Workshop 1

1 Crash course

This section is inspired by code from Everitt's book *An Introduction to Applied Multivariate Analysis with R*. This is potentially a good read if you like the applied side of multivariate analysis.

1.1 Multidimensional data basics

1.1.1 Constructing vectors

Create a vector, using the command `c()` (for concatenate)

```
x <- c(1,2,3,4)
x
```

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

Sum all elements of x:

```
sum(x)
```

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

Square all elements of x:

```
x^2
```

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

Get the third element of x:

```
x[3]
```

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

Add an extra element:

```
x <- c(x,10)
x
```

```
## [1]  1  2  3  4 10
```

Create a vector using the command `seq()` (for sequence)

```
x <- seq(from=1, to=10, by=.1)
x
```

```
##  [1]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3
## [15]  2.4  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7
## [29]  3.8  3.9  4.0  4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1
## [43]  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3  6.4  6.5
## [57]  6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9
## [71]  8.0  8.1  8.2  8.3  8.4  8.5  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3
## [85]  9.4  9.5  9.6  9.7  9.8  9.9 10.0
```

Obtain the length of x:

```
length(x)
```

```
## [1] 91
```

Create a vector using the command ‘rep()’ (for repeat)

```
x <- rep(0,times=10)
x
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

1.1.2 Constructing matrices

Library for the matrix commands.

```
library(Matrix)
```

Construct a matrix.

```
A<-matrix(c(1, 2, 3, 4, 5, 6), byrow=T, ncol=3)
print(A)
```

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

Access elements.

```
A[1,1]
```

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

Access columns.

```
A[1,]
```

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

Access rows.

```
A[,1]
```

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

Construct by column first.

```
B<-matrix(c(1, 2, 3, 4, 5, 6), byrow=F, ncol=3)
print(B)
```

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

Construct a diagonal matrix.

```
D<-diag(c(1,2,3))
```

Construct an identity matrix.

```
I<-diag(c(1,1,1))
```

Construct a matrix of all ones.

```
ONES<-matrix(rep(1,9),ncol=3)
```

1.1.3 Basic operations

Create some vectors and matrices.

```
x<-c(1, 2, 3)
y<-c(4, 5, 6)
z<-seq(1,10,by=1)
```

To make sure that R respects dimensions, turn them into matrices.

```
x<-as.matrix(x)
y<-as.matrix(y)
```

1.1.3.1 Basic operations

Transpose operations.

```
t(A)
```

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

```
t(B)
```

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

```
t(D)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    2    0
## [3,]    0    0    3
```

Element-wise operations on matrices.

```
A+B
```

```
##      [,1] [,2] [,3]
## [1,]    2    5    8
```

```
## [2,]    6    9   12
```

```
A-B
```

```
##      [,1] [,2] [,3]
## [1,]    0   -1   -2
## [2,]    2    1    0
```

```
A*B
```

```
##      [,1] [,2] [,3]
## [1,]    1    6   15
## [2,]    8   20   36
```

```
A/B
```

```
##      [,1]      [,2] [,3]
## [1,]    1 0.6666667  0.6
## [2,]    2 1.2500000  1.0
```

```
A^B
```

```
##      [,1] [,2]  [,3]
## [1,]    1    8   243
## [2,]   16  625 46656
```

Element-wise operations on vectors.

```
x+y
```

```
##      [,1]
## [1,]    5
## [2,]    7
## [3,]    9
```

```
x-y
```

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

```
x*y
```

```
##      [,1]
## [1,]    4
## [2,]   10
```

```
## [3,]   18
```

```
x/y
```

```
##      [,1]
## [1,] 0.25
## [2,] 0.40
## [3,] 0.50
```

```
y^x
```

```
##      [,1]
## [1,]    4
## [2,]   25
## [3,]  216
```

1.1.3.2 Matrix and vector operations

This would give an error message: non-conformable.

```
# A %*% B
```

Check the matrix dimension.

