and cols.

```
x=matrix (data=c(1,2,3,4) , nrow=2, ncol =2)
x
```

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

We can specify the order of the entries using byrow, by default byrow=FALSE, hence the entries are created by column as the default setting.

```
x=matrix (data=c(1,2,3,4) , nrow=2, ncol =2,byrow = TRUE)
x
```

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

Functions work in a vectorized fashion (entry wise)

```
sqrt(x) #square root
```

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

```
x^2 #power
```

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

Random Data generation Ex:rnorm() Normal random variates. Random generation changes each call (use set.seed() to fix a random generator seed) cor() correlation

```
x=rnorm (50)
y=x+rnorm (50, mean=50, sd=.1)
cor(x,y)
```

```
## [1] 0.9907775
```

```
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
```

mean() and var() of a vector, apply sqrt() to the output of var() to obtain standard deviation or simply use sd()

```
set.seed(3)
y=rnorm (100)
mean(y)
```

```
## [1] 0.01103557
```

```
var(y)
```

```
## [1] 0.7328675
```

```
sqrt(var(y))
```

```
## [1] 0.8560768
```

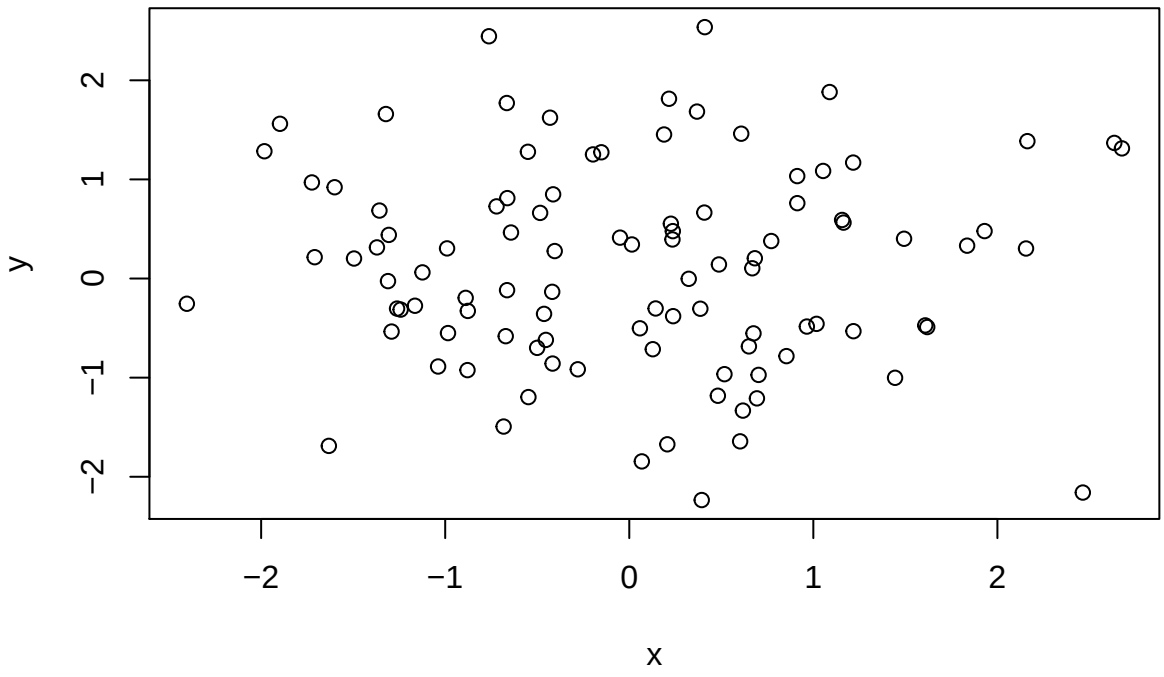
```
sd(y)
```

```
## [1] 0.8560768
```

Graphics

plot() function. Example plot(x,y) produces a scatterplot of the numbers in x versus the numbers in y.

```
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")
```

To create a pdf, we use the pdf() function, and to create a jpeg, pdf() we use the jpeg() function

pdf (" Figure .pdf “) plot(x,y,col =” green ") dev.off () null device

The function dev.off() indicates to R that we are done creating the plot.

The function seq() can be used to create a sequence of numbers. For instance, seq(a,b) makes a vector of integers between a and b. There are many other options: for instance, seq(0,1,length=10) makes a sequence of 10 numbers that are equally spaced between 0 and 1. Typing 3:11 is a shorthand for seq(3,11) for integer arguments.

```
x=seq (1 ,10)
x
```

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

```
x=1:10
x
```

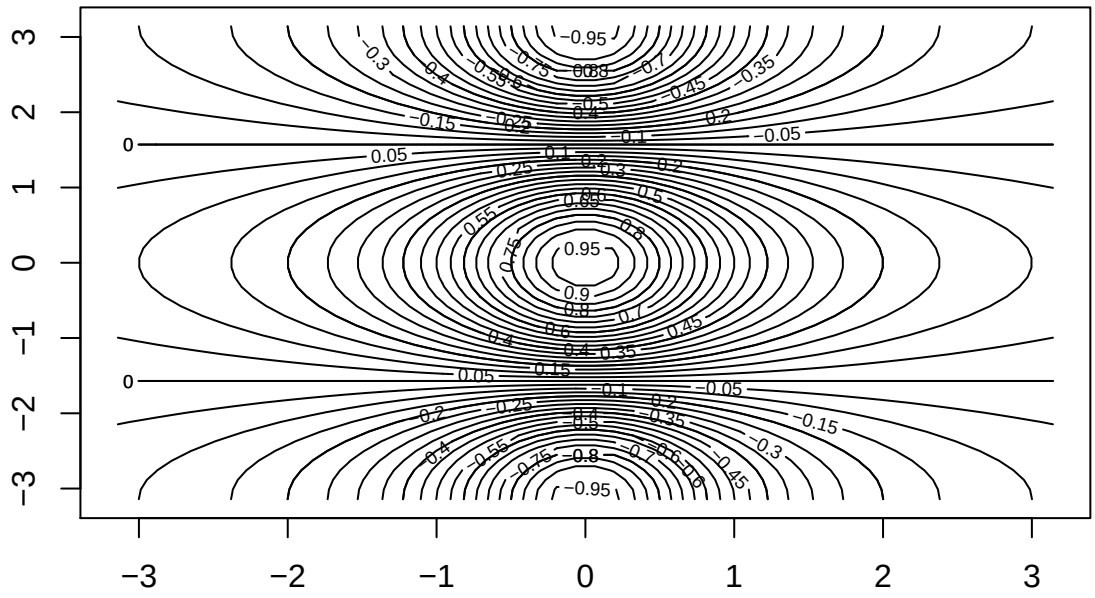
```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
x=seq(-pi ,pi ,length =50)
```

