Introduction to R

Dr Liwan Liyanage - Western Sydney University

2 July 2018

Workshop Overview

Introduction to R

Supervised Learning

Regression

- Simple Linear Regression
- Multiple Linear Regression
- Polynomial Regression
- Interactions

Classification

- Logistic Regression
- Linear Discriminant Analysis
- Support Vector Machines (If time permits)

Cross Validation and Bootstrap

- Leave one out Cross Validation
- k Fold Cross Validation
- Bootstrap

Workshop 2

Tree Based Methods

- Regression Trees
- Classification Trees

Unsupervised Learning

- Dimension reduction Principal Component Analysis
- Clustering: K Means and Hierarchical

Unstructured Data (Will not be covered)

- Text Mining

Some simple R commands.

The best way to learn a new language is to try out the commands.

R can be downloaded from

http://cran.r-project.org/

Upload the following Libraries:

Note: you only need to do this once

- ISLR
- MASS
- readr
- car
- carData
- class
- boot

A function can have any number of inputs.

To create a vector of numbers, we use the function c()

```
x <- c(1,3,2,5)
x
```

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

or

```
x = c(1,3,2,5)
x
```

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

NOTE: Hitting the up arrow multiple times will display the previous commands, which can then be edited.

```
?vector
```

```
## starting httpd help server ... done
```

Can add two vectors as long as they both are of same length

```
y = c(1,4,3,5)

length(x)
```

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

```
length(y)
```

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

```
x+y
```

```
## [1]  2  7  5 10
```

Functions

- The ls() function allows us to look at a list of all of the objects, such as data and functions, that we have saved so far
- The rm() function can be used to delete any that we don't want

```
ls()
```

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

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

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

It's also possible to remove all objects at once:

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

The matrix() function can be used to create a matrix of numbers.

```
?matrix

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

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

NOTE: we could just type

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

NOTE: byrow=TRUE option can be used to populate the matrix in order of the rows.

```
matrix (c(1,2,3,4) ,2,2,byrow =TRUE)
```

Matrix manipulations

- The sqrt() function returns the square root of each element of a vector or matrix.
- The command x^2 raises each element of x to the power 2;
- any powers are possible, including fractional or negative powers.

```
sqrt(x)
```

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

```
x^2
```

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

The rnorm() function

It generates a vector of random normal variables,with first argument n the sample size Each time we call this function, we will get a different answer.Note: By default rnorm() function assumes 0 mean and std deviation 1

```
x=rnorm (15)
x
```

```
##  [1]  0.2649377  0.1723431 -0.9066573  0.8683953 -0.4076
##  [7]  1.3418131  2.7419605 -0.5535527 -0.4127718 -2.022
## [13]  0.4332702 -1.3704214 -0.4647398
```

```
y=rnorm (15)
y
```

```
##  [1]  0.98062176  0.30142343 -0.33042946  0.18285594 -0.
##  [6] -0.06814074  0.07946991  0.50548482  0.60520254 -0.
## [11] -1.56961875  0.83012053  1.90330128 -0.08769260  2.
```

cor() function

Here we create two correlated sets of numbers,x and y, and use the cor() function to compute the correlation between them.

Note: the mean and standard deviation can be altered using the mean and sd arguments, as illustrated below.

```
y=x+rnorm (15, mean=50, sd=.1)
cor(x,y)
```

```
## [1] 0.9963719
```

The set.seed() function takes an (arbitrary) integer argument.

When we want our code to reproduce the exact same set of random numbers; we can use the set.seed() function to do this

```
set.seed (126)
x1=rnorm (10)
x1
```

```
##  [1]  0.36673396  0.39645201 -0.73184372  0.94623642 -0.
##  [6] -0.07305128 -0.05417428  0.77326855 -0.53097278 -0.
```

```
set.seed (126)
x2=rnorm (10)
x2
```

```
##  [1]  0.36673396  0.39645201 -0.73184372  0.94623642 -0.
##  [6] -0.07305128 -0.05417428  0.77326855 -0.53097278 -0.
```

The mean() , var() and sd() functions

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

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

```
var(y)
```

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

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

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

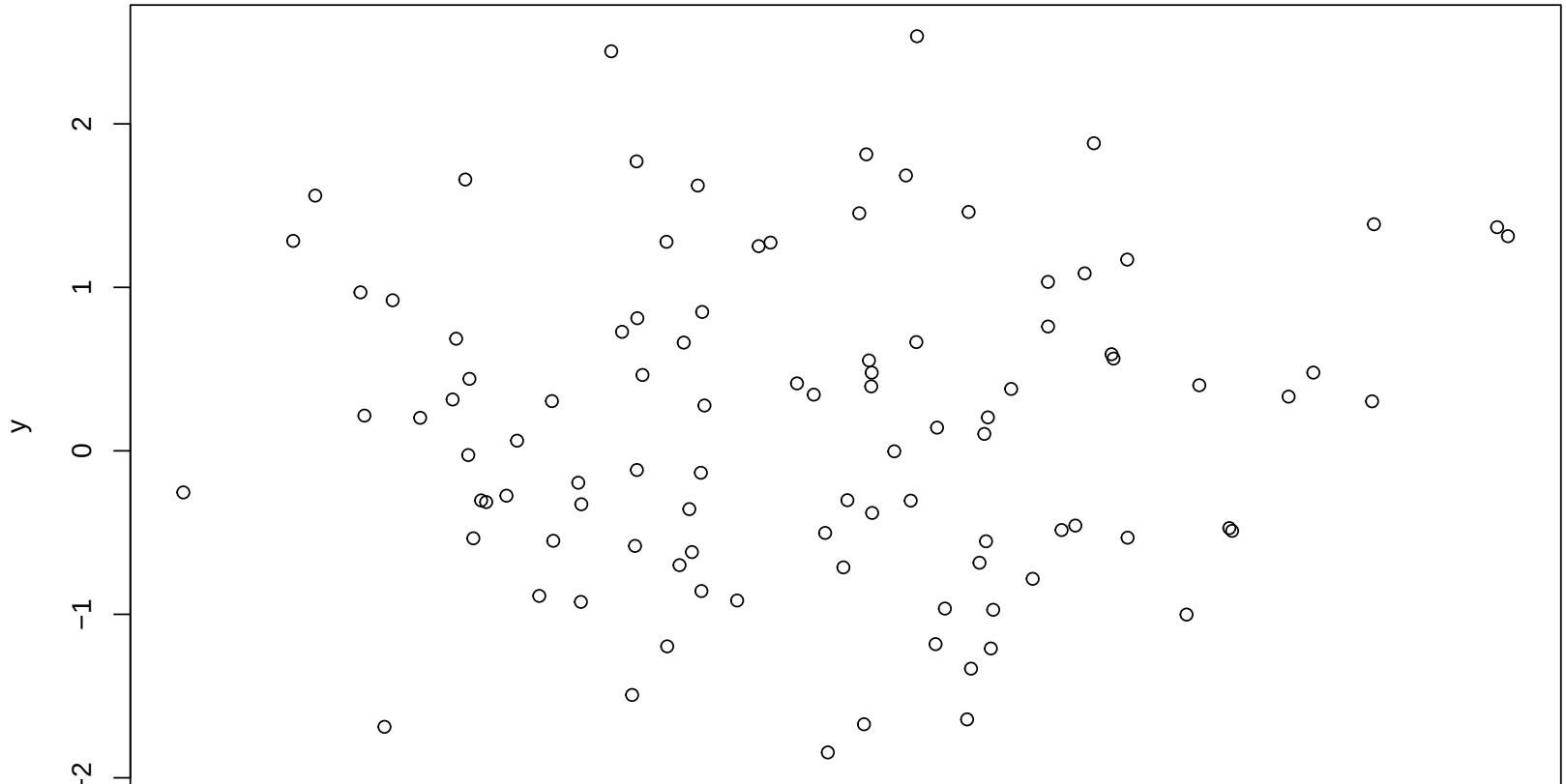
```
sd(y)
```

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

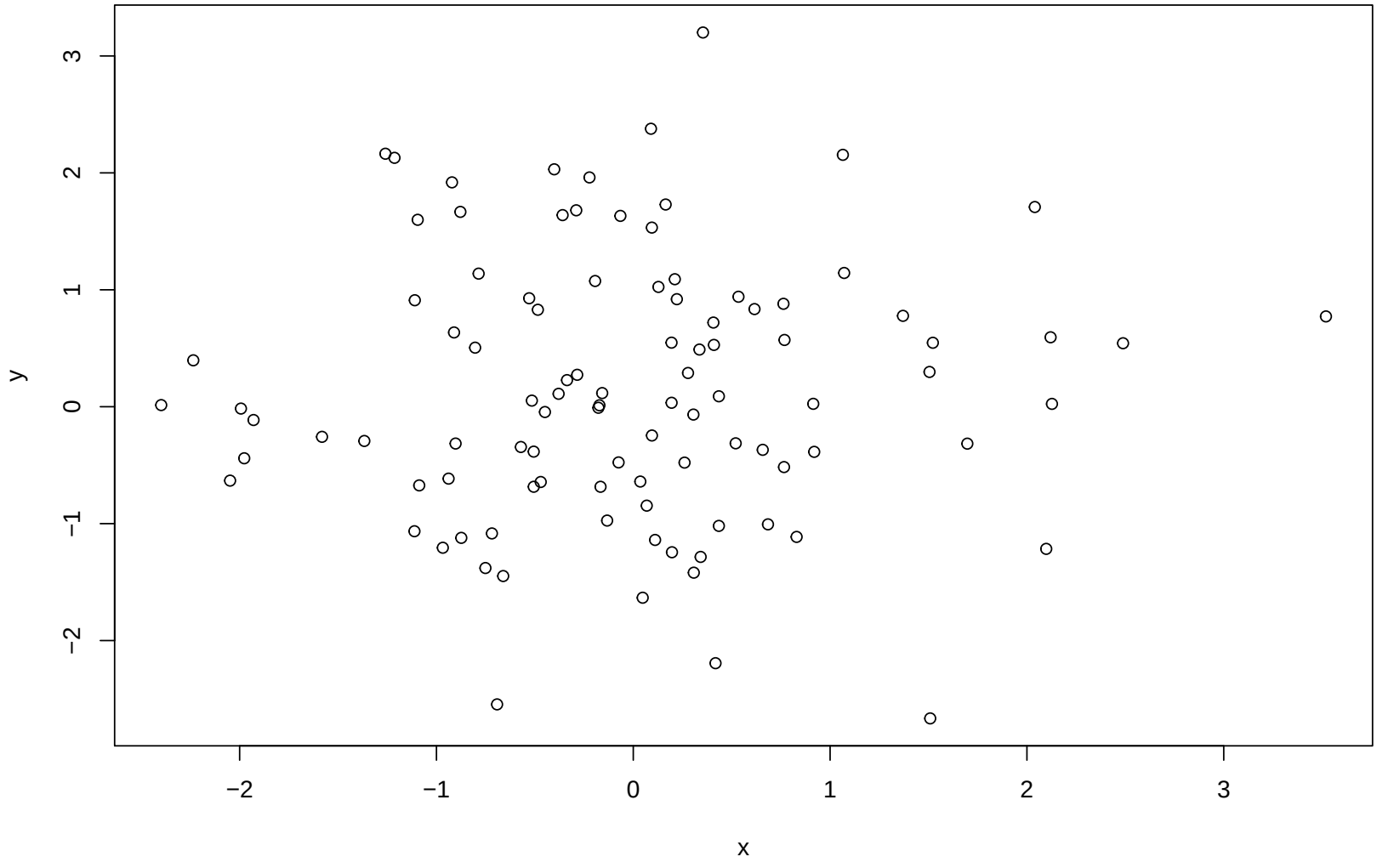
Graphics

The plot() function is the primary way to plot data in R.