```
dim(A)
```

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

```
dim(B)
```

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

A correct calculation

```
A %*% t(B)
```

```
##      [,1] [,2]
## [1,]   22   28
## [2,]   49   64
```

or some alternatives

```
t(A) %*% B
```

```
##      [,1] [,2] [,3]
## [1,]    9   19   29
## [2,]   12   26   40
```

```
## [3,]   15   33   51
```

```
t(B) %*% A
```

```
##      [,1] [,2] [,3]
## [1,]    9   12   15
## [2,]   19   26   33
## [3,]   29   40   51
```

```
B %*% t(A)
```

```
##      [,1] [,2]
## [1,]   22   49
## [2,]   28   64
```

```
x %*% t(y)
```

```
##      [,1] [,2] [,3]
## [1,]    4    5    6
## [2,]    8   10   12
## [3,]   12   15   18
```

```
t(x) %*% y
```

```
##      [,1]
## [1,]   32
```

```
t(x) %*% t(A)
```

```
##      [,1] [,2]
## [1,]   14   32
```

Multiplies each column of B by a number

```
B %*% D
```

```
##      [,1] [,2] [,3]
## [1,]    1    6   15
## [2,]    2    8   18
```

Multiplies each row of B by a number

```
diag(c(3,4)) %*% B
```

```
##      [,1] [,2] [,3]
## [1,]    3    9   15
## [2,]    8   16   24
```

1.1.4 Other operations

Determinant of a matrix

```
det(D)
```

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

```
det(ONES)
```

```
## [1] 0
```

Inverse of a matrix

```
Di<-solve(D)
```

```
D %*% Di
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

```
Di %*% D
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

You can create an almost-singular matrix `(I+N)` by choosing small variance for the noise matrix `N` and see what happens with the inverse.

```
N<-matrix(rnorm(9,sd=10^-1),3,3)
Ii<-solve(ONES+N)
(ONES+N)%*%Ii
```

```
##              [,1]          [,2]         [,3]
## [1,] 1.000000e+00  1.970949e-17 8.881784e-16
## [2,] 4.957927e-16  1.000000e+00 0.000000e+00
## [3,] 0.000000e+00 -2.220446e-16 1.000000e+00
```