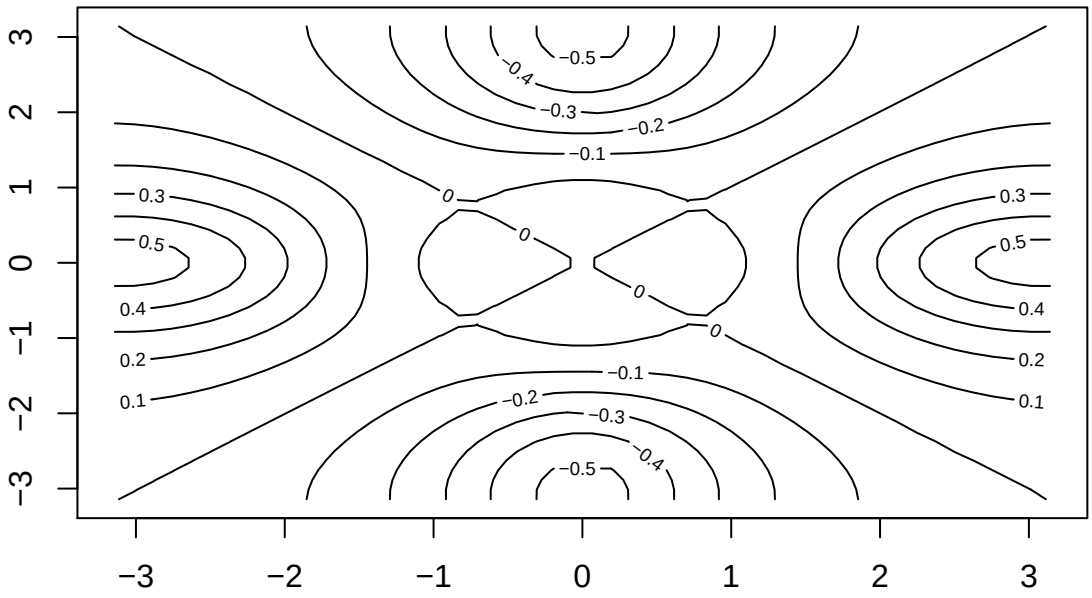
Function contour() produces a contour plot in order to represent three-dimensional data. it is like a topographical map. It takes three arguments:

1. A vector of the x values (the first dimension),
2. A vector of the y values (the second dimension), and
3. A matrix 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)) #outer product
contour (x,y,f)
contour (x,y,f,nlevels =45, add=T)
```

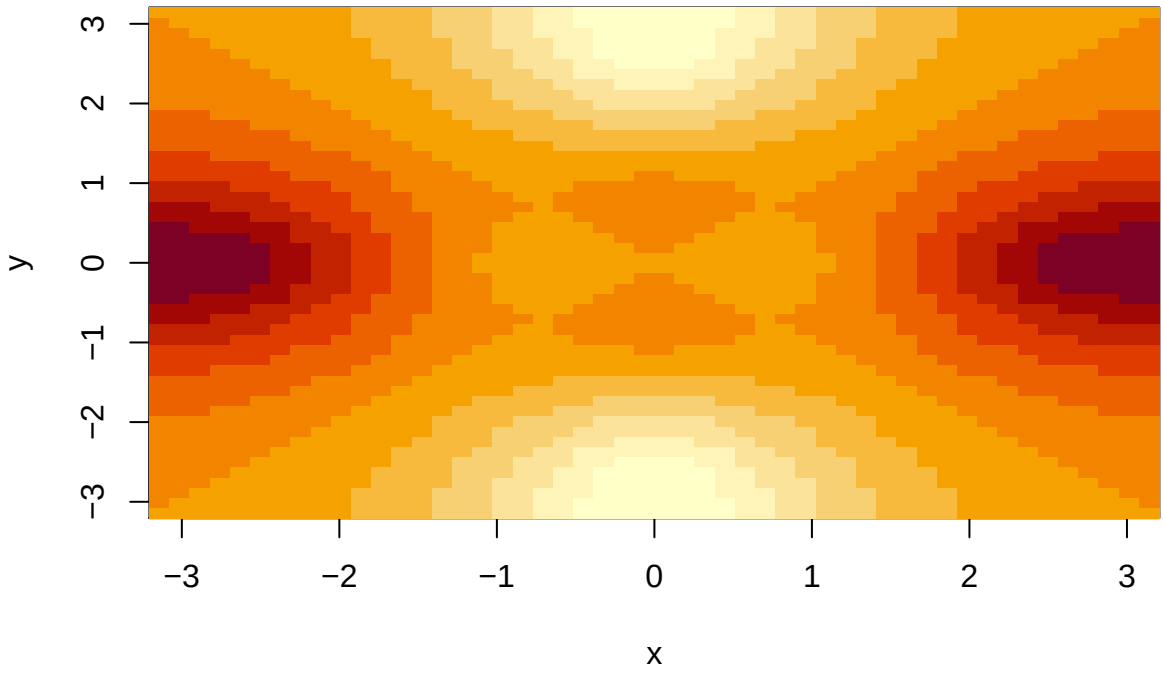