```
x=rnorm (100)
y=rnorm (100)
plot(x,y)
```



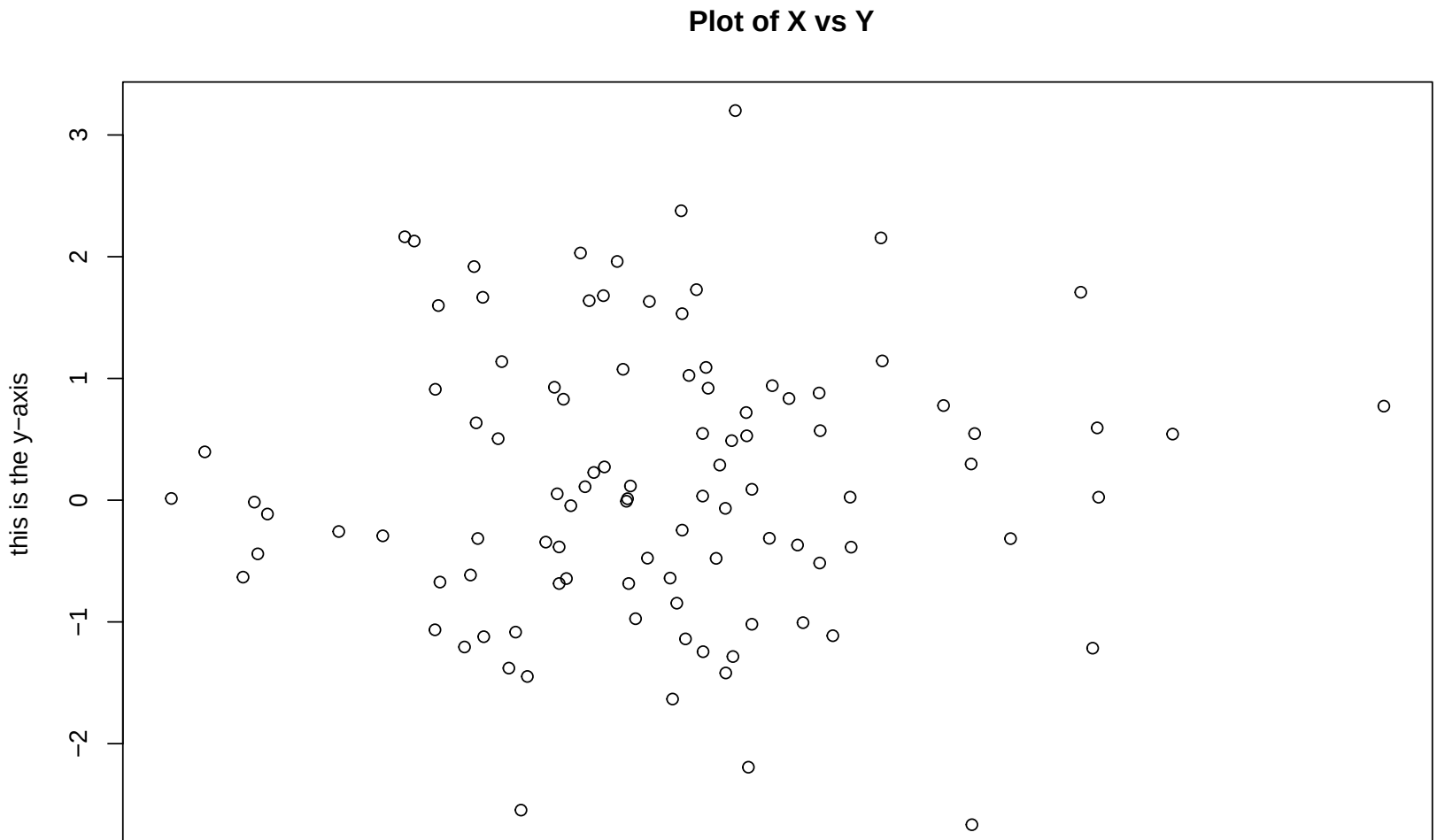
Plot without the code



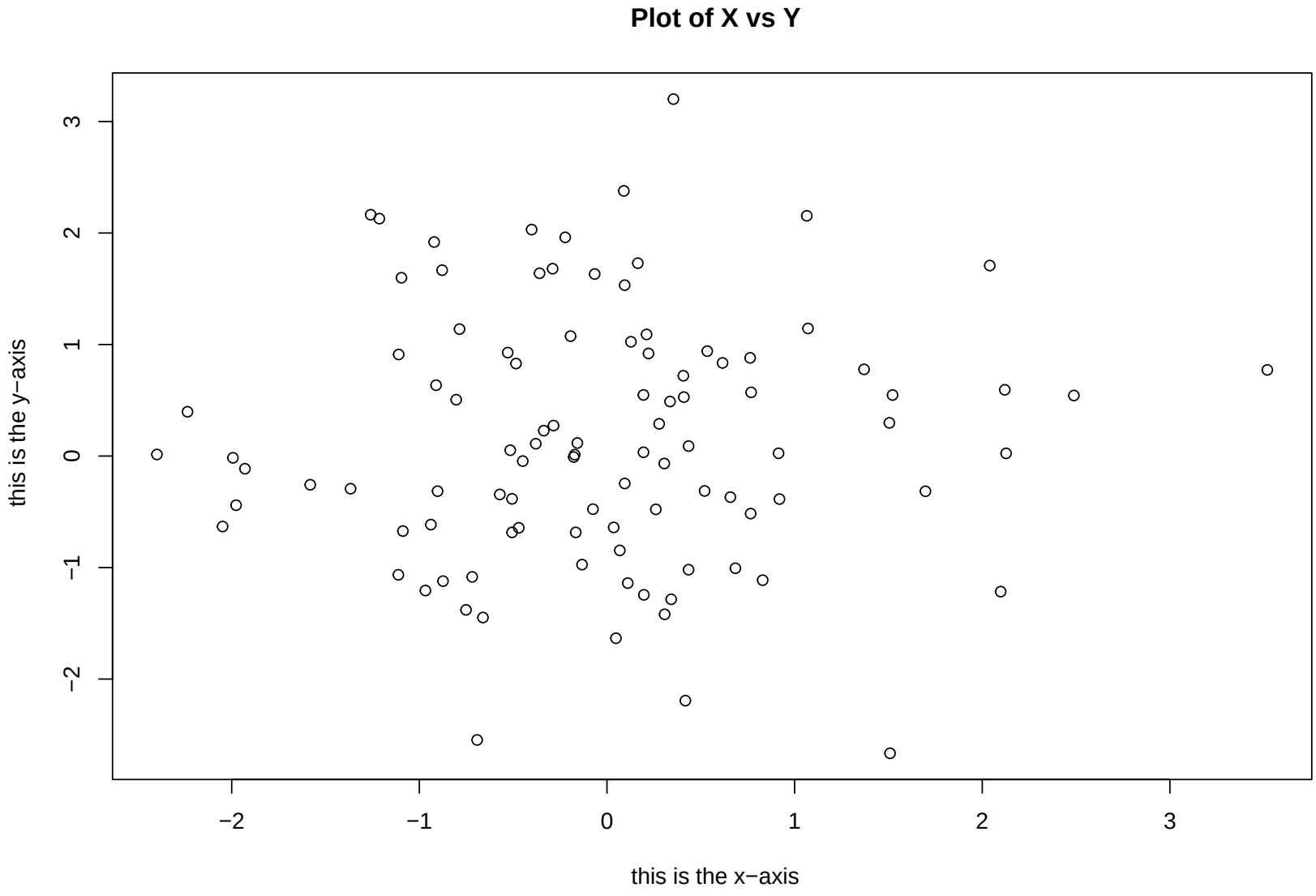
Passing in the argument xlab will result in a label on the x-axis.

To find out more information about the plot() function, type ?plot.