```
Ii%*%(I+N)
```

```
##           [,1]      [,2]      [,3]
## [1,] -4.910364 -3.256198  8.405292
## [2,] -1.549238  4.504265 -2.211232
## [3,]  6.238835 -1.057717 -5.201819
```

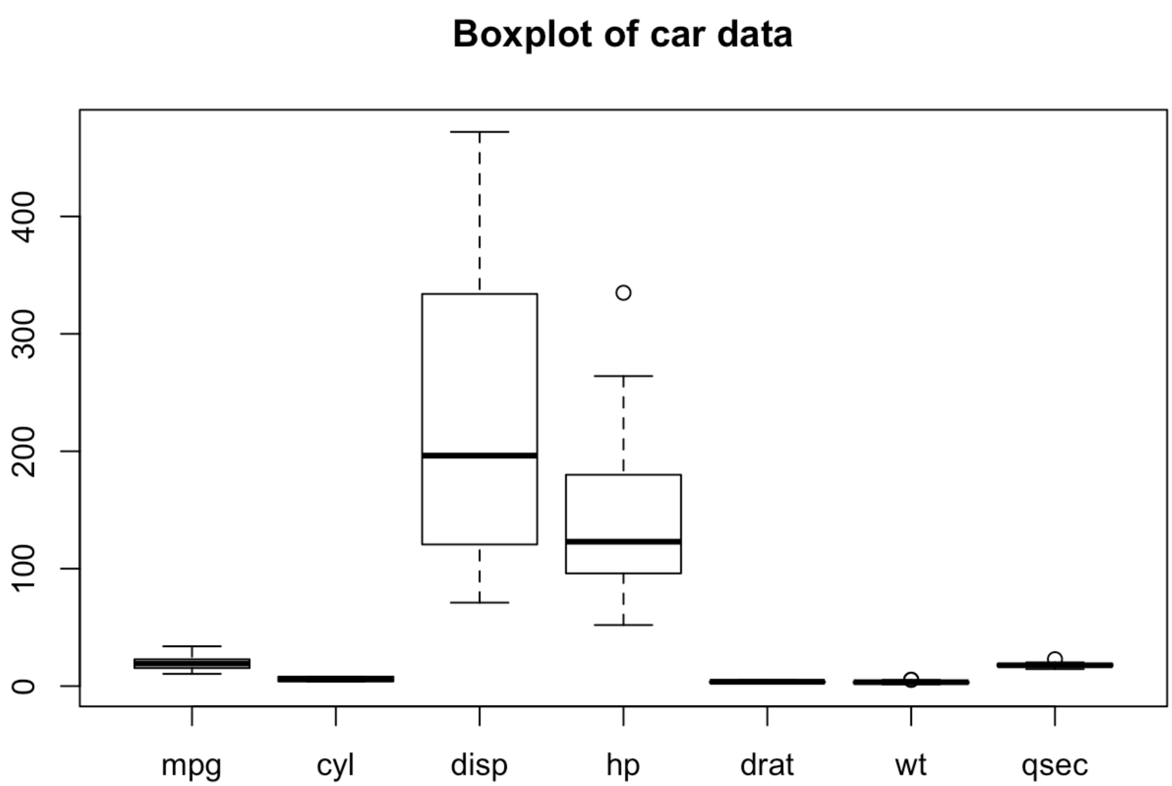
1.2 Plotting multidimensional data