```
fa=(f-t(f))/2 #t() for transpose
contour (x,y,fa,nlevels =15)
```

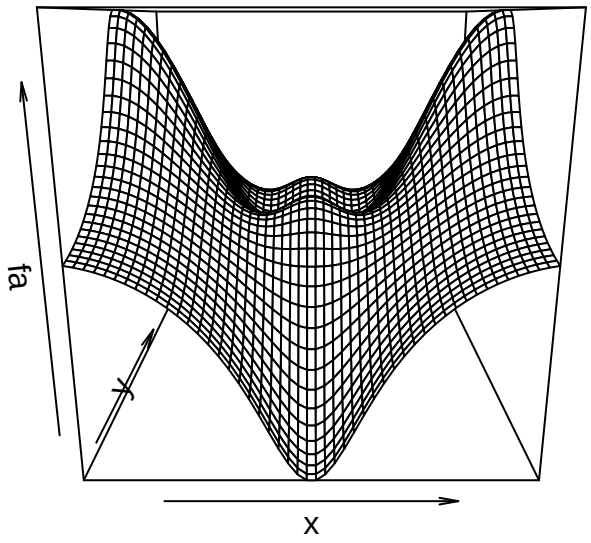


The image() function works the same way as contour(), except that it image() produces a color-coded plot whose colors depend on the z value. This is known as a heatmap, and is sometimes used to plot temperature in weather forecasts. Alternatively, persp() can be used to produce a three-dimensional persp() plot. The arguments theta and phi control the angles at which the plot is viewed.

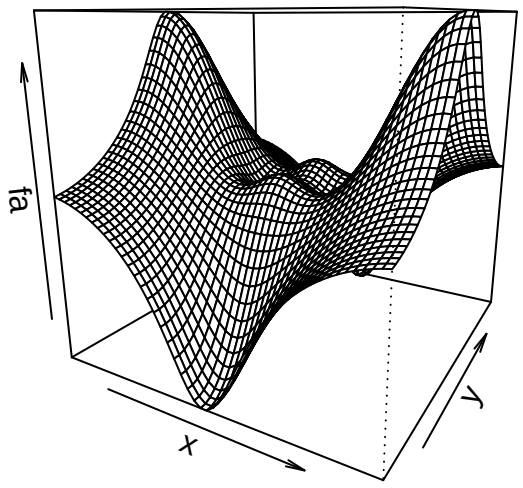
```
image(x,y,fa)
```



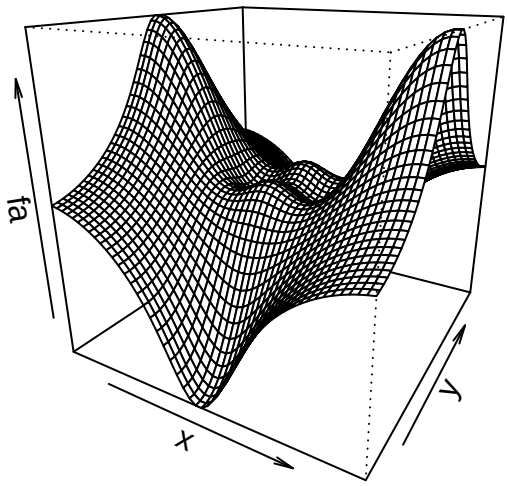

```
persp(x,y,fa)
```



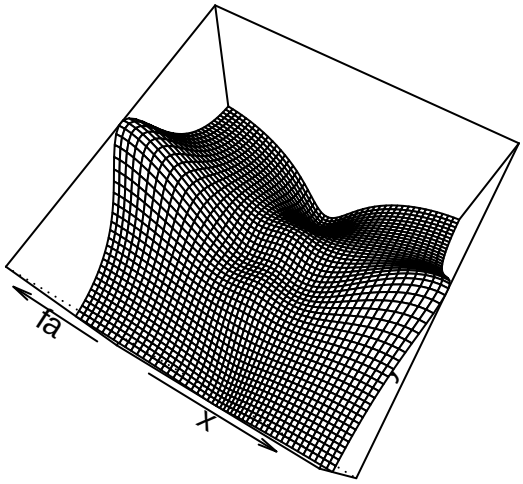

```
persp(x,y,fa ,theta =30)
```



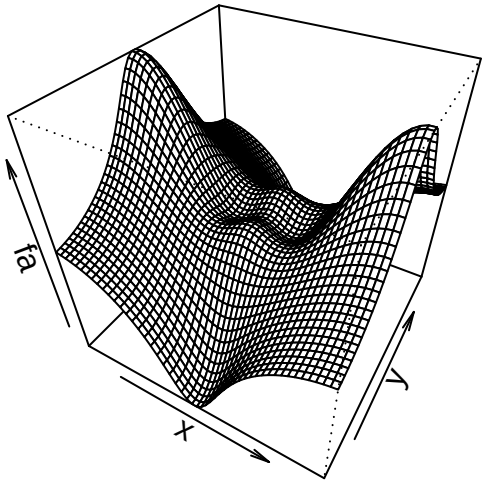

```
persp(x,y,fa ,theta =30, phi =20)
```




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




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



Indexing Data

```
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
```

Indexing and Slicing

```
A[2,3] #2nd row 3rd column element, remember R indexes start at 1.
```

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

```
A[c(1,3) ,c(2,4) ] #Submatrix of rows 1 and 3 columns 2 and 4
```

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

```
A[1:3 ,2:4] #Submatrix from row 1 through 3 and columns from 2 through 4
```

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

```
A[1:2,] #Rows 1 through 2 and all columns
```

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

```
A[,1:2] #All rows and columns 1 through 2
```

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

```
A[1,] #Single row or column is treated as a vector object
```

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

```
A[-c(1,3),]#- for indexes indicates complement, here all rows but row 1 and 3
```

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

```
dim(A) #number of rows by columns
```

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

Loading Data

To import data we can use read.table() (data frame), and to export we can use write.table()

```
Auto=read.table("Auto.data") #loading the Auto data from the current directory, no necessary if we load
fix(Auto) #use to visualize the data but must be closed to continue working with R
```

This particular data set has not been loaded correctly, because R has assumed that the variable names are part of the data and so has included them in the first row. The data set also includes a number of missing observations, indicated by a question mark ?. Missing values are a common occurrence in real data sets. Using the option header=T (or header=TRUE) in the read.table() function 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 or set of characters (such as a question mark), it should be treated as a missing element of the data matrix.

```
Auto=read.table ("Auto.data", header =T,na.strings ="?")
fix(Auto)
```

Comma Separated Values

```
Auto=read.csv ("Auto.csv", header =T,na.strings ="?")
fix(Auto)
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
```

```
Auto=na.omit(Auto) #automatically omit rows with missing values
dim(Auto)
```

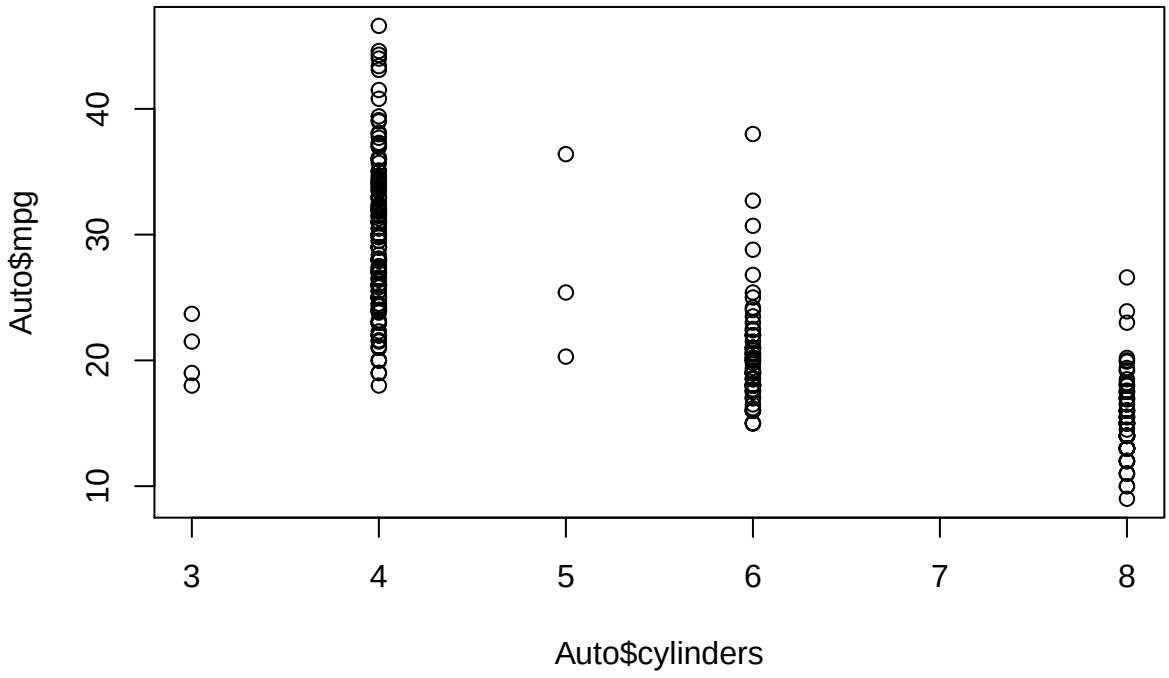
```
## [1] 392   9
```