```
plot(x,y,xlab=" this is the x-axis",ylab=" this is the y-ax
```



Plot without the code



The function seq()

Can be used to create a sequence of numbers

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

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

```
y=1:10
y
```

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

```
z=seq(-pi ,pi ,length =10)
z
```

```
##  [1] -3.1415927 -2.4434610 -1.7453293 -1.0471976 -0.349(
##  [7]  1.0471976  1.7453293  2.4434610  3.1415927
```

The contour() function

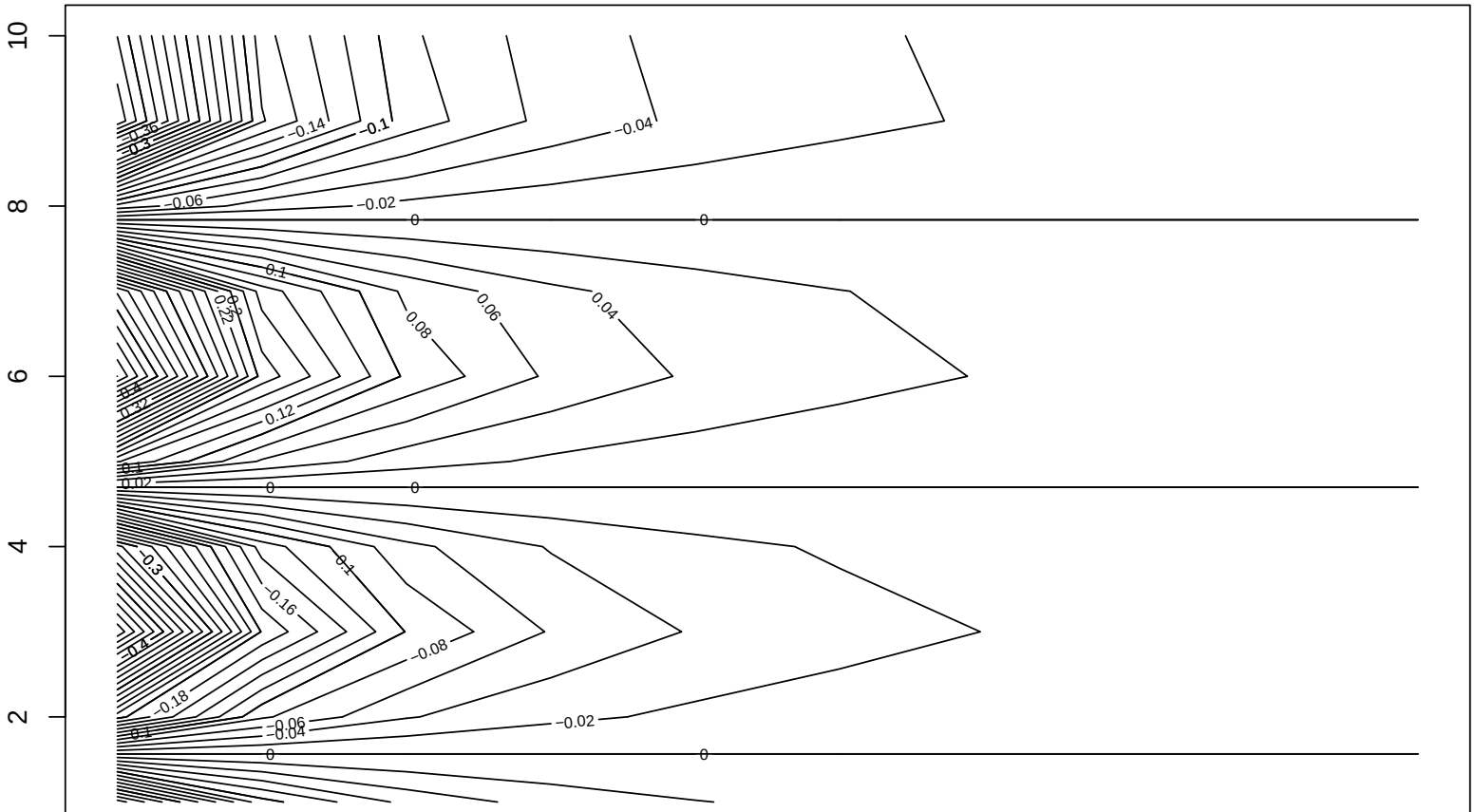
Produces a contour plot in order to represent three-dimensional data; it is like a topographical map. It takes three arguments:

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.

To learn more about these, take a look at the help file by typing ?contour.

The contour() function

```
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)
```



The image() function

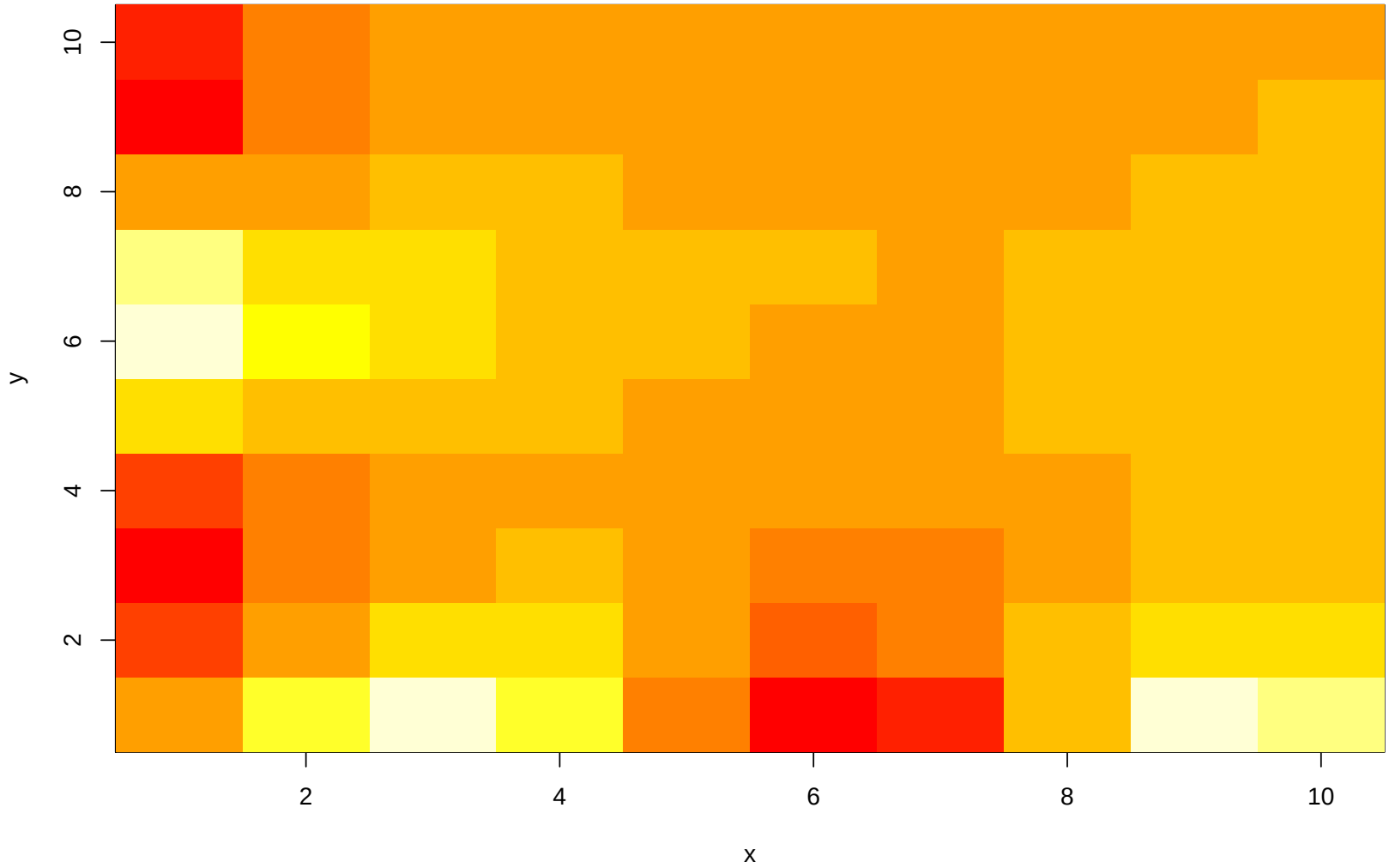
Works the same way as contour(), except that it produces a color-coded plot whose colors depend on the z value.

This is known as a heatmap, and is sometimes used to plot temperature in weather forecasts.

Alternatively, persp() can be used to produce a perspective plots, a three-dimensional plot. The arguments theta and phi control the angles at which the plot is viewed.

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

Map without code



Indexing Data

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

```
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
```

```
#Then, typing
A[2,3]
```

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

will select the element corresponding to the second row and the third column.

We can also select multiple rows and columns at a time, by providing vectors as the indices.

```
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
```

```r
A[1:2 ,]
```

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

```r
A[ ,1:2]
```

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

The last two examples include either no index for the columns or no index for the rows. These indicate that R should include all columns or all rows, respectively. R treats a single row or column of a matrix as a vector.

```
A[1,]
```

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

```
#The use of a negative sign - in the index tells R to keep

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
```

The dim() function

Outputs the number of rows followed by the number of columns of a given matrix.