Visualisation is very important when exploring a new data set. Here are some useful ways to look at multi-dimensional data.

1.2.1 Box plot

Standard boxplot.

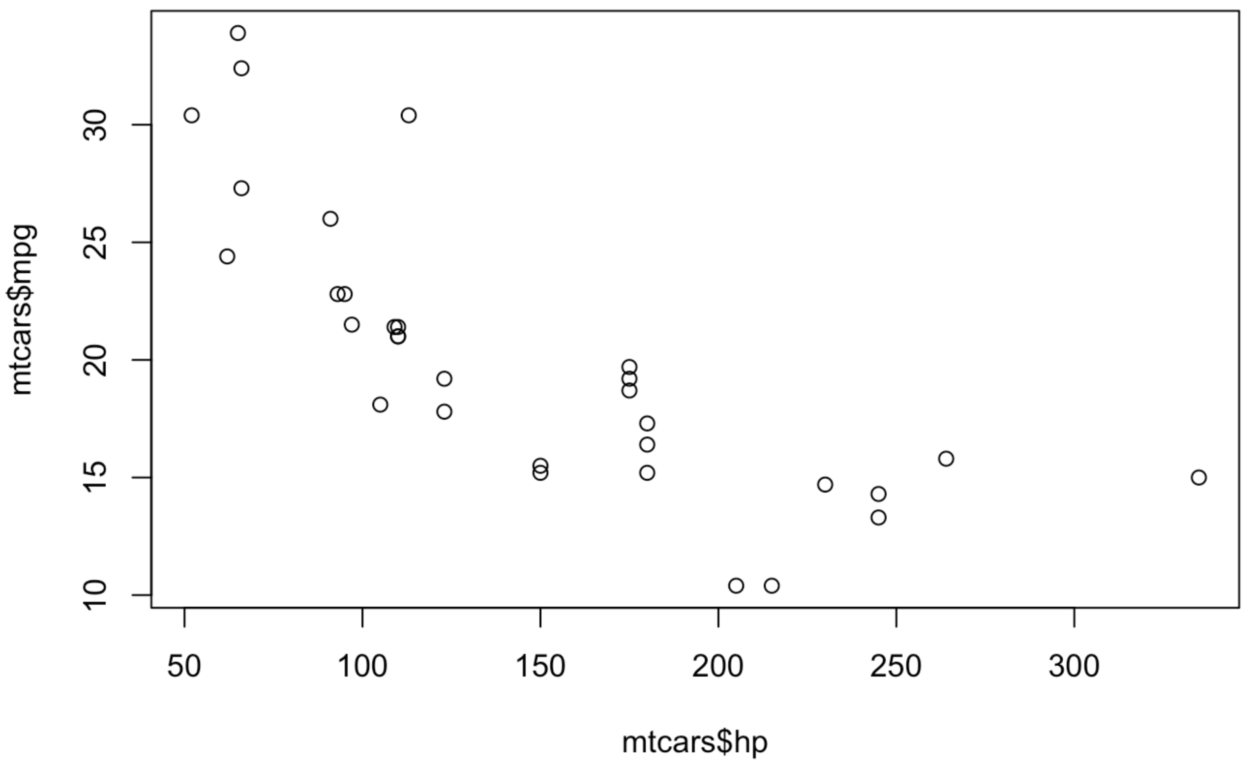
```
boxplot(mtcars[,1:7], main="Boxplot of car data")
```



1.2.2 Scatter plot

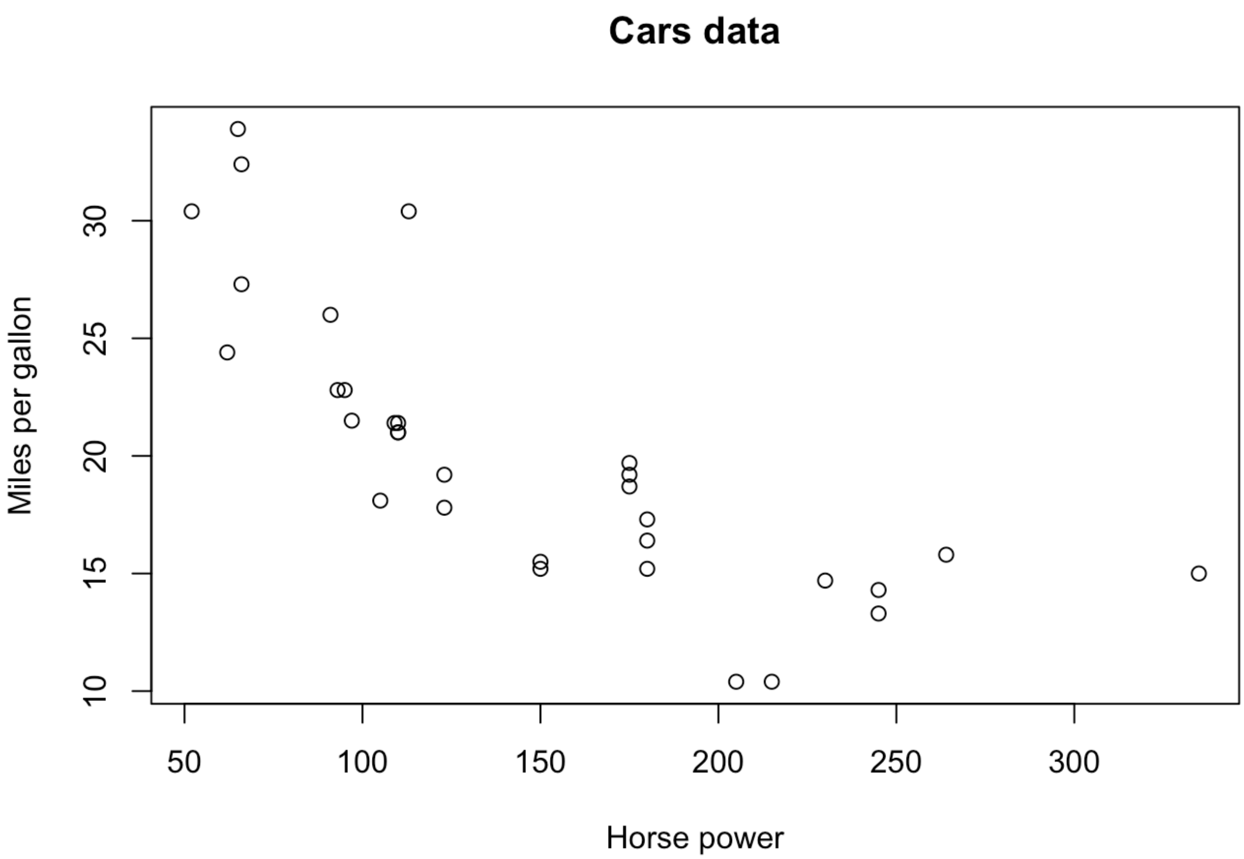
Bivariate scatter plot.