```
names(Auto) #using names to check for the variable names
```

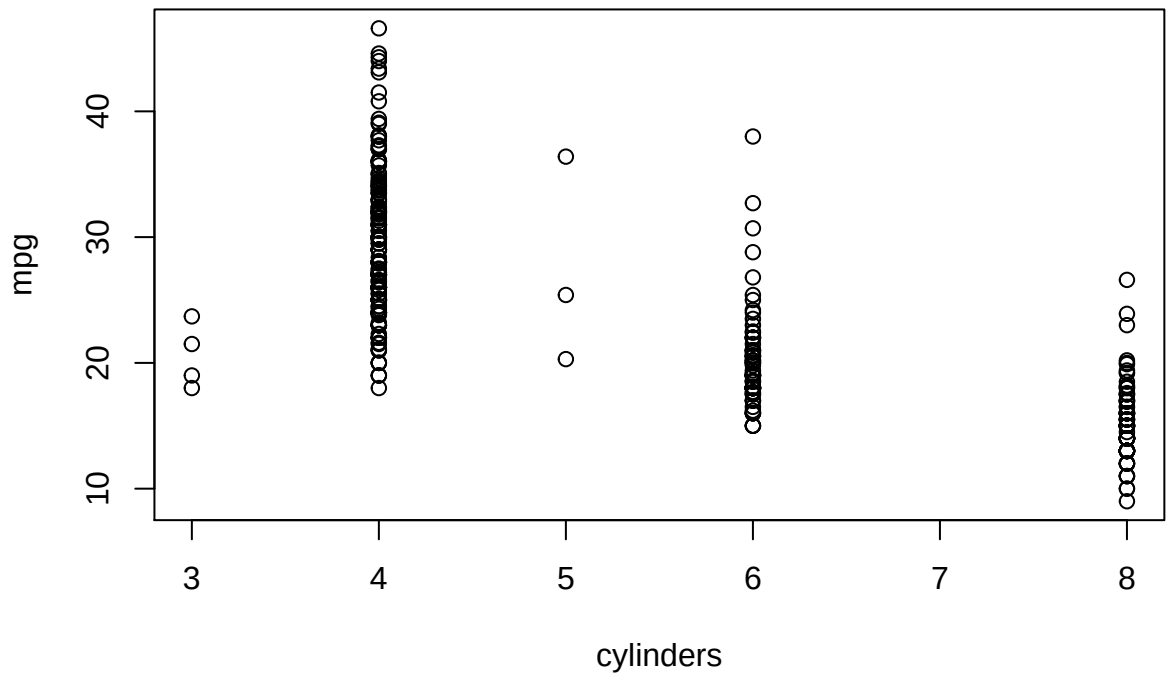
```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

Additional Graphical and Numerical Summaries

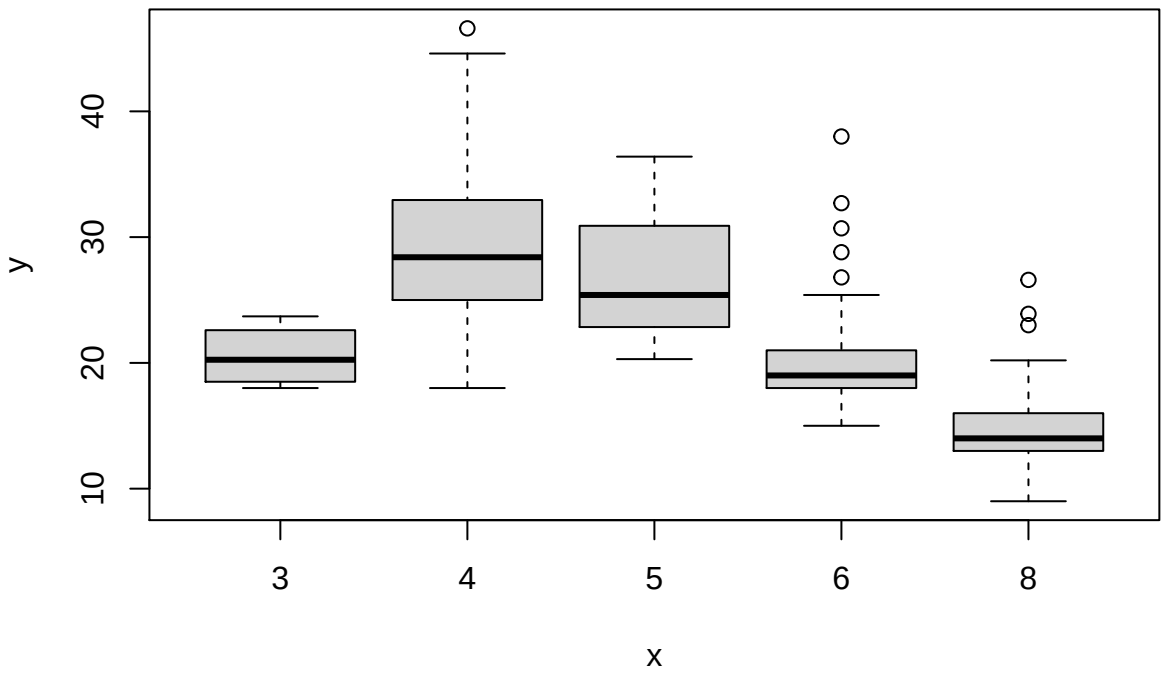
```
plot(Auto$cylinders , Auto$mpg ) #We use $ to access the variables of a data frame
```