```
dim(A)
```

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

Loading Data

For most analyses, the first step involves importing a data set into R

Reading a data set saved in the working folder

```
library(readr)
Auto <- read_csv("~/Kelania/Kelaniya/Statistical Learning/S
```

```
## Parsed with column specification:
## cols(
##   mpg = col_double(),
##   cylinders = col_integer(),
##   displacement = col_double(),
##   horsepower = col_character(),
##   weight = col_integer(),
##   acceleration = col_double(),
##   year = col_integer(),
##   origin = col_integer(),
##   name = col_character()
## )
```

Reading data from a library and View variable names

```
library("ISLR")
```

```
##
## Attaching package: 'ISLR'

## The following object is masked _by_ '.GlobalEnv':
##
##     Auto
```

```
attach(Auto)
View(Auto)
```

Data Explore

```
dim(Auto)
```

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

```
names(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsep
## [5] "weight"       "acceleration" "year"         "origin
## [9] "name"
```

Reading table ???

- Auto=read.table (“Auto.data”)
- fix(Auto)

Header in and mark missing values by “?”

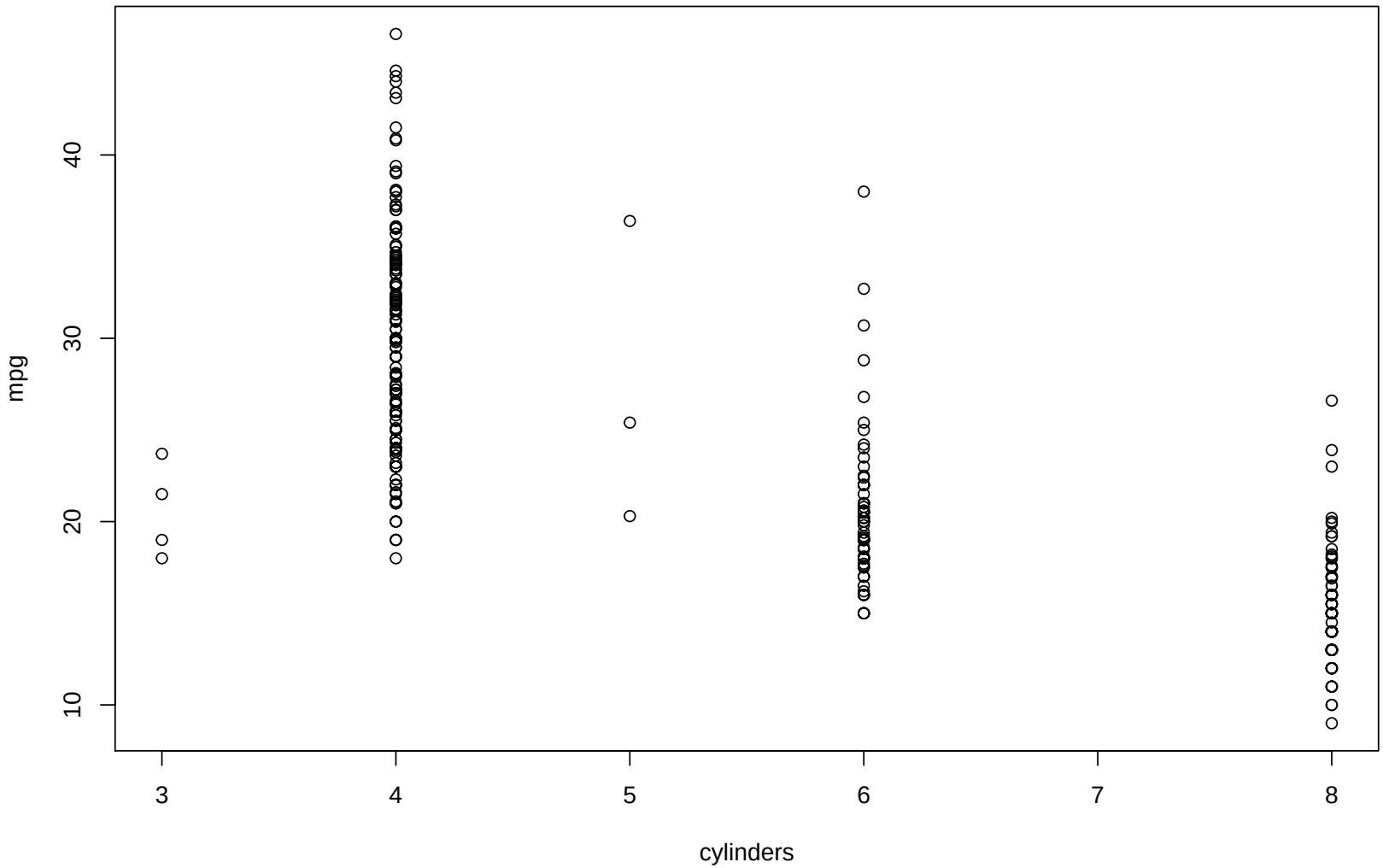
- Auto=read.table (“Auto.data”, header =T,na.strings =“?”)
- fix(Auto)

Omit the rows with missing values

- Auto=na.omit(Auto)
- dim(Auto)

Additional Graphical and Numerical Summaries

```
plot(cylinders , mpg)
```



as.factor() function

The cylinders variable is stored as a numeric vector, so R has treated it as quantitative. However, since there are only a small number of possible values for cylinders, one may prefer to treat it as a qualitative variable.

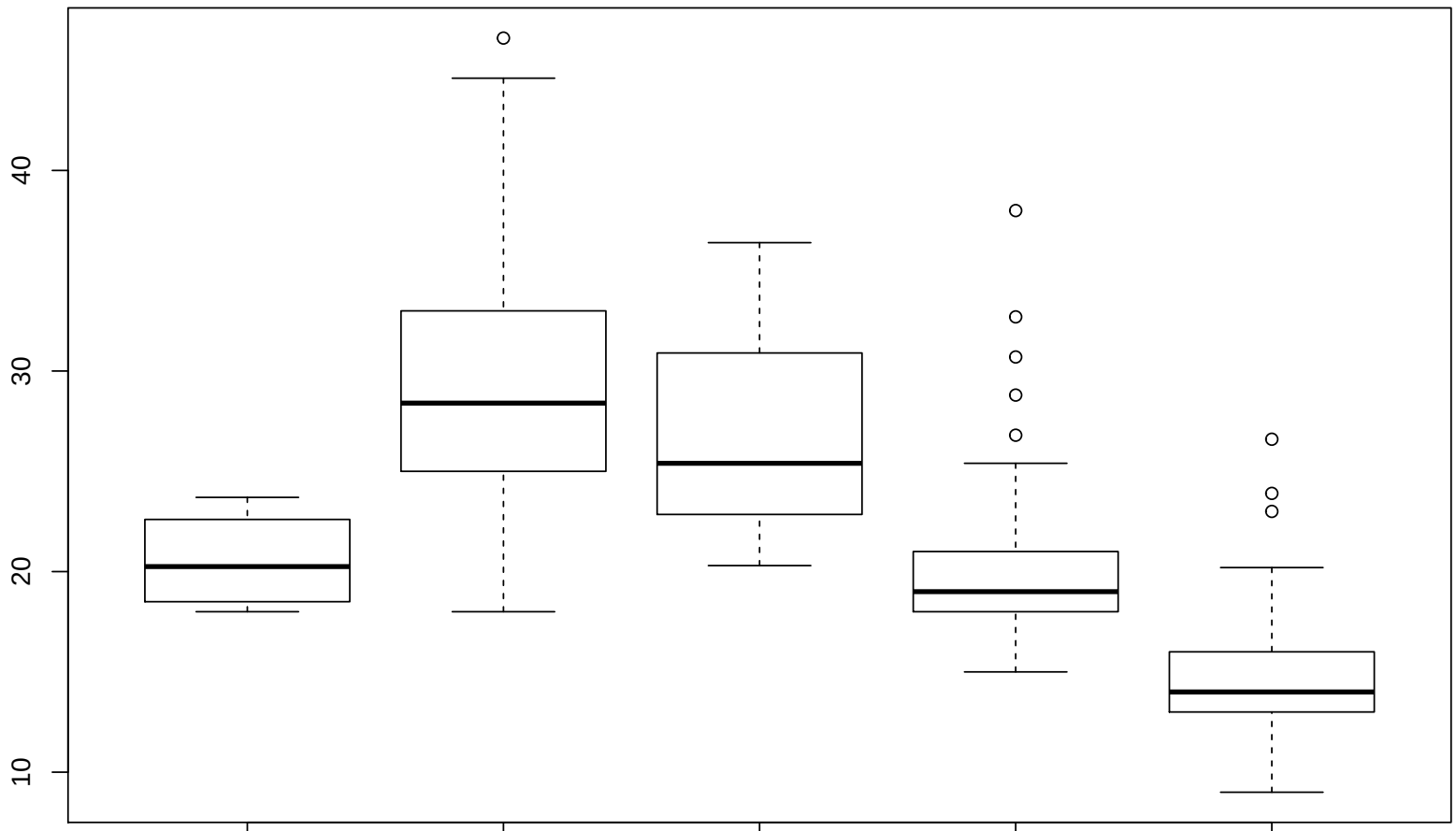
```
cylinders =as.factor(cylinders )
class(cylinders)
```