```
plot(mtcars$hp, mtcars$mpg)
```



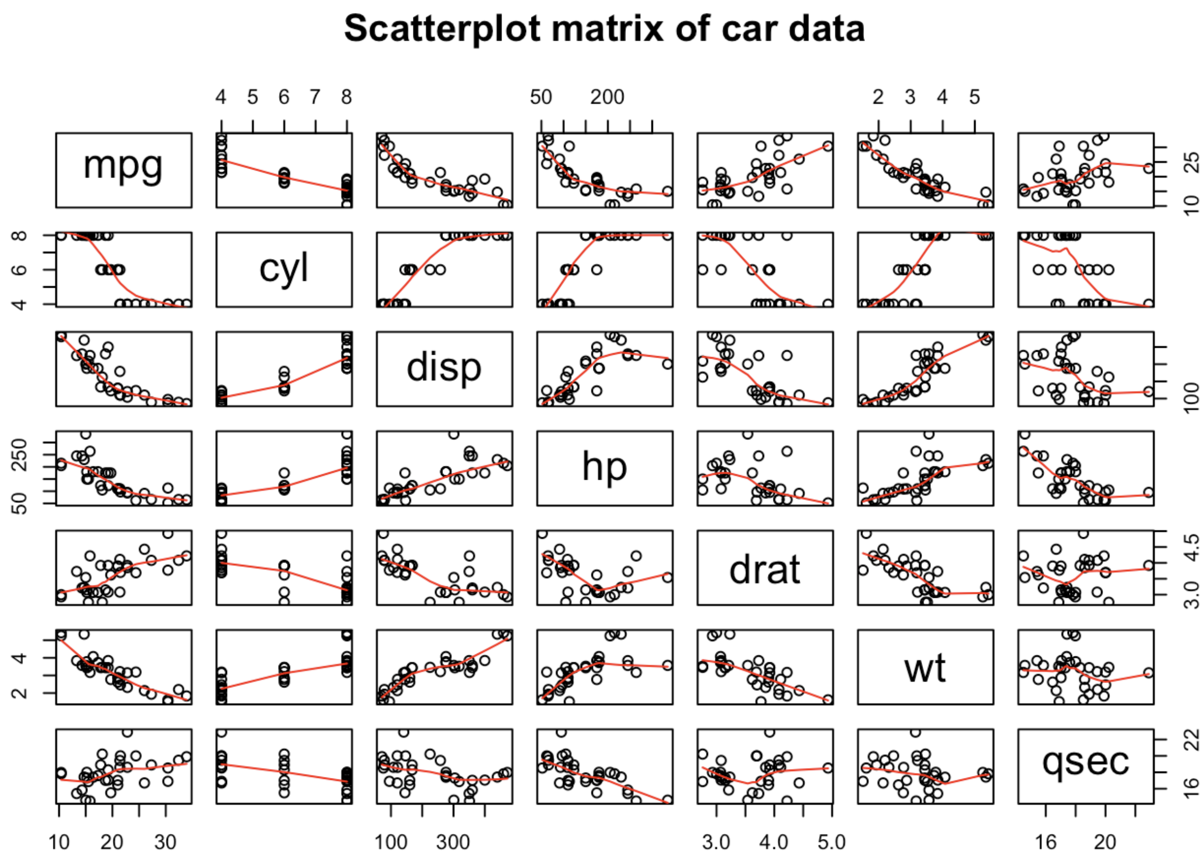
Add labels along the axes

