

# Statistical Learning

ISLR- Done by Prateek Minhas

2025-10-18

## Ch-2: Statistical Learning

### Lab

#### Basic commands

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

## [1] 1 3 4 5

y=c(4,5,80,-3)
y

## [1] 4 5 80 -3
length(x)

## [1] 4
length(x)+length(y)

## [1] 8
x-y

## [1] -3 -2 -76    8
z=x+y

ls()

ls() and Rm()

## [1] "x" "y" "z"

rm(z)
ls()

## [1] "x" "y"
rm(list=ls()) #empty out the list
ls()

## character(0)
```

```
x = matrix(data=c(1,2,3,4,5,6), nrow=2 , ncol = 3, byrow=TRUE)
x
```

matrix() function

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
y = matrix(data=c(1,2,3,4,5,6), nrow=2 , ncol = 3, byrow=FALSE)
y
```

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

removing the data=, nrow=, ncol= we can also write directly , by default we get byrows = false meaning the columns get filled first

```
z= matrix(c(1,2,3,4,5,6), 3,2)
z
```

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

```
sqrt(x)
```

```
##      [,1]      [,2]      [,3]
## [1,]    1 1.414214 1.732051
## [2,]    2 2.236068 2.449490
```

```
y^2
```

```
##      [,1] [,2] [,3]
## [1,]    1    9   25
## [2,]    4   16   36
```

rnorm() gives random values every time we use it

```
x=rnorm(50)
```

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

```
## [1] 0.9930791
```

we use set seed() so that we get same random numbers every time for a particular seed value

```
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)

## [1] 0.01103557

var(y)

## [1] 0.7328675

sqrt(var(y))

## [1] 0.8560768

sd(y)

## [1] 0.8560768
#as we see variance = standard deviation.sq

```

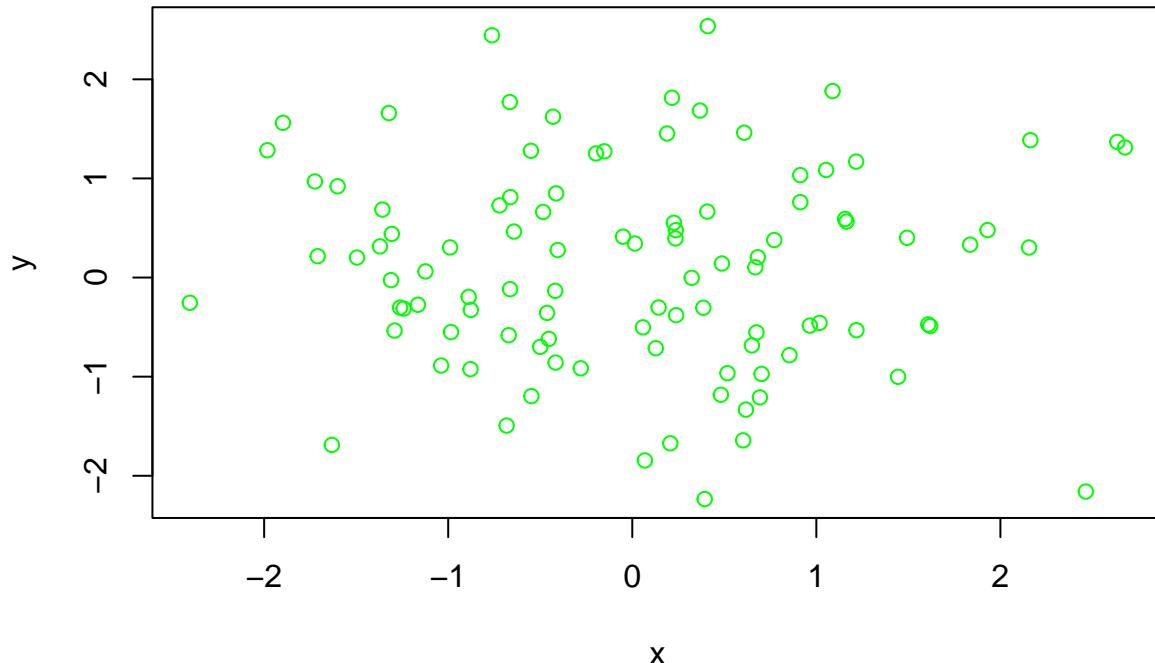
## Graphics

```

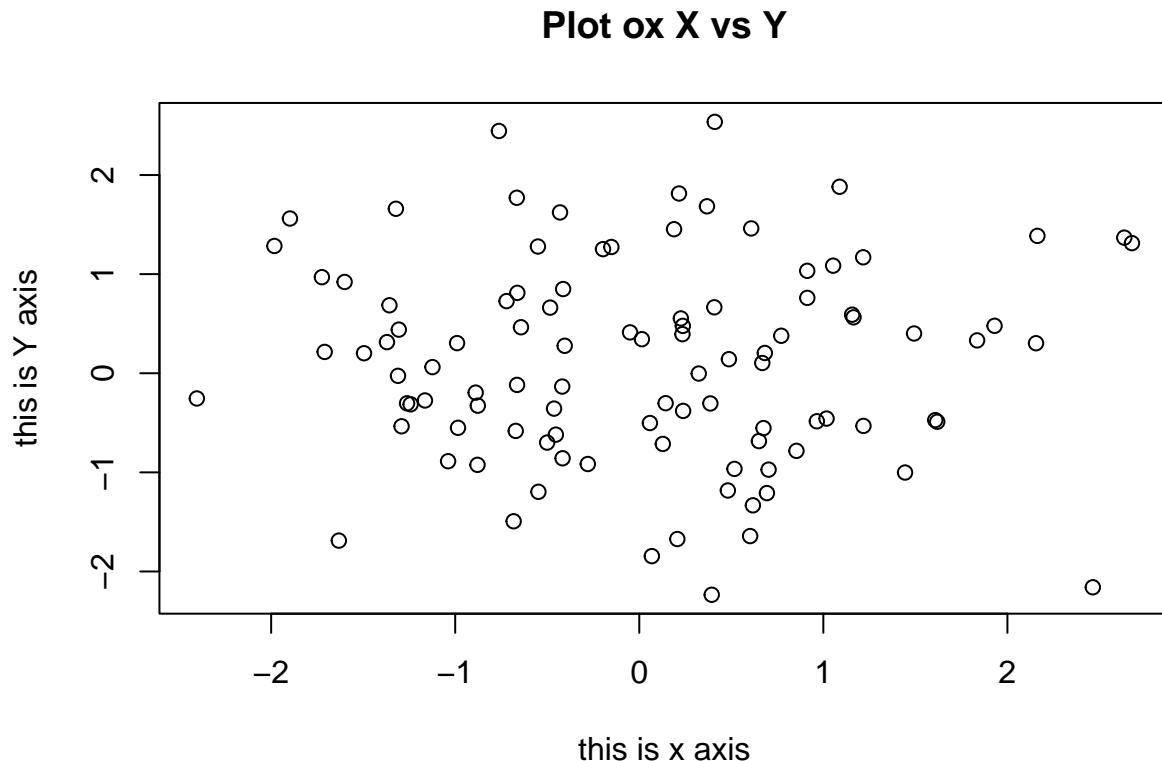
plot()

x=rnorm(100)
y=rnorm(100)
plot(x,y, col="green")

```



```
plot(x,y,xlab = "this is x axis", ylab = "this is Y axis", main="Plot ox X vs Y")
```



```
Seq()
```

```
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=50)
```

```
z
```

```
## [1] -3.14159265 -3.01336438 -2.88513611 -2.75690784 -2.62867957 -2.50045130
```

```
## [7] -2.37222302 -2.24399475 -2.11576648 -1.98753821 -1.85930994 -1.73108167
```

```
## [13] -1.60285339 -1.47462512 -1.34639685 -1.21816858 -1.08994031 -0.96171204
```

```
## [19] -0.83348377 -0.70525549 -0.57702722 -0.44879895 -0.32057068 -0.19234241
```

```
## [25] -0.06411414  0.06411414  0.19234241  0.32057068  0.44879895  0.57702722
```

```
## [31]  0.70525549  0.83348377  0.96171204  1.08994031  1.21816858  1.34639685
```

```
## [37]  1.47462512  1.60285339  1.73108167  1.85930994  1.98753821  2.11576648
```

```
## [43]  2.24399475  2.37222302  2.50045130  2.62867957  2.75690784  2.88513611
```