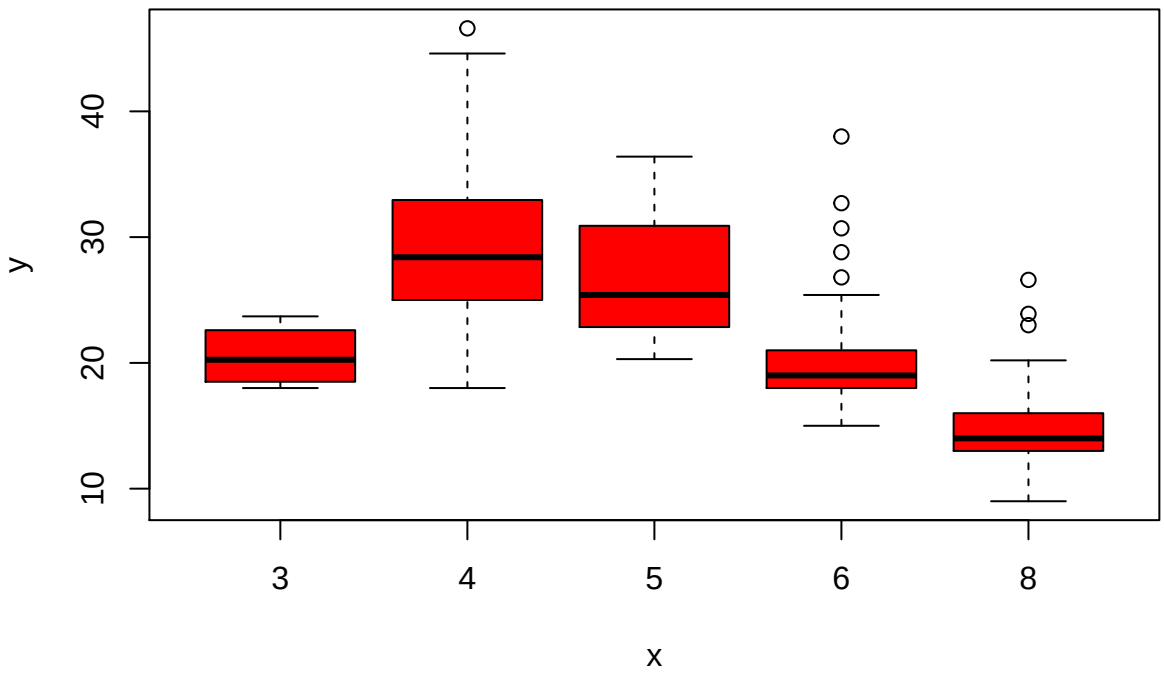

```
attach (Auto) #We can also use attach to make the variables in this data frame available by name
plot(cylinders , mpg)
```



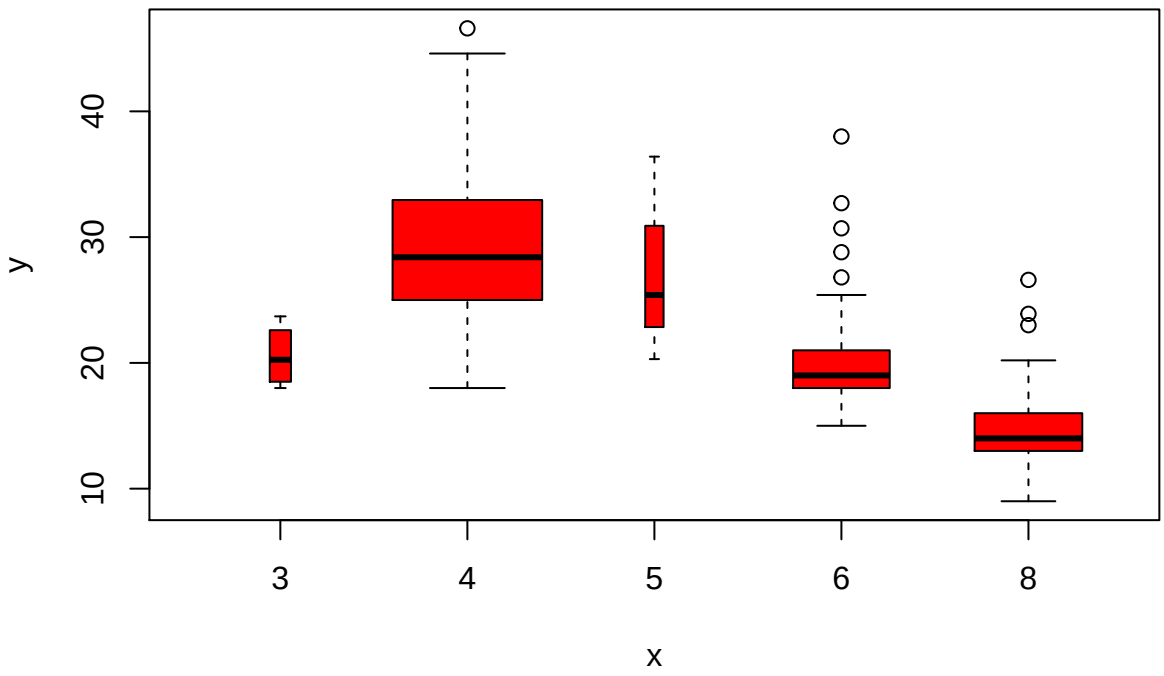

```
#We can change a cylinders to a qualitative variable using as.factor
cylinders =as.factor (cylinders)
plot(cylinders , mpg) #since cylinders is a categorical variable now, then we obtain a boxplot
```