```
## [1] "factor"
```

Boxplots

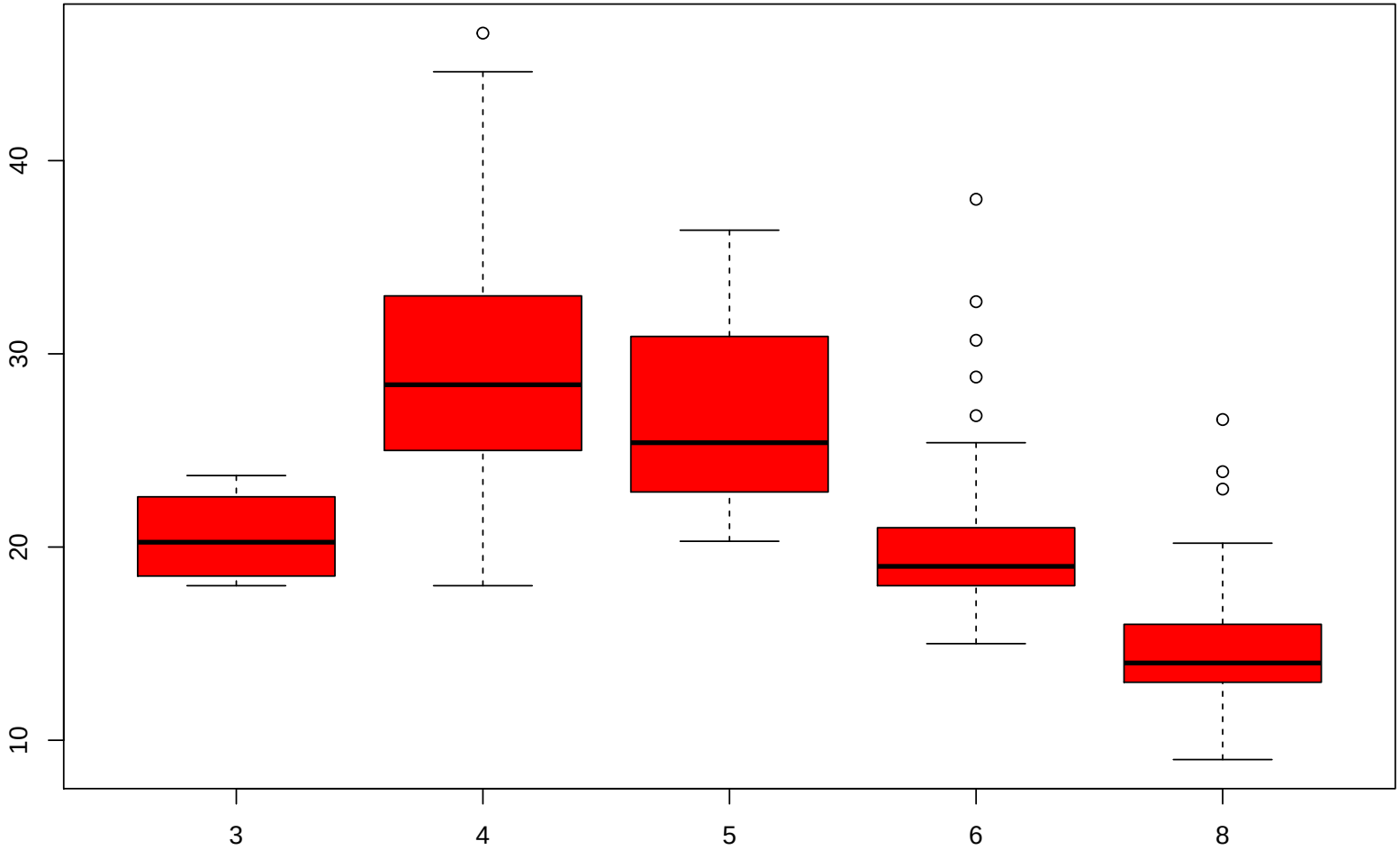
If the variable plotted on the x-axis is categorial, then boxplots will automatically be produced by the plot() function.