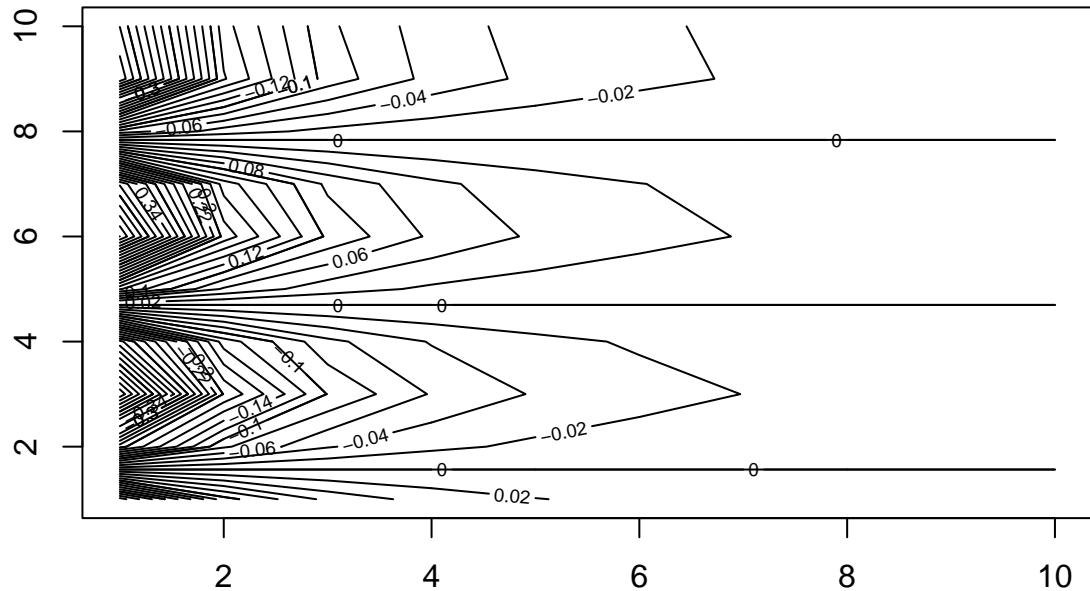
```
## [49]  3.01336438  3.14159265
```

```
contour() for 3d graphs
```

```

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)

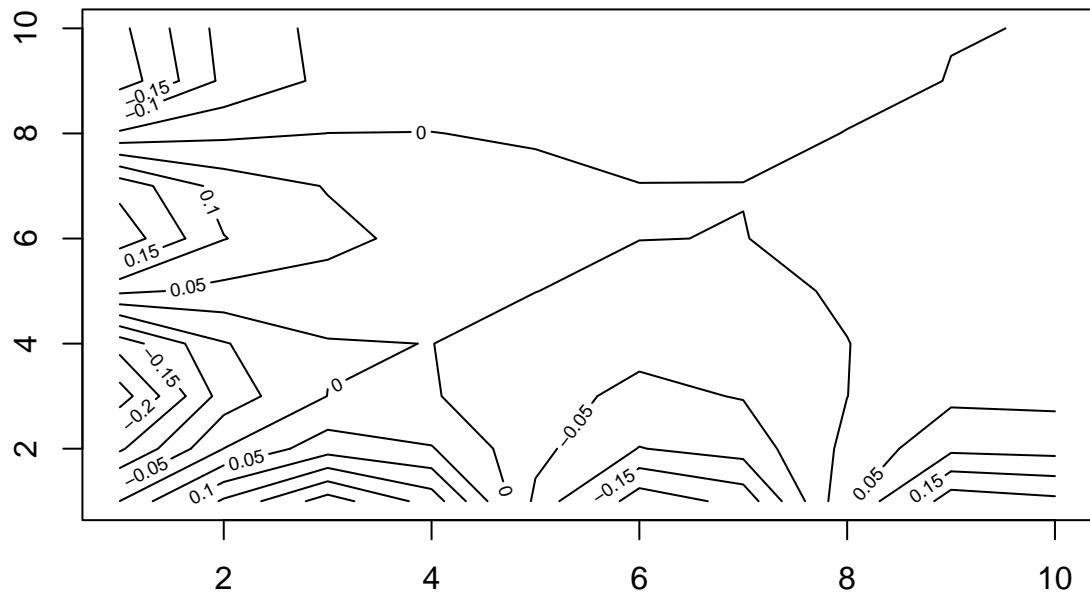
```



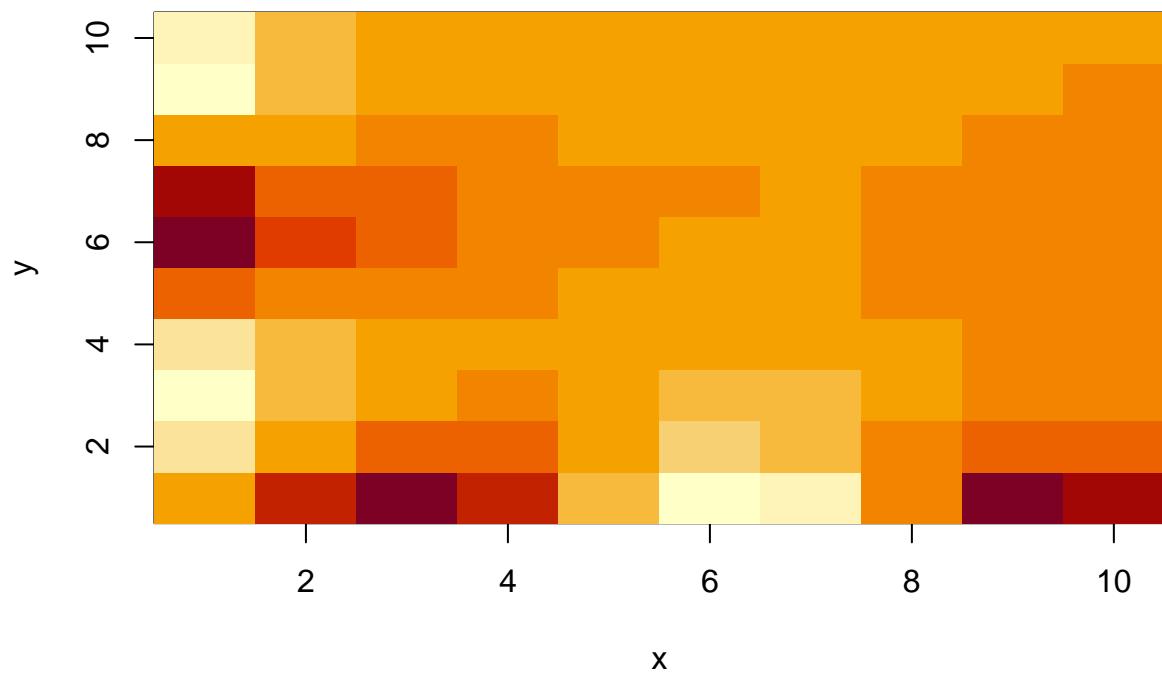
```

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

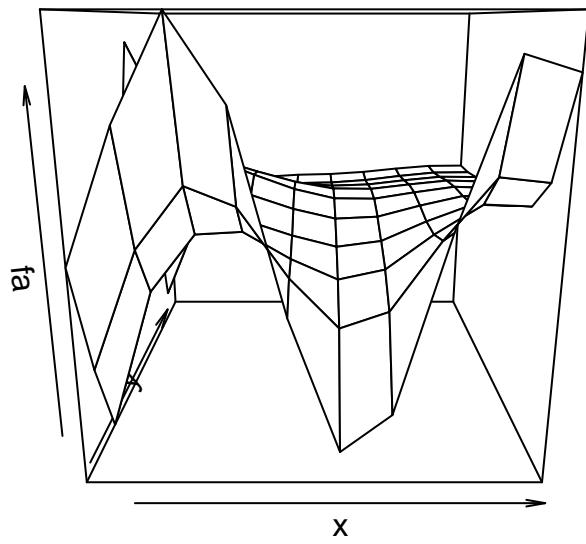
```