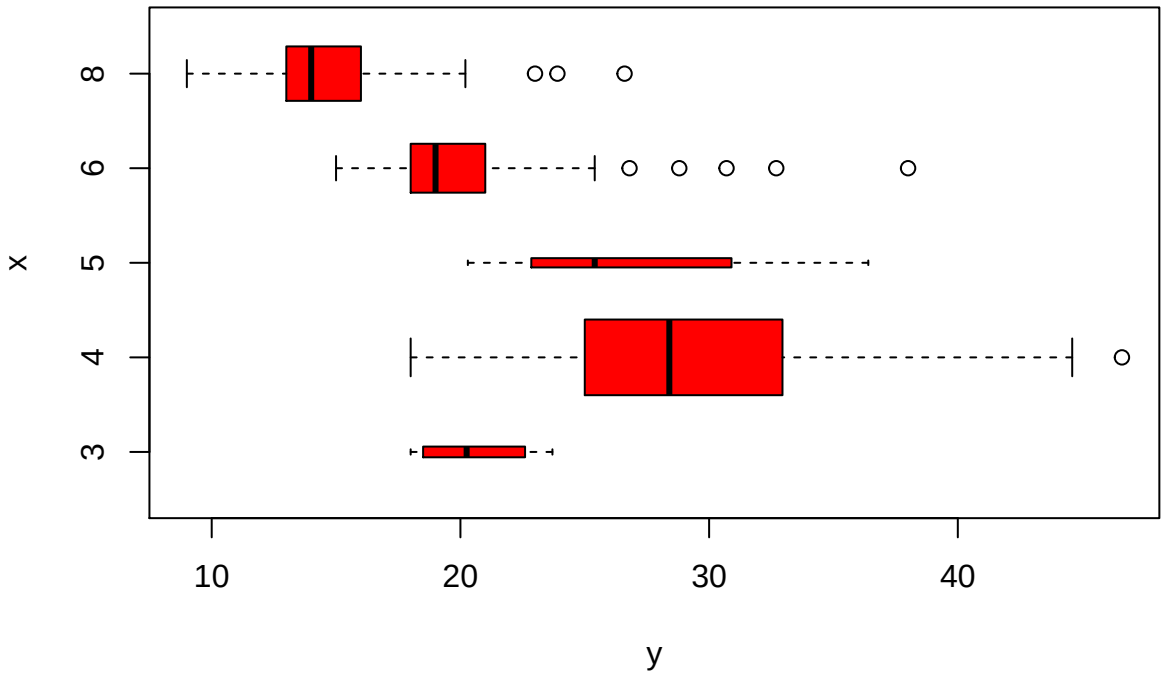

```
plot(cylinders , mpg , col ="red ")
```



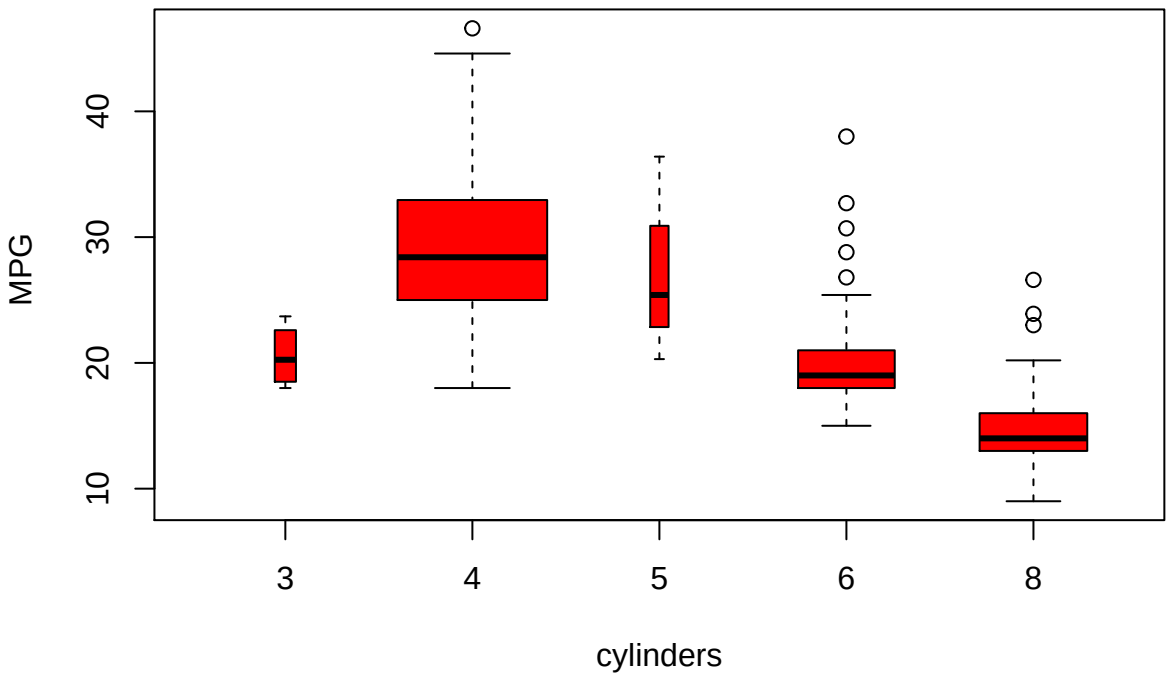

```
plot(cylinders , mpg , col ="red", varwidth =T)
```




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

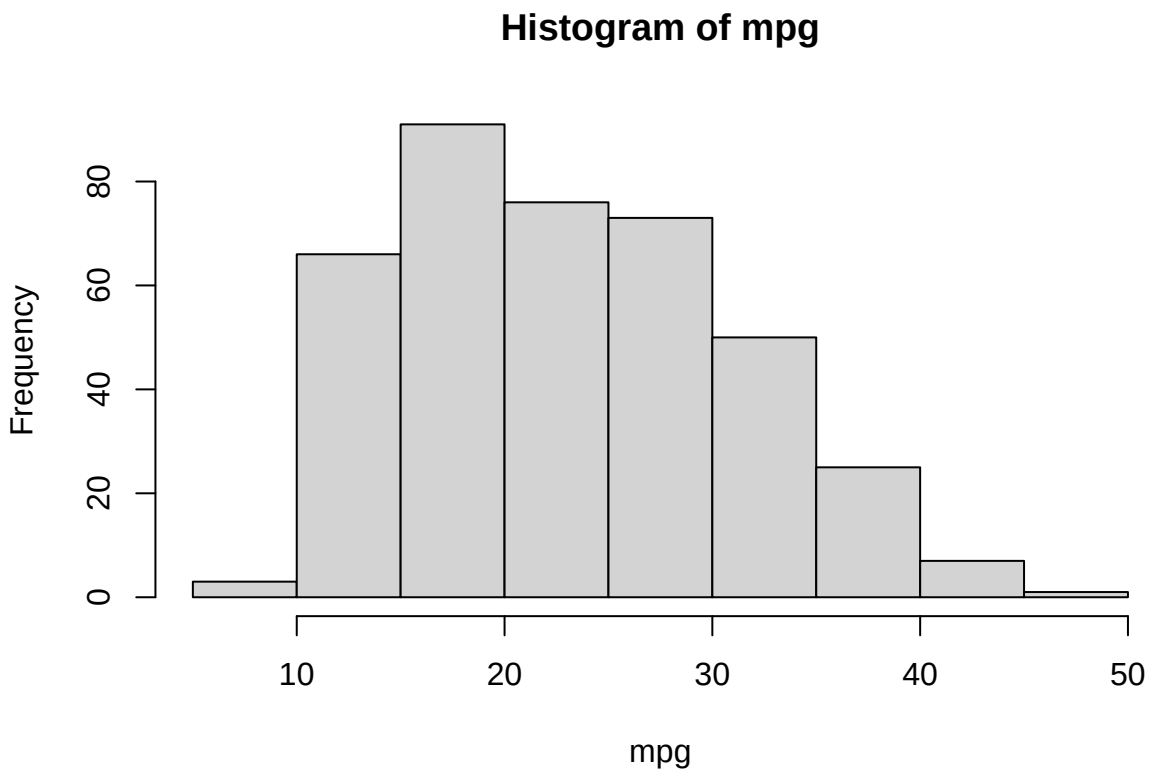



```
plot(cylinders , mpg , col ="red", varwidth =T, xlab=" cylinders ",ylab ="MPG ")
```