```
plot(cylinders , mpg)
```

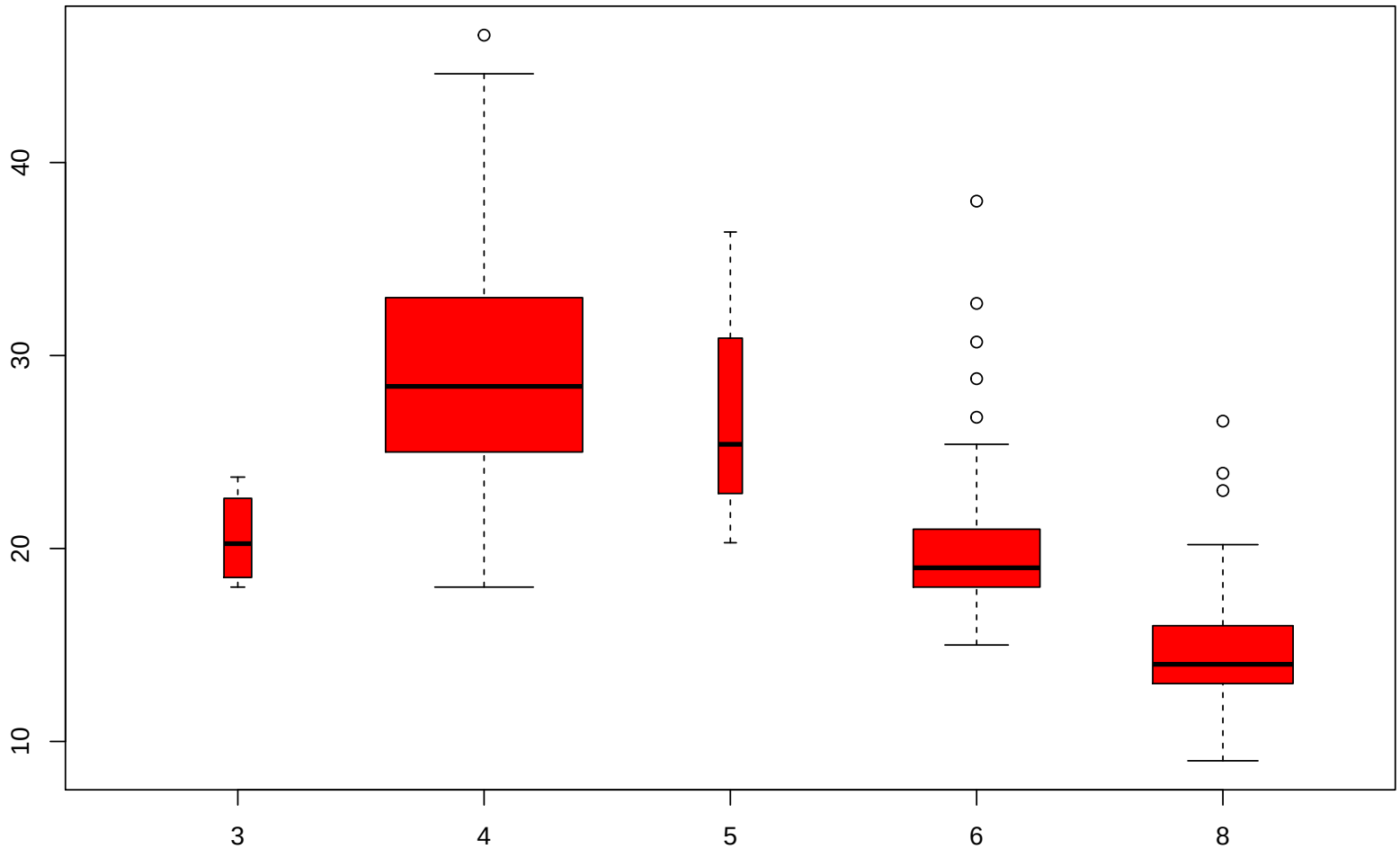


Costomise the plots

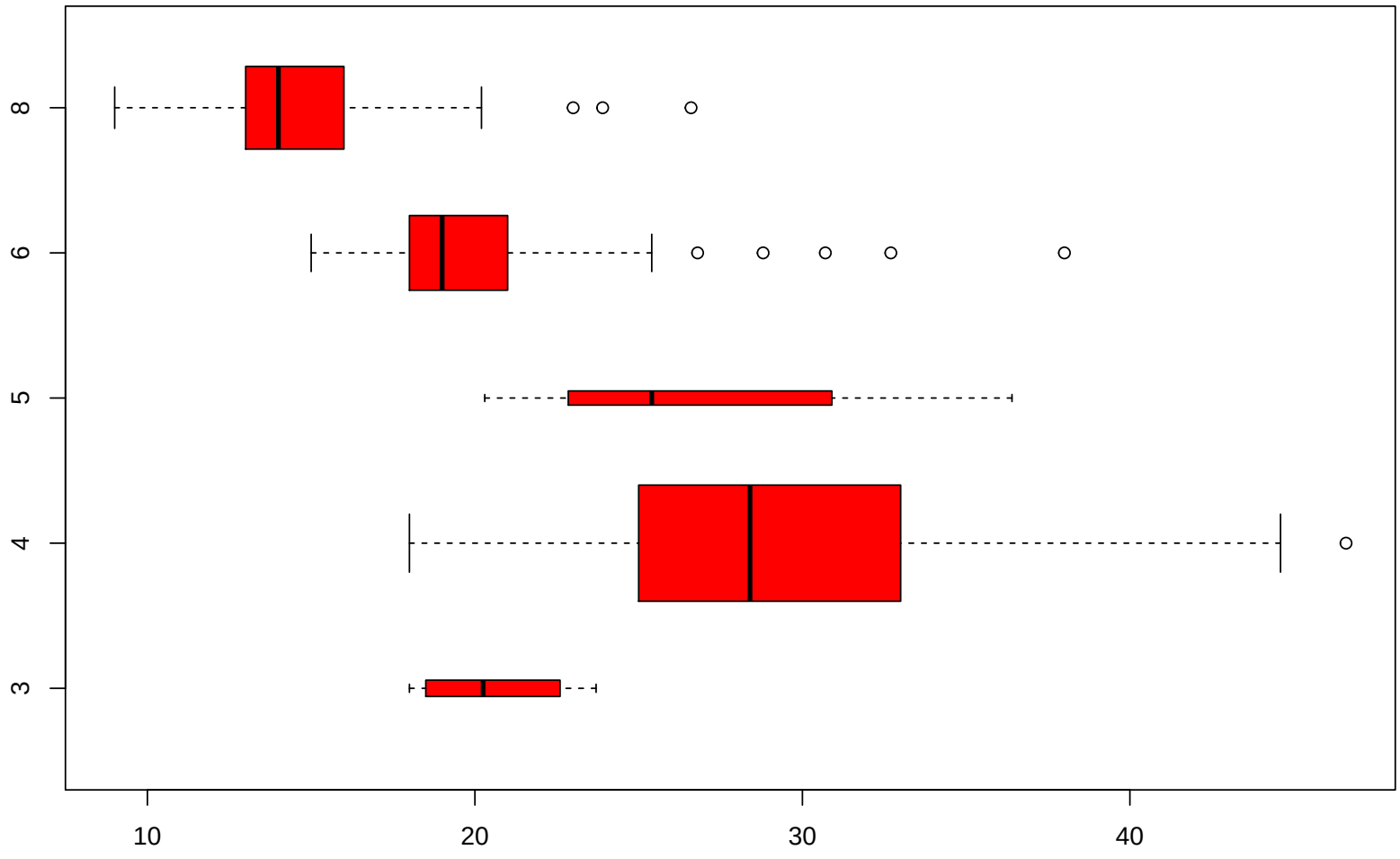
```
plot(cylinders , mpg , col ="red ")
```



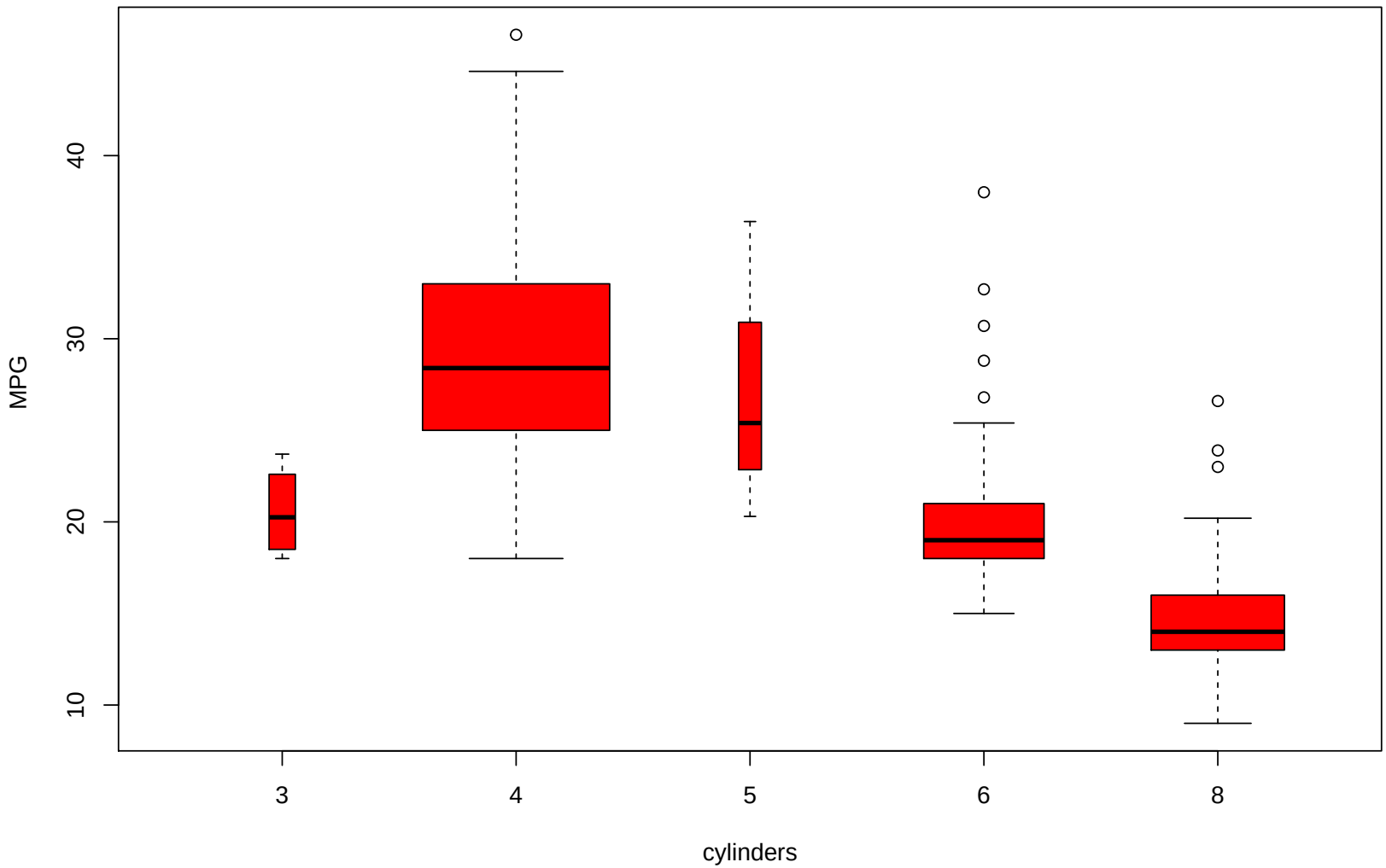
```
plot(cylinders , mpg , col ="red", varwidth =T)
```



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



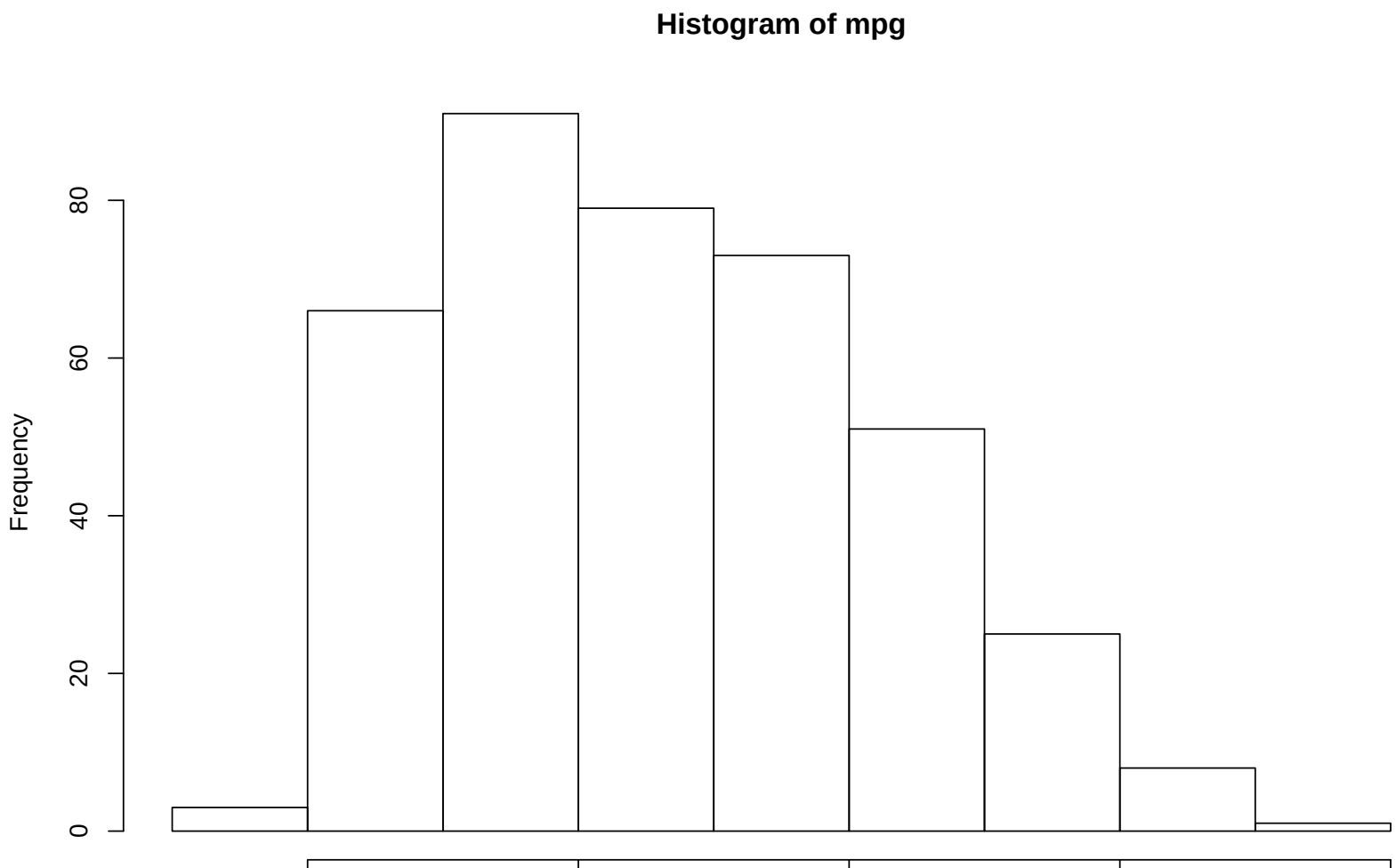
```
plot(cylinders , mpg , col ="red", varwidth =T, xlab=" cyli
ylab ="MPG ")
```



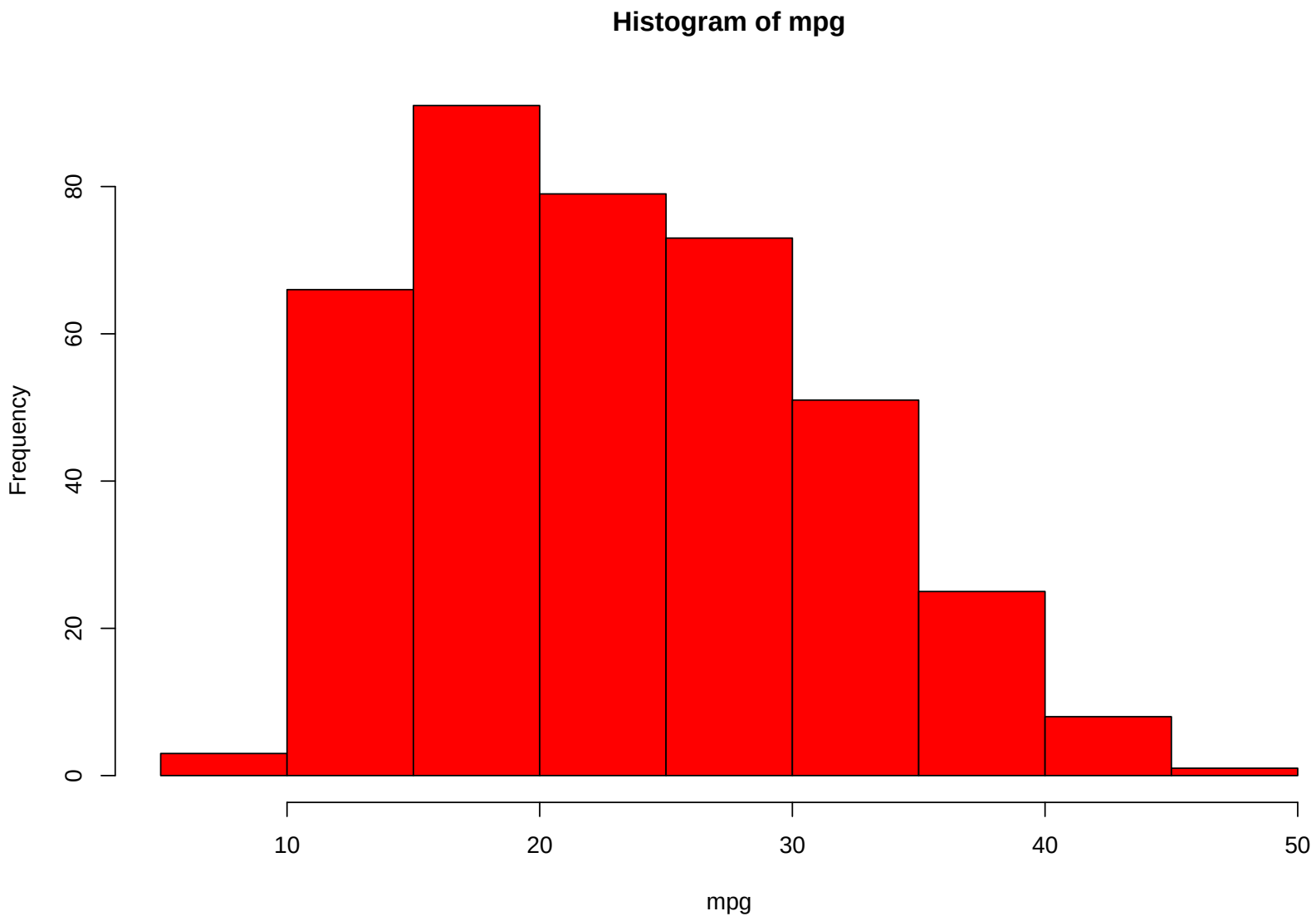
The hist() function

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

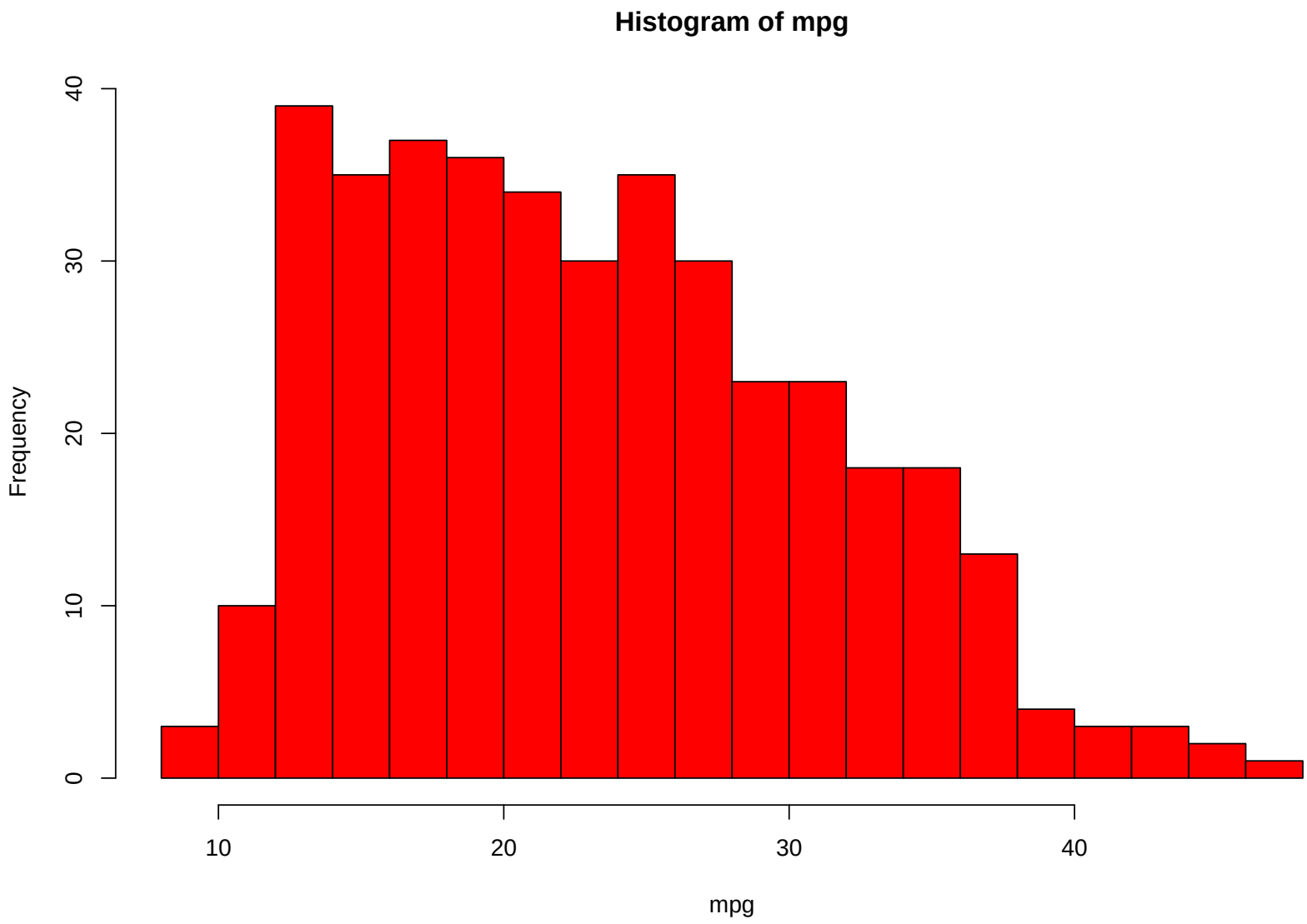
```
hist(mpg)
```