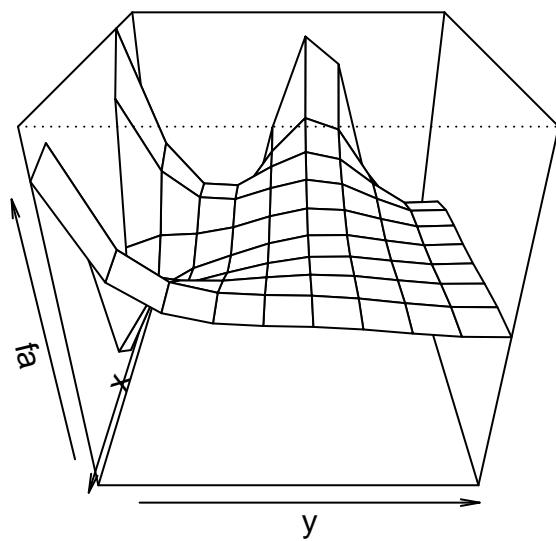
```
image(x,y,fa)
```



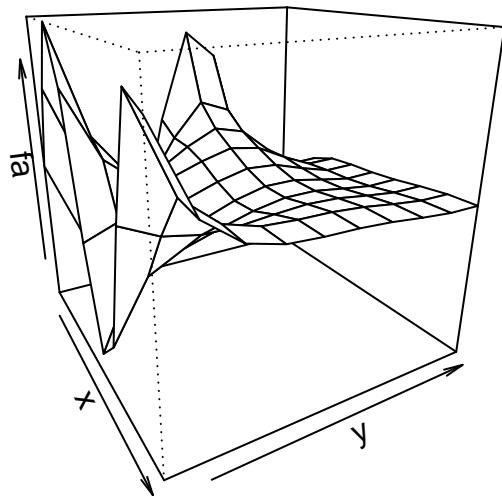
```
persp(x,y,fa)
```



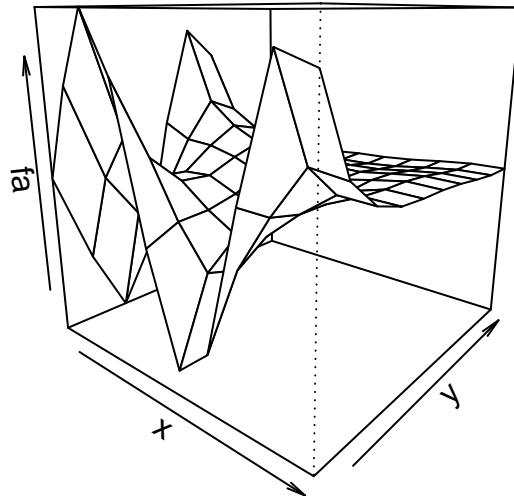
```
persp(x,y,fa, theta = 90, phi=30)
```



```
persp(x,y,fa, theta = 60, phi=20)
```



```
persp(x,y,fa, theta = 40, phi=15)
```



## 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

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

A[,1:2]

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

A[1,]

## [1] 1 5 9 13

-ve sign indicates that select all the elements Except those mentioned , it is like a negation
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[c(1,3),]

##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    3    7   11   15

A[-c(1,3),]#second and last row

##      [,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

```

Auto = read.table("Auto.data")
#View(Auto)
head(Auto)

##      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
Auto = read.table("Auto.data", na.strings = "?", header = T , stringsAsFactors = T )
head(Auto)

```

```

##   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
## 5 17         8            302       140    3449        10.5    70      1
## 6 15         8            429       198    4341        10.0    70      1
##
##           name
## 1 chevrolet chevelle malibu
## 2          buick skylark 320
## 3        plymouth satellite
## 4          amc rebel sst
## 5          ford torino
## 6        ford galaxie 500

```

```

Auto = read.csv("Auto.csv", na.strings = "?", stringsAsFactors = T)
attach(Auto)
head(Auto)

```

```

##   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
## 5 17         8            302       140    3449        10.5    70      1
## 6 15         8            429       198    4341        10.0    70      1
##
##           name
## 1 chevrolet chevelle malibu
## 2          buick skylark 320
## 3        plymouth satellite
## 4          amc rebel sst
## 5          ford torino
## 6        ford galaxie 500

```

the strings as factors argument tells that if the value is string , treat the variable as qualitative

```
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

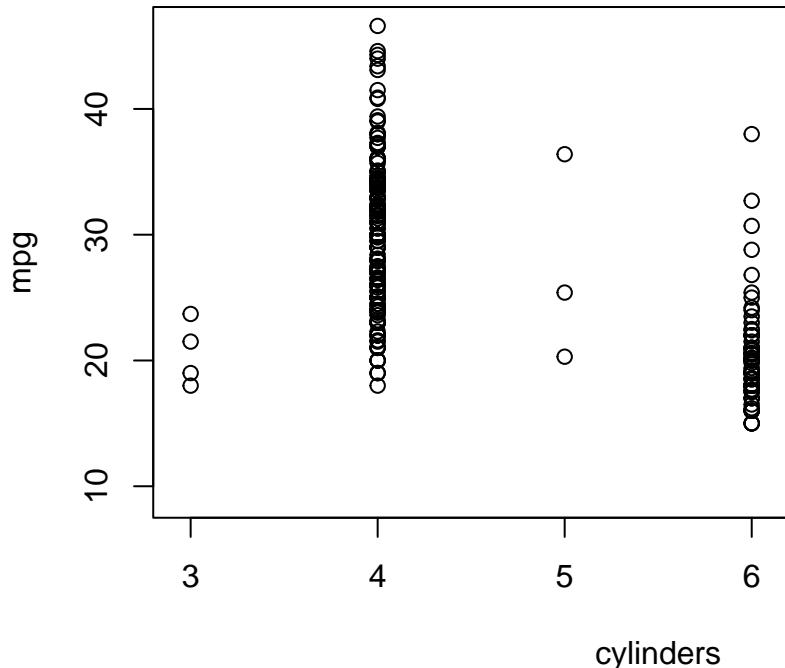
Auto= na.omit(Auto)
dim(Auto) #as we see number of rows are reduced

## [1] 392   9
names(Auto)

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

plot(cylinders,mpg)

```



### Additional graphical and numerical Summaries

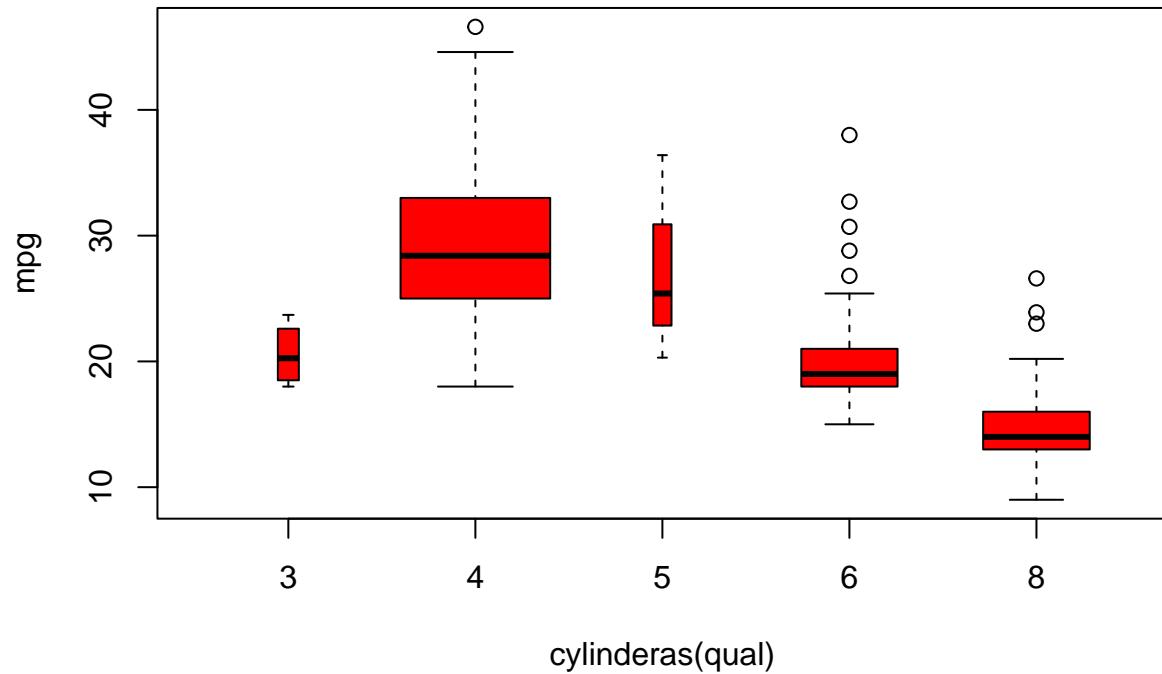
```
#the cylinders is quantitative
```

The `as.factor()` function converts quantitative variables into qualitative variables.