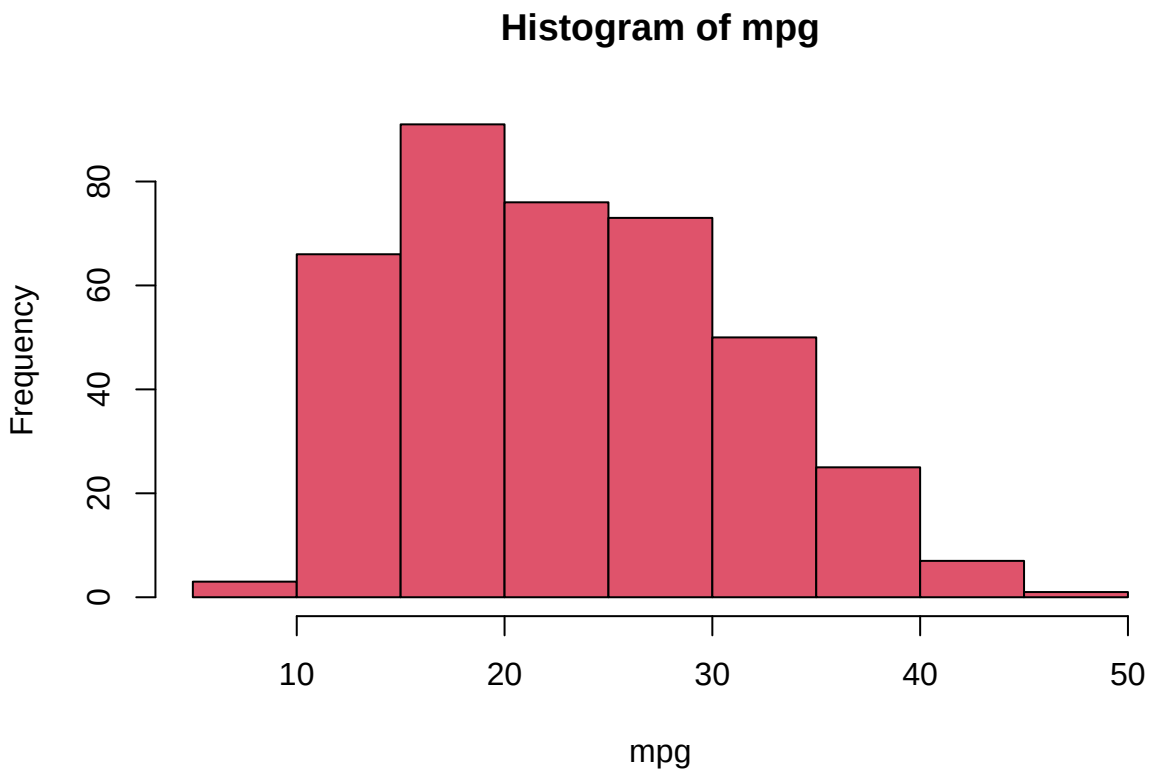


The hist() function can be used to plot a histogram. Note that col=2 has the same effect as col="red".