```r
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.

Before using pairs function check if all variales in the data frame is numeric. If not change all variables to numeric before using pairs function.

```
sapply(Auto, class)
```

```
##          mpg    cylinders displacement   horsepower
##    "numeric"    "integer"    "numeric"  "character"    "
## acceleration         year       origin         name
##    "numeric"    "integer"    "integer"  "character"
```

Change all variables to numeric variables

```
Auto[, c(2,4, 5, 7, 8)] <- sapply(Auto[, c(2,4, 5, 7, 8)],a
```

```
## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced
```

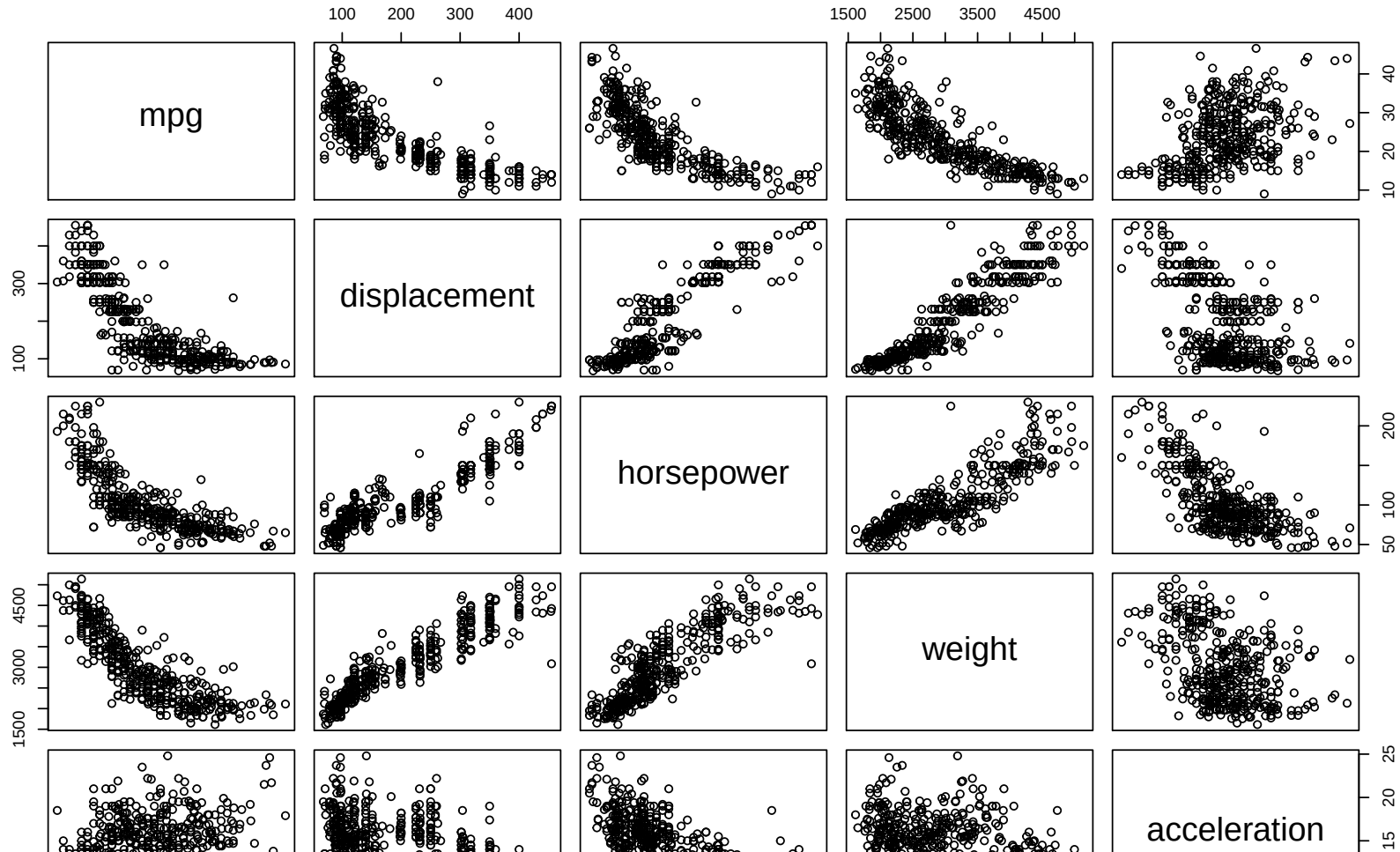
```
sapply(Auto, class)
```

```
##          mpg    cylinders displacement   horsepower
##    "numeric"    "numeric"    "numeric"    "numeric"
## acceleration         year       origin         name
##    "numeric"    "numeric"    "numeric"  "character"
```