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

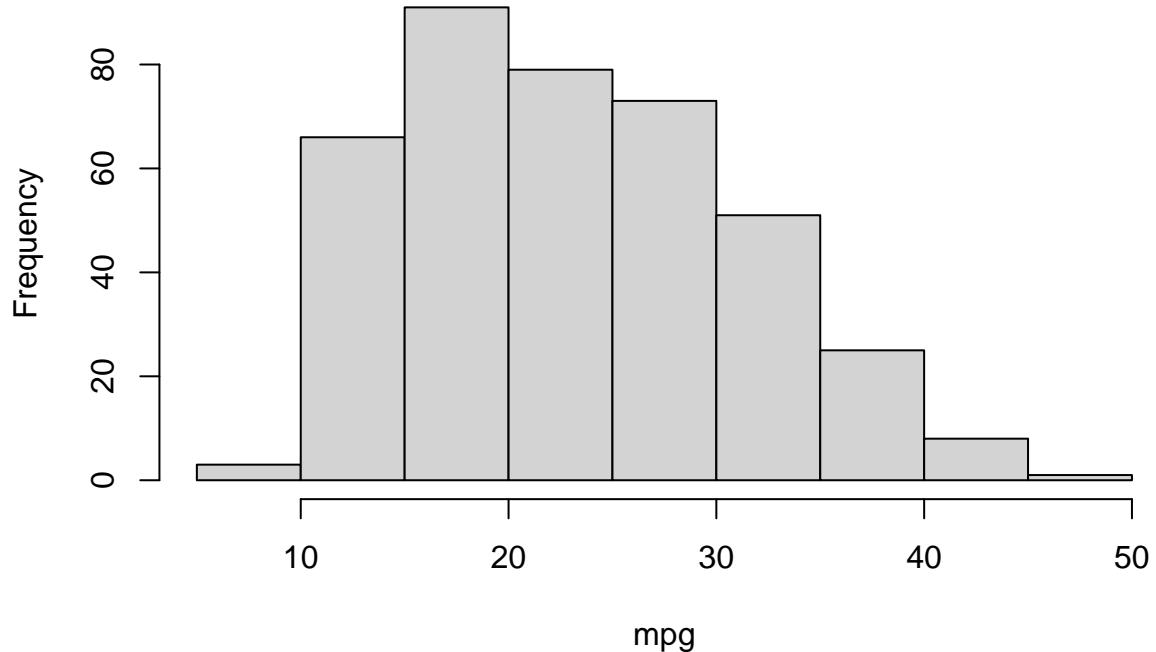
*If the variable plotted on the x-axis is qualitative, then boxplots will automatically be produced by the `plot()` function.*