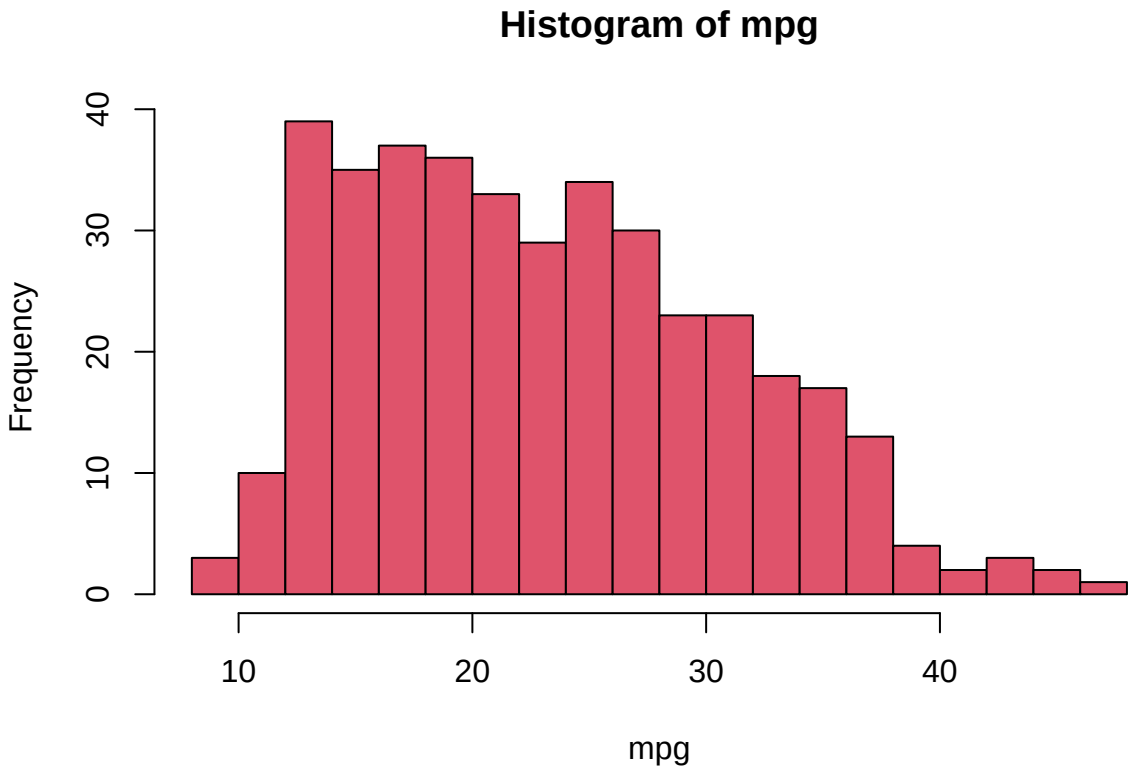
```
hist(mpg)
```




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

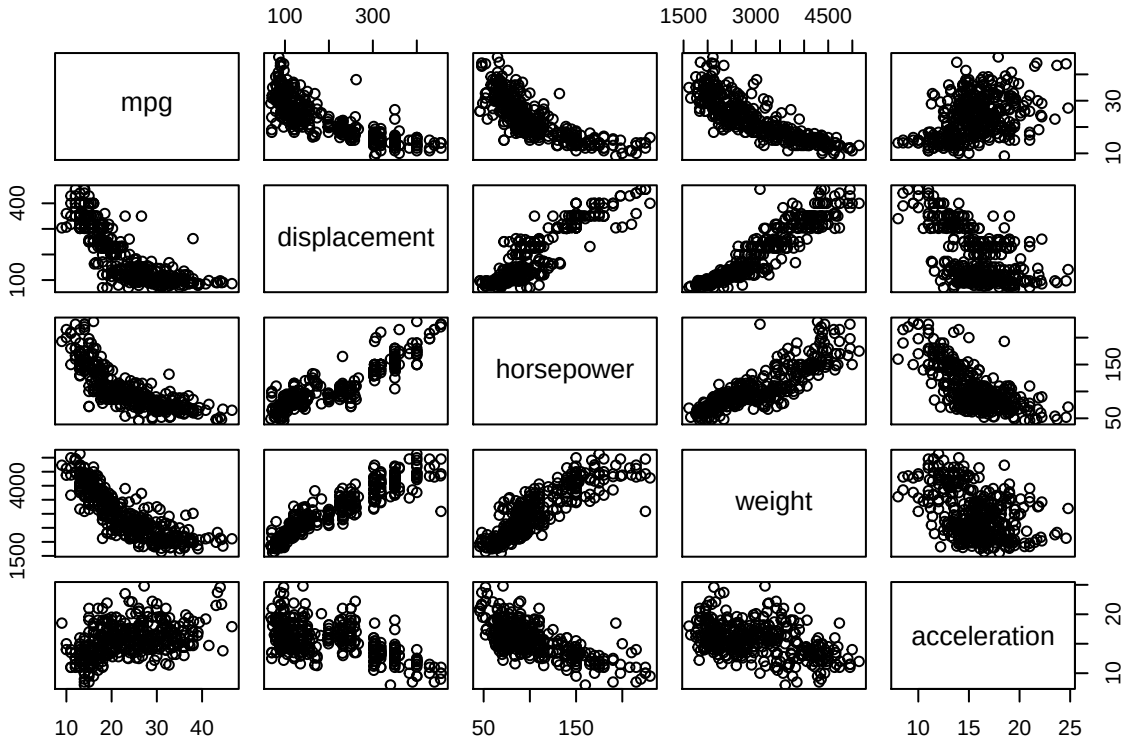



```
hist(mpg ,col =2, breaks =15)
```



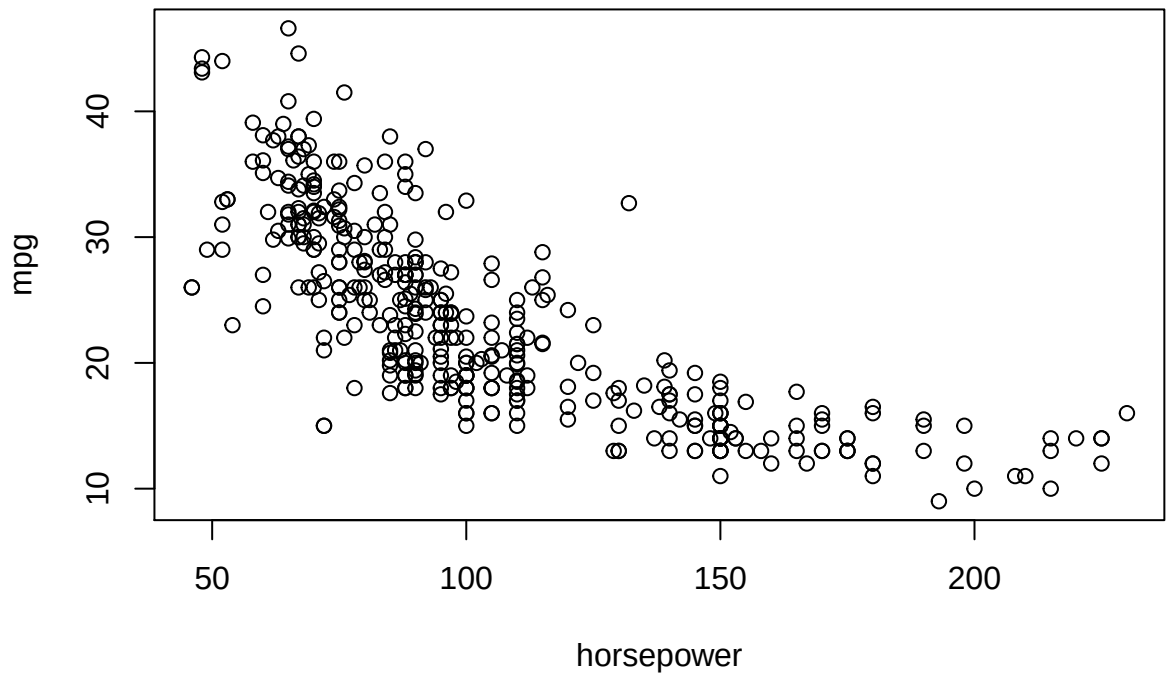
The pairs() function creates a scatterplot matrix i.e. a scatterplot for every pair of variables for any given data set. We can also produce scatterplots for just a subset of the variables.

```
#pairs(Auto) #Auto has the variable name as character, so this causes an error for pairs
pairs(~ mpg + displacement + horsepower + weight + acceleration , Auto)
```



In conjunction with the plot() function, identify() provides a useful identify() interactive method for identifying the value for a particular variable for points on a plot. We pass in three arguments to identify(): the x-axis variable, the y-axis variable, and the variable whose values we would like to see printed for each point. Then clicking on a given point in the plot will cause R to print the value of the variable of interest. Right-clicking on the plot will exit the identify() function (control-click on a Mac). The numbers printed under the identify() function correspond to the rows for the selected points.

```
plot(horsepower ,mpg)
identify (horsepower ,mpg ,name)
```




```
## integer(0)
```

The summary() function produces a numerical summary of each variable in summary() a particular data set. For qualitative variables such as name, R will list the number of observations that fall in each category. We can also produce a summary of just a single variable.

```
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   Length:392
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   Class :character
##  Median :15.50   Median :76.00   Median :1.000   Mode  :character
##  Mean   :15.54   Mean   :75.98   Mean   :1.577
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :24.80   Max.   :82.00   Max.   :3.000
```

```
summary(mpg)
```

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

Once we have finished using R, we type q() in order to shut it down, or quit. When exiting R, we have the

option to save the current workspace so that all objects (such as data sets) that we have created in this R session will be available next time. Before exiting R, we may want to save a record of all of the commands that we typed in the most recent session; this can be accomplished using the savehistory() function. Next time we enter R, we can load that history using the loadhistory() function.