```
plot(mtcars$hp, mtcars$mpg, xlab="Horse power",
     ylab="Miles per gallon", main="Cars data")
```



All possible bivariate scatter plots.

```
pairs(mtcars[,1:7], panel=panel.smooth,
      main="Scatterplot matrix of car data")
```



We can make a nicer version but we need to create two helper functions first. This one puts histograms on the diagonal.

```
panel.hist <- function(x, ...) {
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(usr[1:2], 0, 1.5) )
  h <- hist(x, plot = FALSE)
  breaks <- h$breaks; nB <- length(breaks)
  y <- h$counts; y <- y/max(y)
  rect(breaks[-nB], 0, breaks[-1], y,
       col = "cyan", ...)
}
```

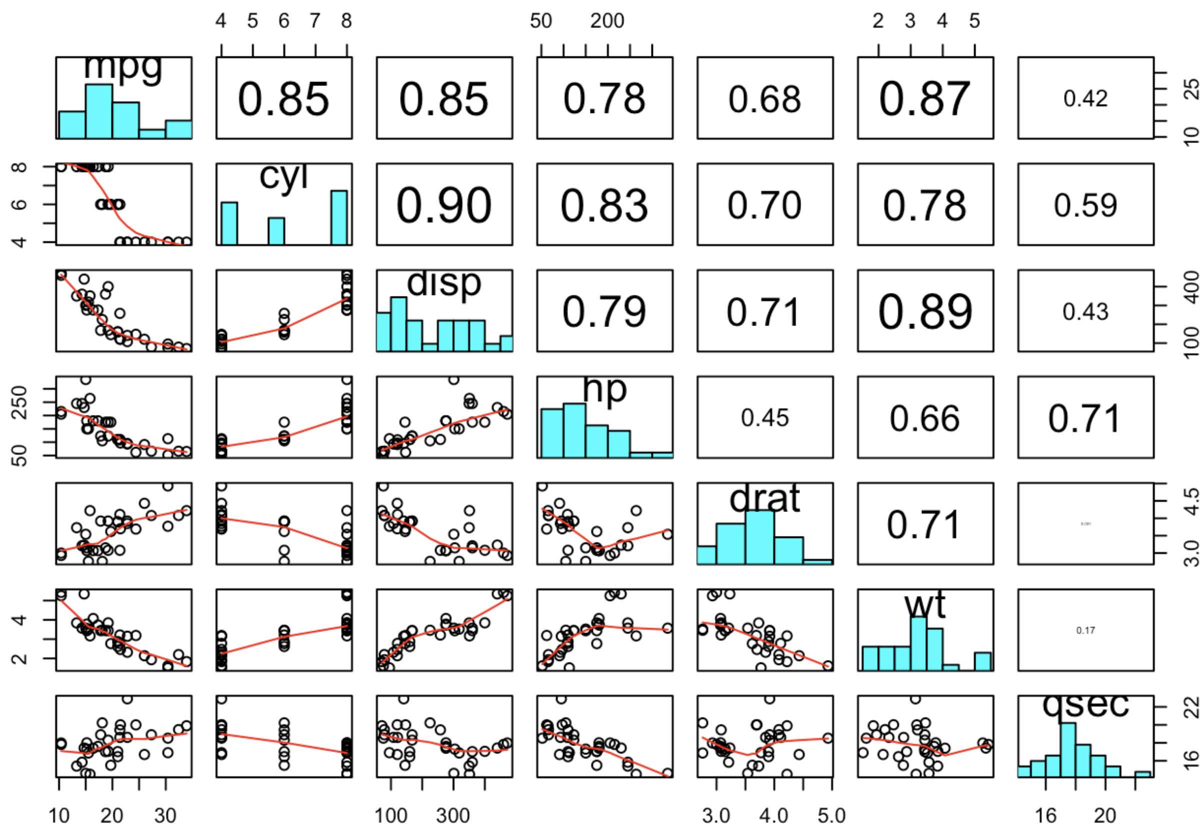
This one puts (absolute) correlations on the upper panels, with size proportional to the correlations.

```
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...) {
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
```

```
  text(0.5, 0.5, txt, cex = cex.cor * r)
}
```

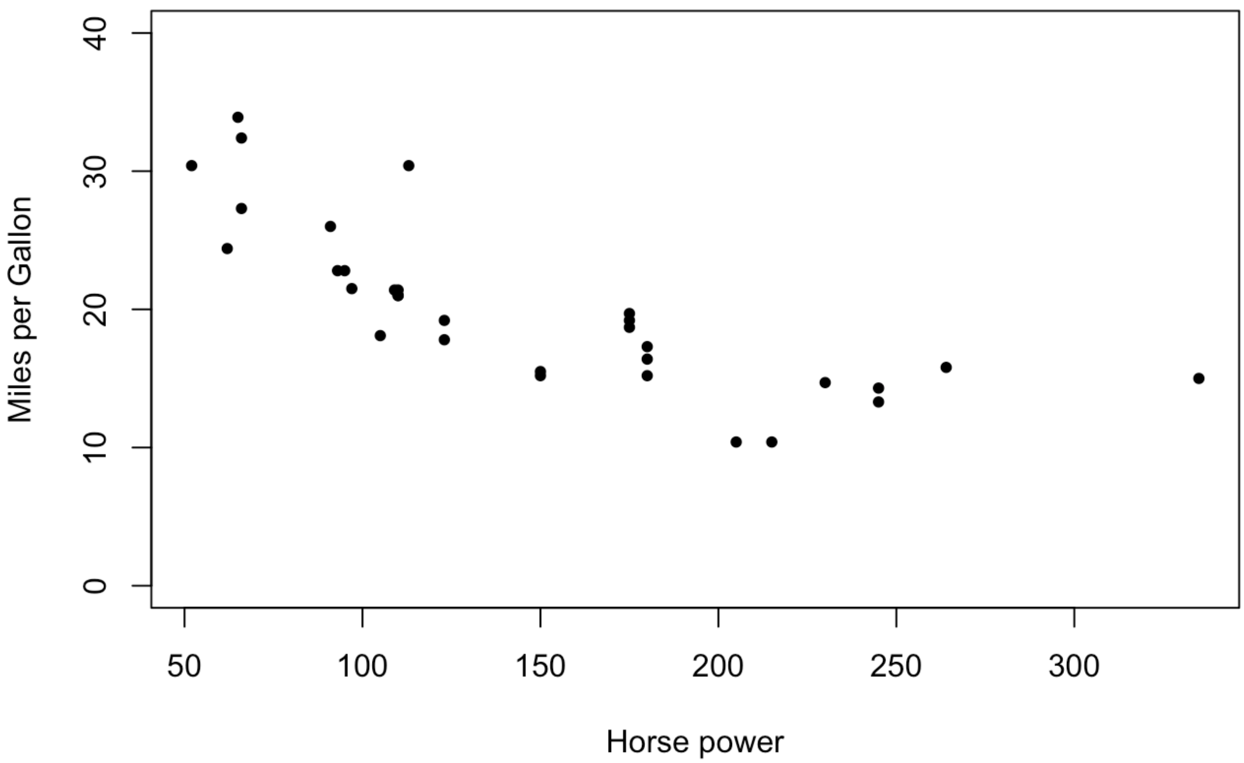
Now let's do it.