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



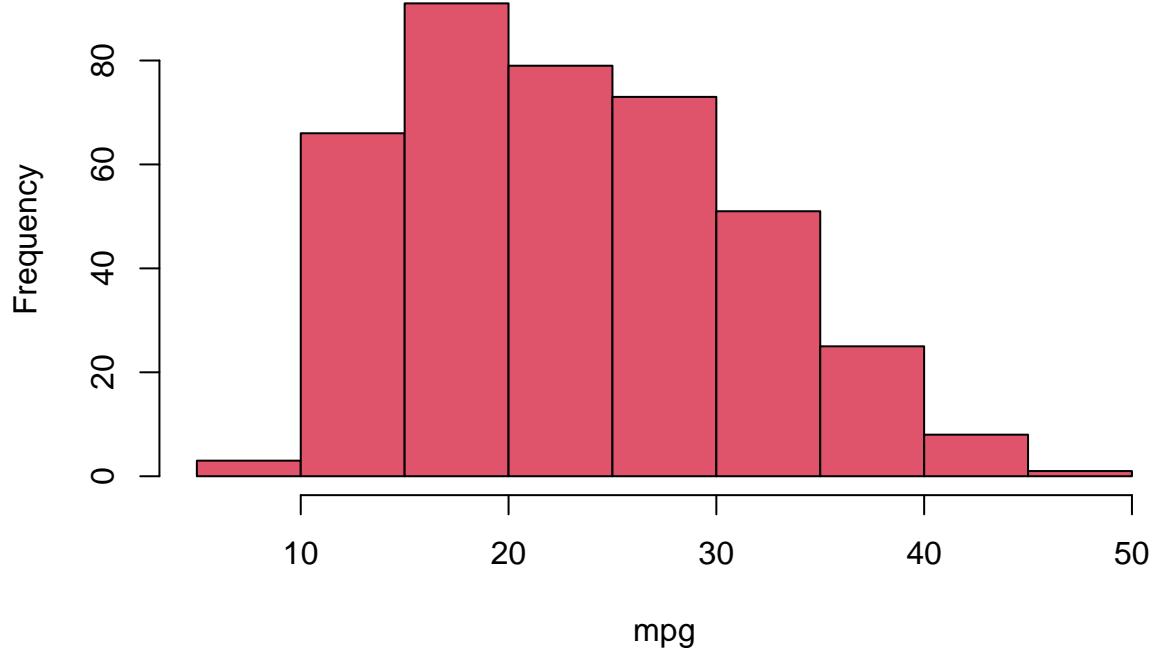
```
hist(mpg)
```

### Histogram of mpg



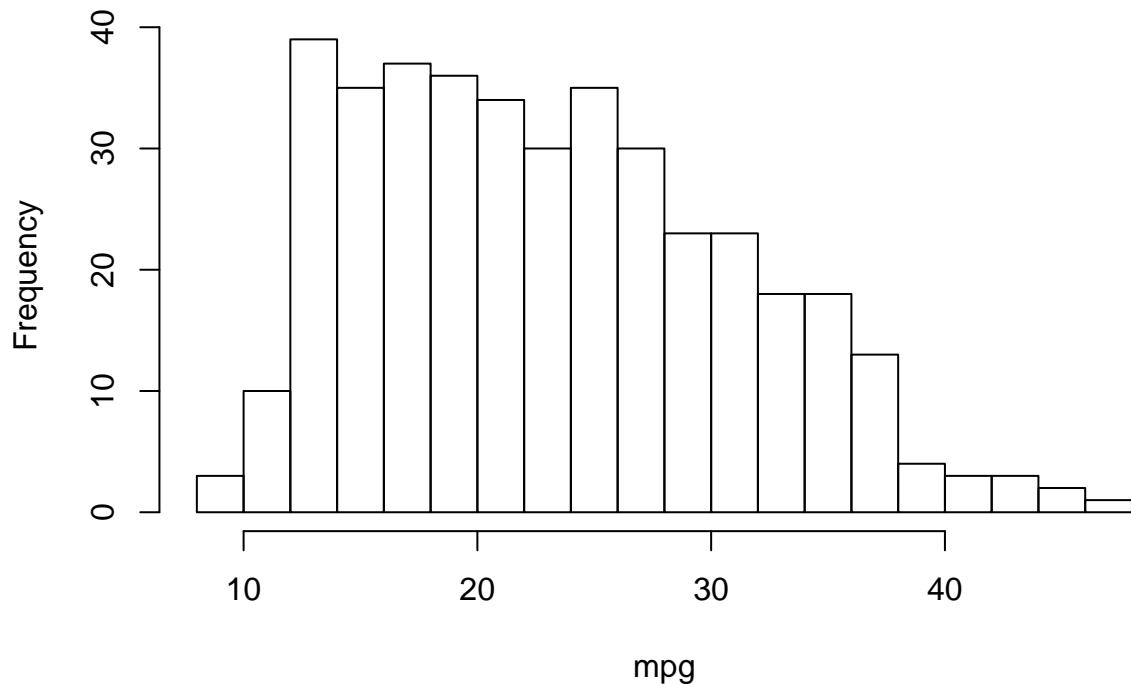
```
hist(mpg, col=2)
```

### Histogram of mpg



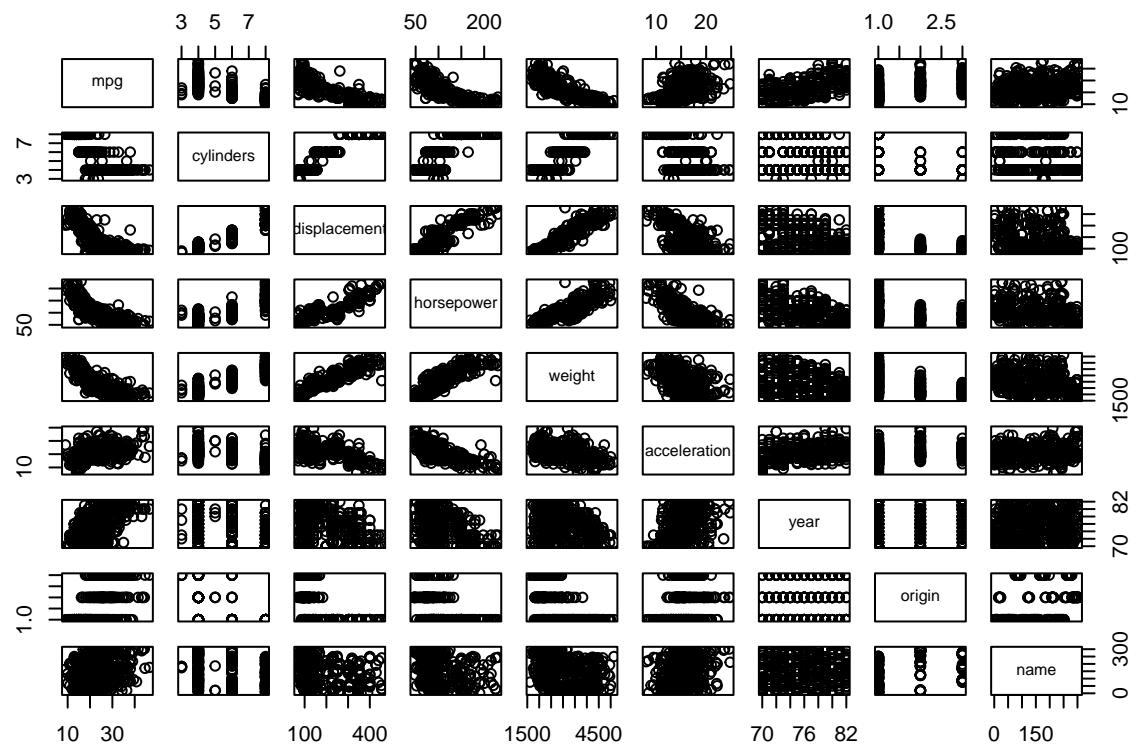
```
hist(mpg, col=0, breaks=15)
```

## Histogram of mpg

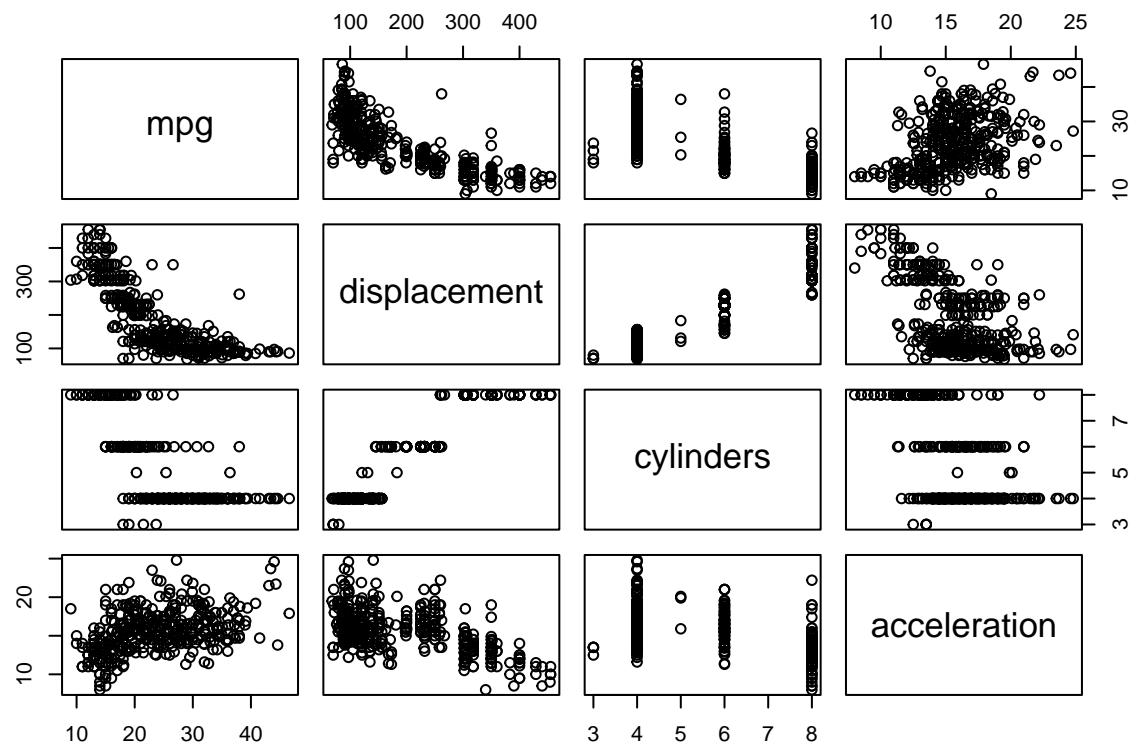


The **pairs()** function creates a scatterplotmatrix, i.e. a scatterplot for every pair of variables.

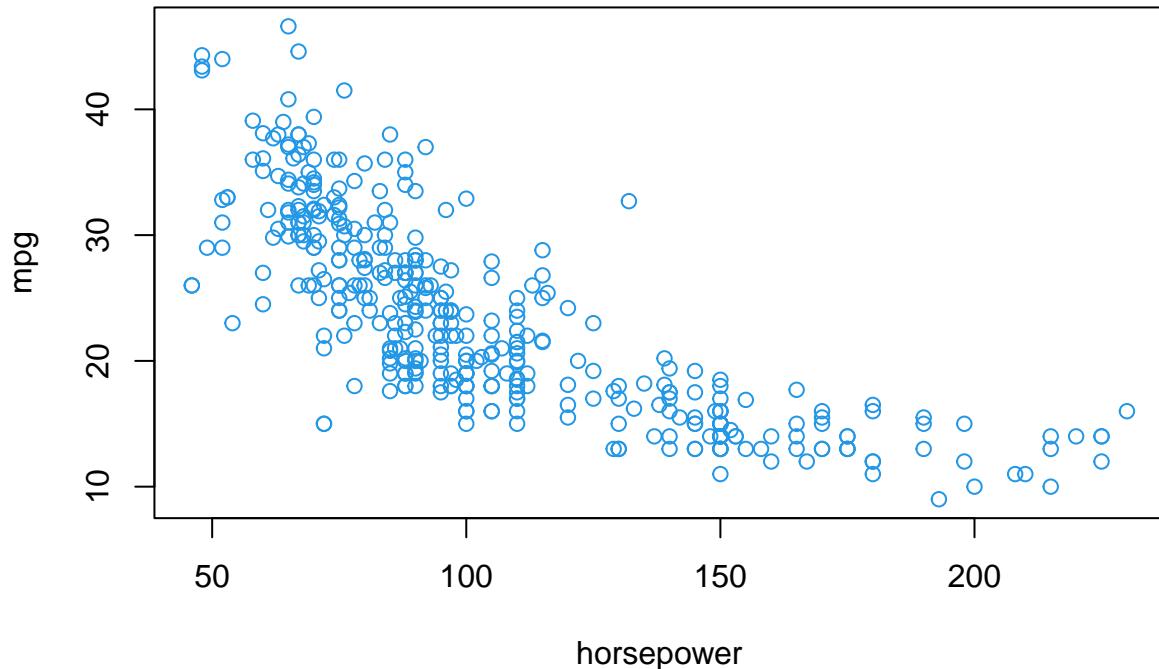
```
pairs(Auto)
```



```
pairs(~mpg + displacement + cylinders + acceleration, data=Auto)
```



```
#windows()  
plot(horsepower,mpg, col=12)
```



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

Use windows() to run the identify interactive function

```
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

```
summary(displacement)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
##	68.0	104.0	146.0	193.5	262.0	455.0