Matrix plot

We can also produce scatterplots matrix for just a subset of the variables.

```
pairs(~ mpg + displacement + horsepower + weight + accelera
```



identify() function

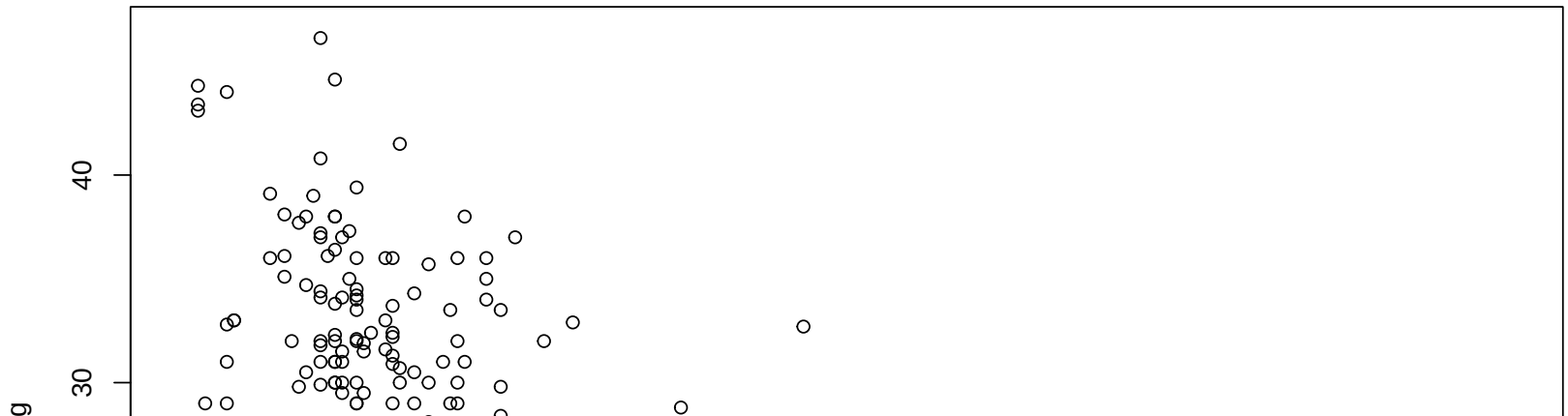
correspond to the rows for the selected points. Right-clicking on the plot will exit the identify() function

```
plot(horsepower ,mpg)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): NAs int
```

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

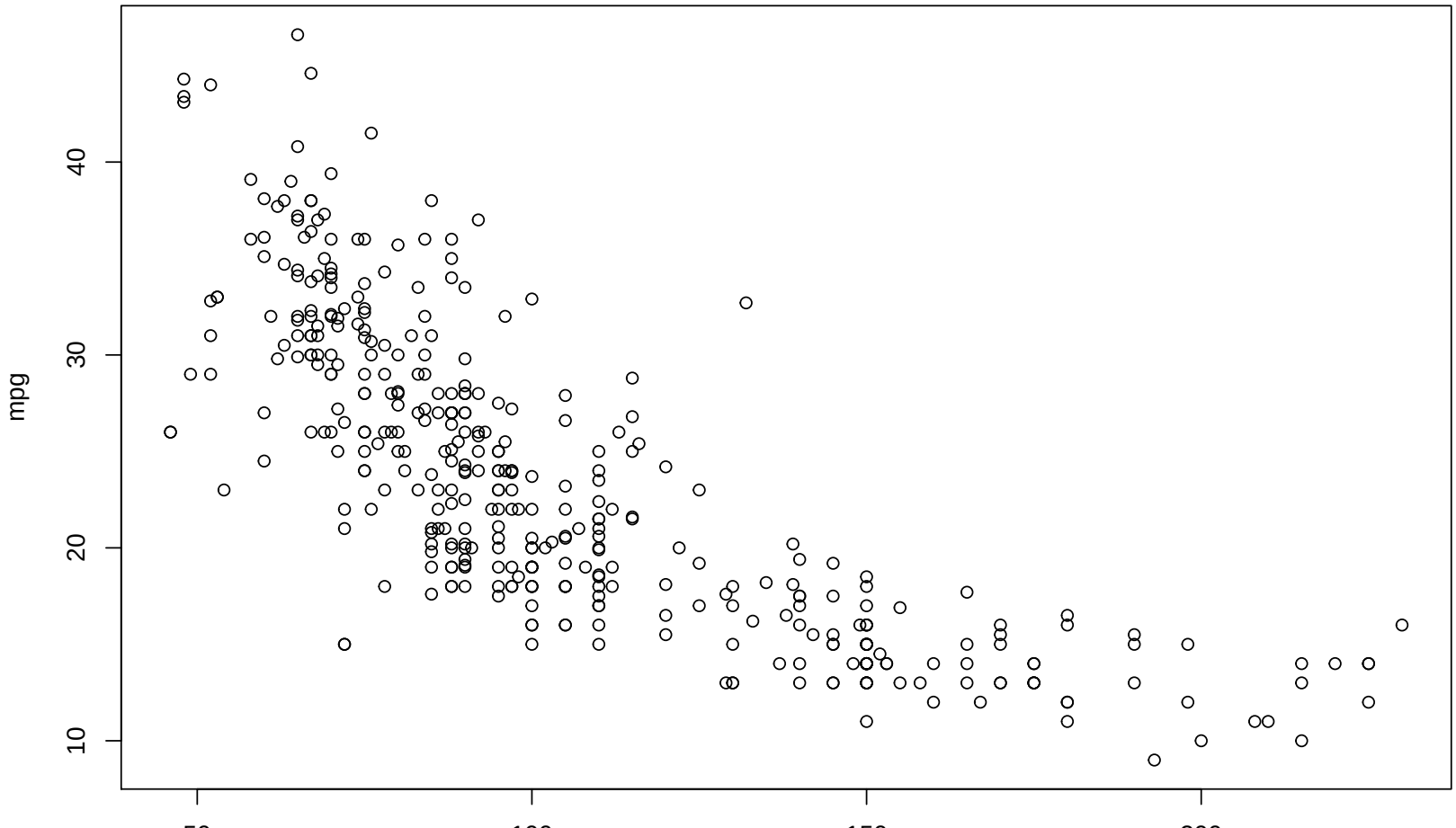
```
## Warning in xy.coords(x, y, setLab = FALSE): NAs introduc
```



Plot without code

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): NAs int

## Warning in xy.coords(x, y, setLab = FALSE): NAs introduc
```



The summary() function

produces a numerical summary of each variable in a particular data set. For qualitative variables such as name, R will list the number of observations that fall in each category.

```
##       mpg          cylinders      displacement     horse
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.
##  1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.0   1st Qu.
##  Median :23.00   Median :4.000   Median :146.0   Median
##  Mean   :23.52   Mean   :5.458   Mean   :193.5   Mean
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0   3rd Qu.
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.
##                                                  NA's
##      weight      acceleration        year           orig
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.
##  1st Qu.:2223   1st Qu.:13.80   1st Qu.:73.00   1st Qu.
##  Median :2800   Median :15.50   Median :76.00   Median
##  Mean   :2970   Mean   :15.56   Mean   :75.99   Mean
##  3rd Qu.:3609   3rd Qu.:17.10   3rd Qu.:79.00   3rd Qu.
```

We can also produce a summary of just a single variable.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.00   17.50   23.00   23.52   29.00   46.60
```

Quitting R: q() function, savehistory() function, loadhistory() function

- 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.

write.table() function to export data.

To A Tab Delimited Text File

- write.table(mydata, "c:/mydata.txt", sep="\t")

To an Excel Spreadsheet

- library(xlsx)
- write.xlsx(mydata, "c:/mydata.xlsx")