```
pairs(mtcars[,1:7], panel=panel.smooth,
      upper.panel=panel.cor, diag.panel=panel.hist)
```



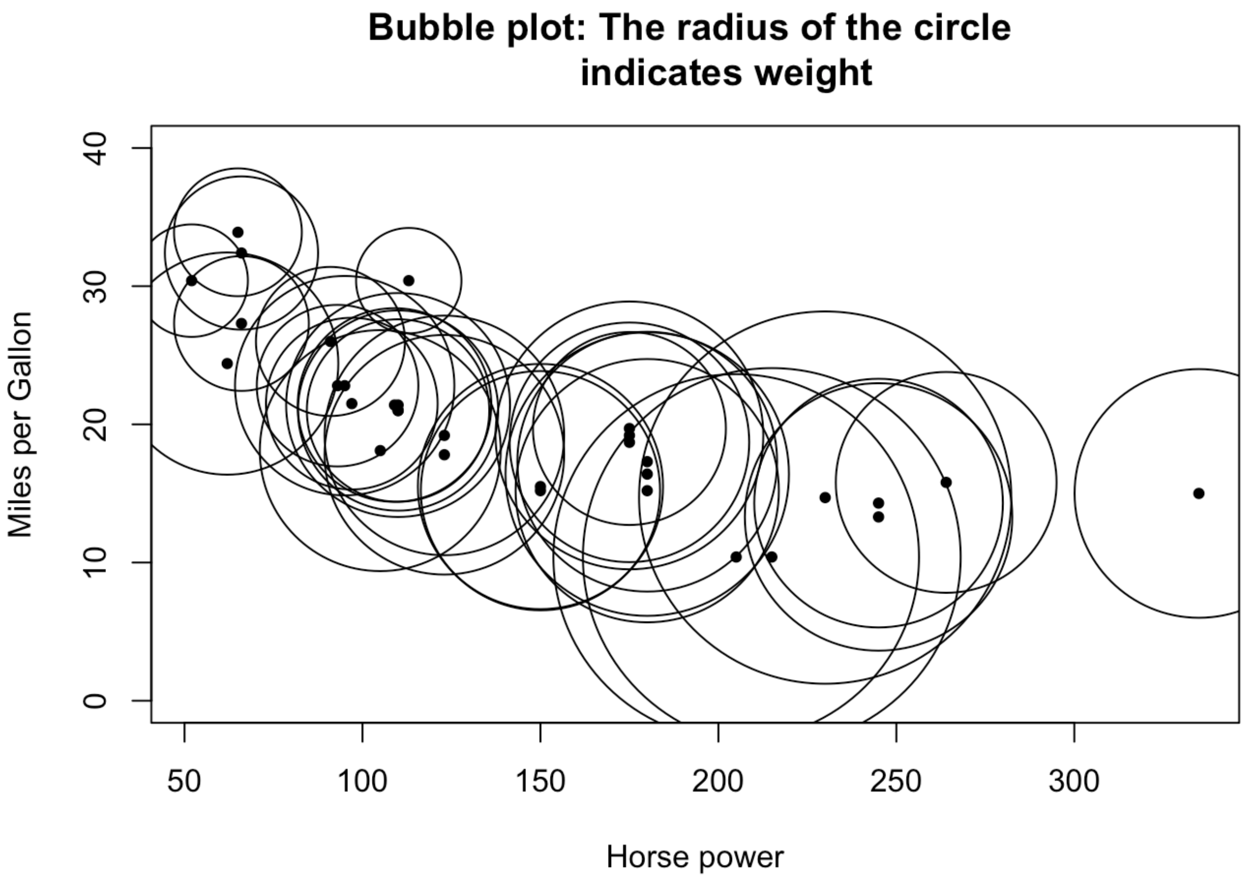
1.2.3 Bubble plots