## APPLIED EXERCISE

8. This exercise relates to the College data set, which can be found in the file College.csv on the book website. It contains a number of variables for 777 different universities and colleges in the US

- (a) Use the read.csv() function to read the data into R. Call the loaded data college. Make sure that you have the directory set to the correct location for the data.

```
college = read.csv("College.csv", stringsAsFactors = T)
attach(college)
View(college)
dim(college)
```

```
## [1] 777 19
```

- (b) Look at the data using the View() function. You should notice that the first column is just the name of each university. We don't really want R to treat this as data.

```
rownames(college) = college[, 1]
college = college[, -1]
```

- (c) i. Use the summary() function to produce a numerical summary of the variables in the data set.

```
summary(college)
```

```
## Private          Apps          Accept          Enroll          Top10perc
## No :212   Min.   : 81   Min.   : 72   Min.   : 35   Min.   : 1.00
## Yes:565  1st Qu.: 776  1st Qu.: 604  1st Qu.: 242  1st Qu.:15.00
##                   Median :1558  Median :1110  Median :434   Median :23.00
##                   Mean   :3002  Mean   :2019  Mean   :780   Mean   :27.56
##                   3rd Qu.:3624  3rd Qu.:2424  3rd Qu.:902   3rd Qu.:35.00
##                   Max.   :48094 Max.   :26330 Max.   :6392   Max.   :96.00
## Top25perc        F.Undergrad      P.Undergrad      Outstate
## Min.   : 9.0   Min.   : 139   Min.   : 1.0   Min.   : 2340
## 1st Qu.: 41.0  1st Qu.: 992   1st Qu.: 95.0  1st Qu.: 7320
## Median : 54.0  Median :1707   Median :353.0  Median :9990
## Mean   : 55.8  Mean   :3700   Mean   :855.3  Mean   :10441
## 3rd Qu.: 69.0  3rd Qu.:4005   3rd Qu.:967.0  3rd Qu.:12925
## Max.   :100.0  Max.   :31643   Max.   :21836.0 Max.   :21700
## Room.Board        Books          Personal          PhD
## Min.   :1780   Min.   : 96.0  Min.   : 250   Min.   :  8.00
## 1st Qu.:3597   1st Qu.: 470.0  1st Qu.: 850   1st Qu.: 62.00
## Median :4200   Median : 500.0  Median :1200   Median : 75.00
## Mean   :4358   Mean   :549.4   Mean   :1341   Mean   : 72.66
## 3rd Qu.:5050   3rd Qu.: 600.0  3rd Qu.:1700   3rd Qu.: 85.00
## Max.   :8124   Max.   :2340.0  Max.   :6800   Max.   :103.00
## Terminal         S.F.Ratio      perc.alumni      Expend
## Min.   : 24.0   Min.   : 2.50  Min.   : 0.00  Min.   : 3186
## 1st Qu.: 71.0   1st Qu.:11.50  1st Qu.:13.00  1st Qu.: 6751
## Median : 82.0   Median :13.60  Median :21.00  Median : 8377
## Mean   : 79.7   Mean   :14.09  Mean   :22.74  Mean   : 9660
## 3rd Qu.: 92.0   3rd Qu.:16.50  3rd Qu.:31.00  3rd Qu.:10830
## Max.   :100.0   Max.   :39.80  Max.   :64.00  Max.   :56233
## Grad.Rate
## Min.   : 10.00
## 1st Qu.: 53.00
## Median : 65.00
## Mean   : 65.46
```

```

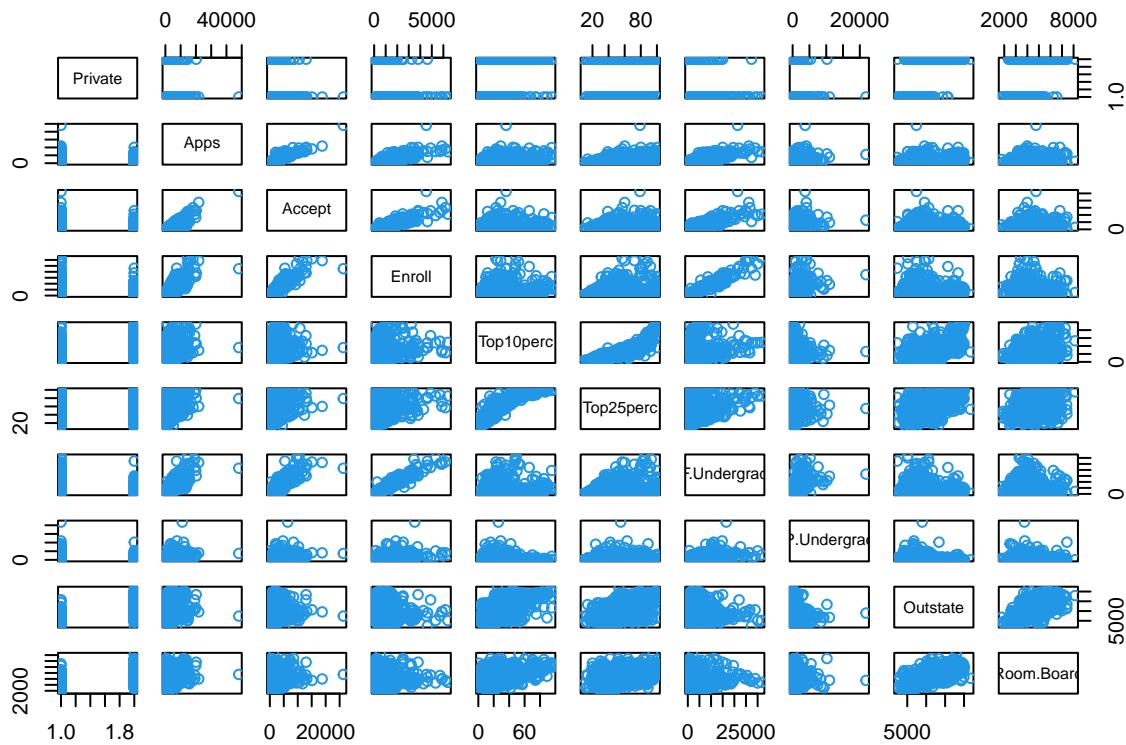
## 3rd Qu.: 78.00
## Max. :118.00
names(college)

## [1] "Private"      "Apps"        "Accept"       "Enroll"       "Top10perc"
## [6] "Top25perc"    "F.Undergrad"  "P.Undergrad"  "Outstate"     "Room.Board"
## [11] "Books"         "Personal"     "PhD"          "Terminal"     "S.F.Ratio"
## [16] "perc.alumni"   "Expend"      "Grad.Rate"

```

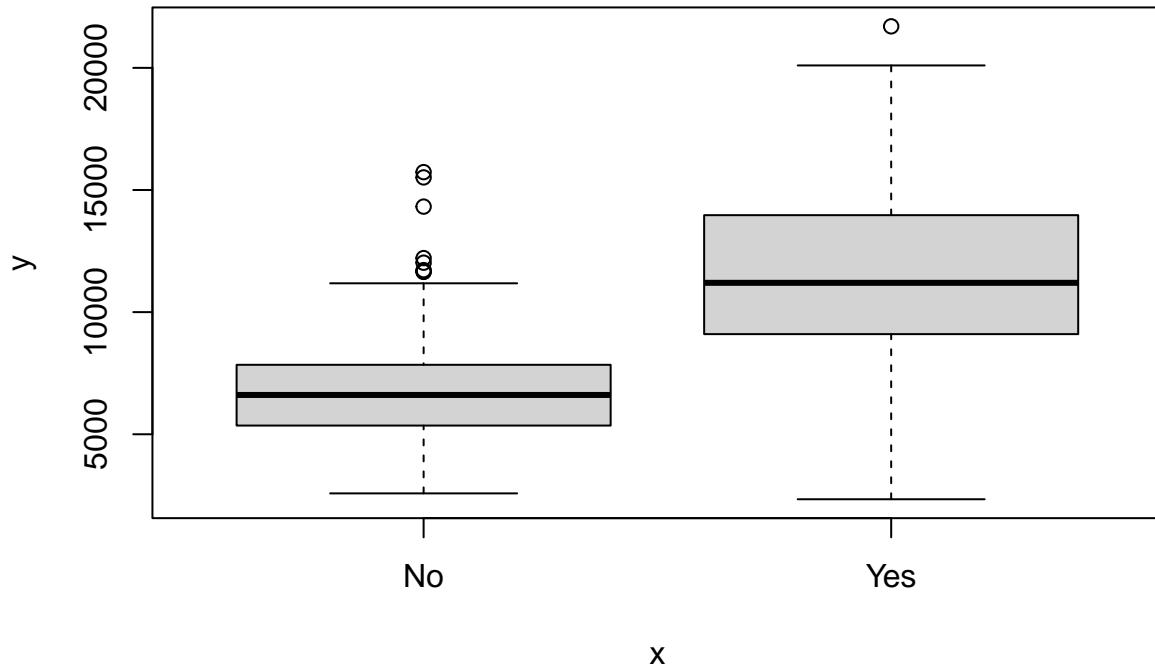
ii. Use the pairs() function to produce a scatterplot matrix of the first ten columns or variables of the data. Recall that you can reference the first ten columns of a matrix A using A[,1:10].

```
pairs(college[,1:10], col=12)
```



iii. Use the plot() function to produce side-by-side boxplots of Outstate versus Private.

```
plot(Private, Outstate)
```



Here we can see that in average Private Institutes have more out of state tuition

iv. Create a new qualitative variable, called Elite, by binning the Top10perc variable. We are going to divide universities into two groups based on whether or not the proportion of students coming from the top 10% of their high school classes exceeds 50%.

```
if (!("Elite" %in% colnames(college))) {
  Elite = rep("No", nrow(college))
  Elite[Top10perc>50] = "Yes"
  Elite=as.factor(Elite)
  college=data.frame(college,Elite)
}

head(college)
```

	Private	Apps	Accept	Enroll	Top10perc	Top25perc
## Abilene Christian University	Yes	1660	1232	721	23	52
## Adelphi University	Yes	2186	1924	512	16	29
## Adrian College	Yes	1428	1097	336	22	50
## Agnes Scott College	Yes	417	349	137	60	89
## Alaska Pacific University	Yes	193	146	55	16	44
## Albertson College	Yes	587	479	158	38	62
	F.Undergrad	P.Undergrad	Outstate	Room.Board	Books	
## Abilene Christian University	2885		537	7440	3300	450
## Adelphi University	2683		1227	12280	6450	750
## Adrian College	1036		99	11250	3750	400
## Agnes Scott College	510		63	12960	5450	450

```

## Alaska Pacific University          249      869      7560      4120      800
## Albertson College                 678       41     13500      3335      500
##                                         Personal PhD Terminal S.F.Ratio perc.alumni Expend
## Abilene Christian University    2200      70       78      18.1       12    7041
## Adelphi University                1500      29       30      12.2       16   10527
## Adrian College                   1165      53       66      12.9       30    8735
## Agnes Scott College               875       92       97      7.7        37   19016
## Alaska Pacific University        1500      76       72      11.9        2   10922
## Albertson College                  675      67       73      9.4       11    9727
##                                         Grad.Rate Elite
## Abilene Christian University     60      No
## Adelphi University                 56      No
## Adrian College                     54      No
## Agnes Scott College                59     Yes
## Alaska Pacific University         15      No
## Albertson College                  55      No

```

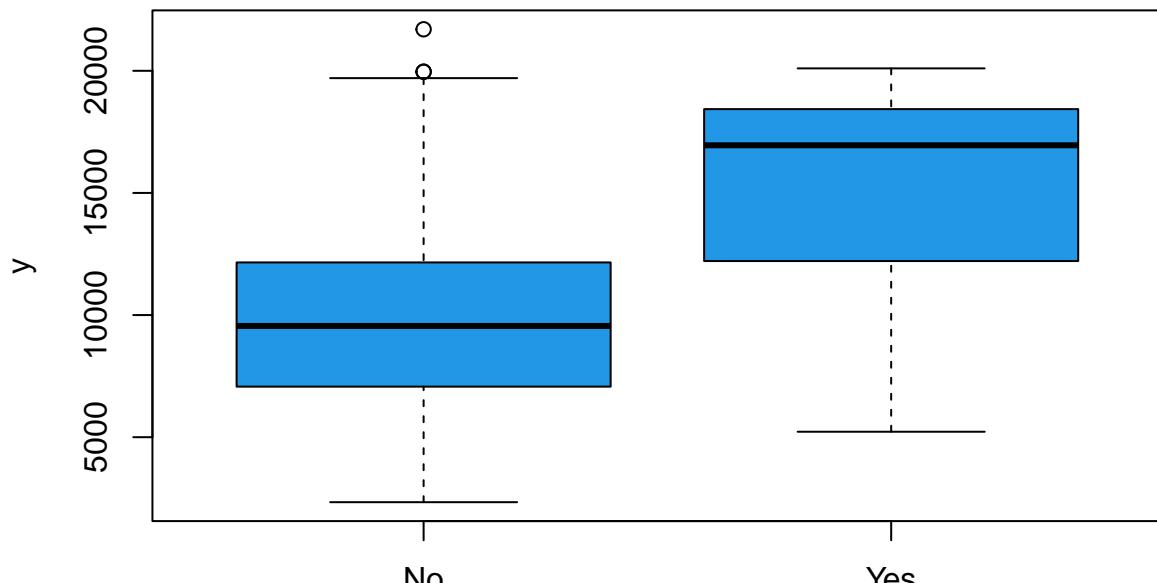
Use the summary() function to see how many elite universities there are. Now use the plot() function to produce side-by-side boxplots of Outstate versus Elite.

```
summary(Elite)
```

```

##  No Yes
## 699  78
plot(Elite,Outstate, col=12)

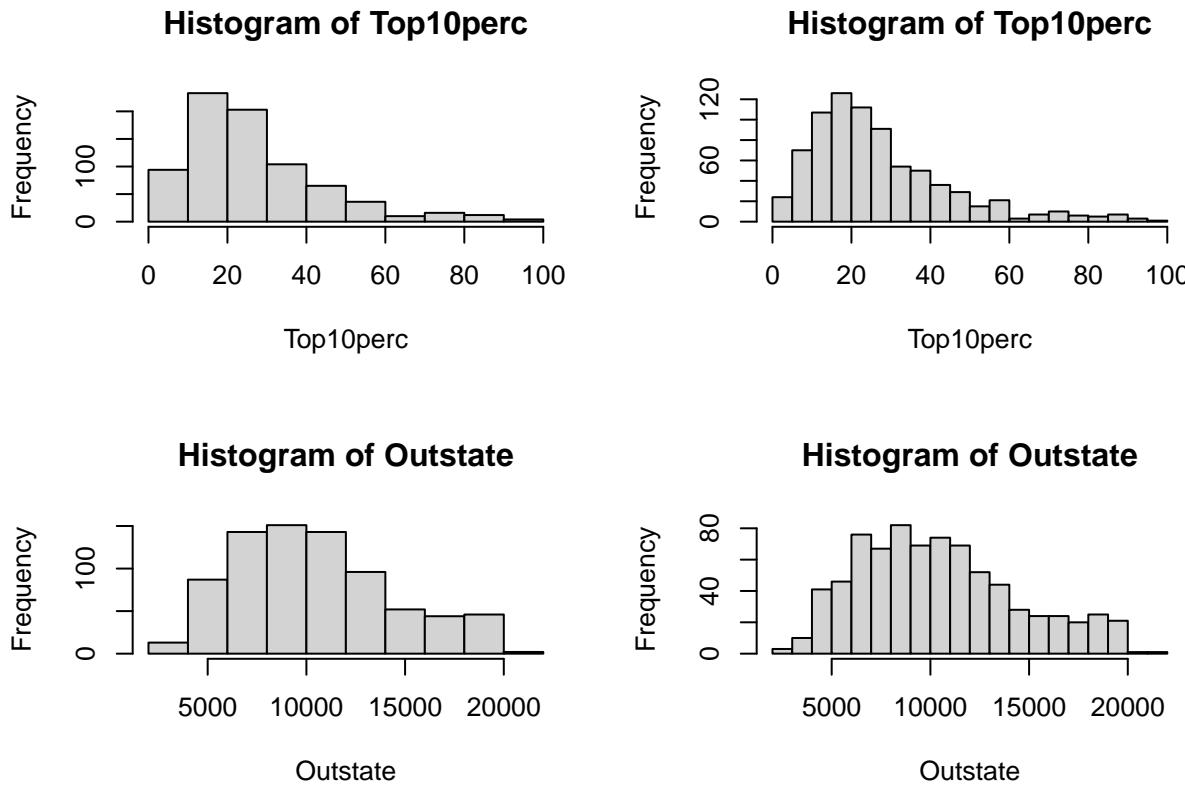
```



*So in average elite people have more out of state tuition*

v. Use the `hist()` function to produce some histograms with differing numbers of bins for a few of the quantitative variables. You may find the command `par(mfrow = c(2, 2))` useful: it will divide the print window into four regions so that four plots can be made simultaneously. Modifying the arguments to this function will divide the screen in other ways.

```
par(mfrow=c(2,2))
hist(Top10perc)
hist(Top10perc, breaks = 15)
hist(Outstate)
hist(Outstate, breaks=15)
```



vi. Continue exploring the data, and provide a brief summary of what you discover.

9. This exercise involves the Auto data set studied in the lab. Make sure that the missing values have been removed from the data. (a) Which of the predictors are quantitative, and which are qualitative?

```
head(Auto)
```

```
##   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
## 5 17          8           302        140    3449         10.5    70     1
## 6 15          8           429        198    4341         10.0    70     1
##
##                                     name
## 1 chevrolet chevelle malibu
## 2 buick skylark 320
```

```

## 3      plymouth satellite
## 4          amc rebel sst
## 5          ford torino
## 6          ford galaxie 500
Auto= na.omit(Auto)

```

*Quantitative- mpg, cylinders, displacement, horsepower, weight, acceleration,  
Qualitative - Year, Origin, name*

(b) What is the range of each quantitative predictor? You can answer this using the range() function.

```

cylinders= as.numeric(as.character(cylinders))
range(mpg)

```

```

## [1] 9.0 46.6
range(cylinders)

```

```

## [1] 3 8
range(displacement)

```

```

## [1] 68 455
range(horsepower, na.rm=T)

```

```

## [1] 46 230
range(weight)

```

```

## [1] 1613 5140
range(acceleration)

```

```

## [1] 8.0 24.8

```

(c) What is the mean and standard deviation of each quantitative predictor?

```

cat("MPG Mean:", mean(mpg), "\n")

```

```

## MPG Mean: 23.51587

```

```

cat("MPG SD:", sd(mpg), "\n")

```

```

## MPG SD: 7.825804

```

```

cat("displacement Mean:", mean(displacement), "\n")

```

```

## displacement Mean: 193.5327

```

```

cat("displacement SD:", sd(displacement), "\n")

```

```

## displacement SD: 104.3796

```

```

cat("weight Mean:", mean(weight), "\n")

```

```

## weight Mean: 2970.262

```

```

cat("weight SD:", sd(weight), "\n")

```

```

## weight SD: 847.9041

```

```

cat("horsepower Mean:", mean(horsepower), "\n")

## horsepower Mean: NA
cat("horsepower SD:", sd(horsepower), "\n")

## horsepower SD: NA
cat("acceleration Mean:", mean(acceleration), "\n")

## acceleration Mean: 15.55567
cat("acceleration SD:", sd(acceleration), "\n")

## acceleration SD: 2.749995

```

- (d) Now remove the 10th through 85th observations. What is the mean, and standard deviation of each predictor in the subset of the data that remains?

```

Autosubset = Auto[-c(10:84),]

with(Autosubset,{
  cat("MPG Mean:", mean(mpg), "\n")
  cat("MPG SD:", sd(mpg), "\n")
  cat("displacement Mean:", mean(displacement), "\n")
  cat("displacement SD:", sd(displacement), "\n")
  cat("weight Mean:", mean(weight), "\n")
  cat("weight SD:", sd(weight), "\n")
  cat("horsepower Mean:", mean(horsepower), "\n")
  cat("horsepower SD:", sd(horsepower), "\n")
  cat("acceleration Mean:", mean(acceleration), "\n")
  cat("acceleration SD:", sd(acceleration), "\n")})

```

```

## MPG Mean: 24.36845
## MPG SD: 7.880898
## displacement Mean: 187.7539
## displacement SD: 99.93949
## weight Mean: 2939.644
## weight SD: 812.6496
## horsepower Mean: 100.9558
## horsepower SD: 35.89557
## acceleration Mean: 15.7183
## acceleration SD: 2.693813

```

- e),f) are similar - Suppose that we wish to predict gas mileage (mpg) on the basis of the other variables. Do your plots suggest that any of the other variables might be useful in predicting mpg? Justify your answer.

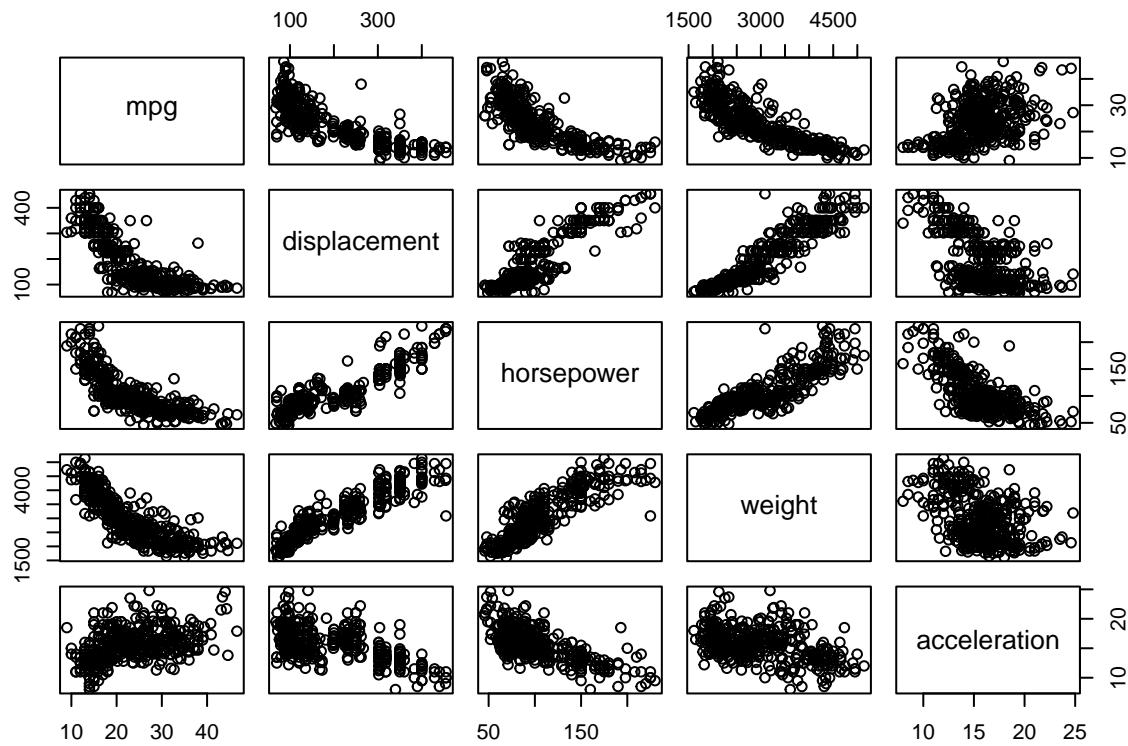
```

colnames(Auto)

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

```

```
plot(~mpg + displacement + horsepower + weight + acceleration)
```



we can see mpg is linearly corelated with displacement , horsepower and weight ; so these can be used as predictors for mpg , checking for corelation values

```
Auto$horsepower[Auto$horsepower == "?"] <- NA
Auto$horsepower <- as.numeric(Auto$horsepower)
Auto <- na.omit(Auto)
with(Auto, {
  cat("mpg vs displacement:", cor(mpg, displacement), "\n")
  cat("mpg vs horsepower:", cor(mpg, horsepower), "\n")
  cat("mpg vs weight:", cor(mpg, weight), "\n")
  cat("mpg vs acceleration:", cor(mpg, acceleration), "\n")
})
## mpg vs displacement: -0.8051269
## mpg vs horsepower: -0.7784268
## mpg vs weight: -0.8322442
## mpg vs acceleration: 0.4233285
```

as we can see apart from acceleration , rest all have high correlation values

```
with(Auto,{
  cylinders= as.factor(cylinders)
  year= as.factor(year)
  origin= as.factor(origin)
  plot(cylinders,mpg)
  plot(origin,mpg)
```

```
plot(year,mpg})
```