```
plot(mtcars$hp, mtcars$mpg, pch = 20,
     xlab = "Horse power", ylab = "Miles per Gallon",
     ylim=c(0,40))
```



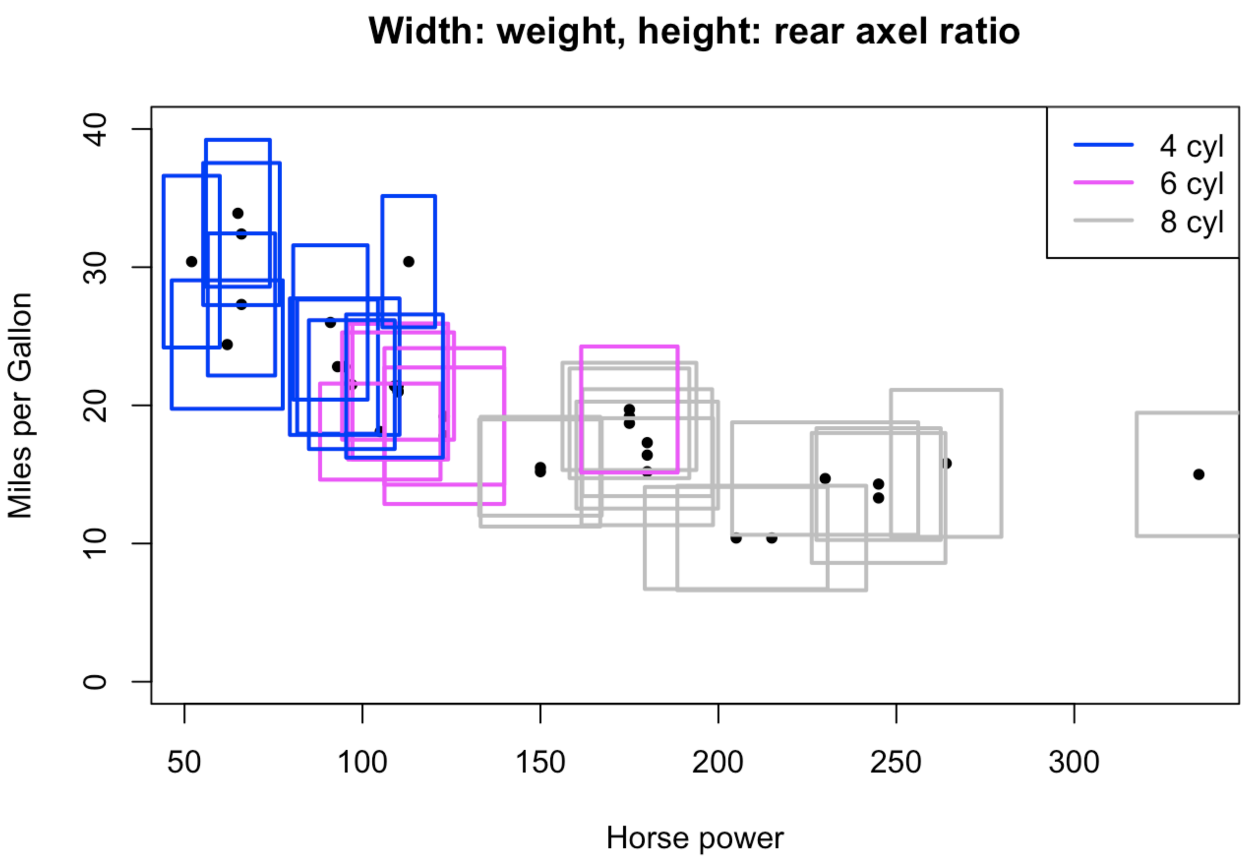
We can add a third variable by using circles.

```
plot(mtcars$hp, mtcars$mpg, pch = 20,
     xlab = "Horse power", ylab = "Miles per Gallon",
     ylim=c(0,40))
symbols(mtcars$hp, mtcars$mpg, circles = mtcars$wt,
        add = TRUE)
title("Bubble plot: The radius of the circle
      indicates weight")
```



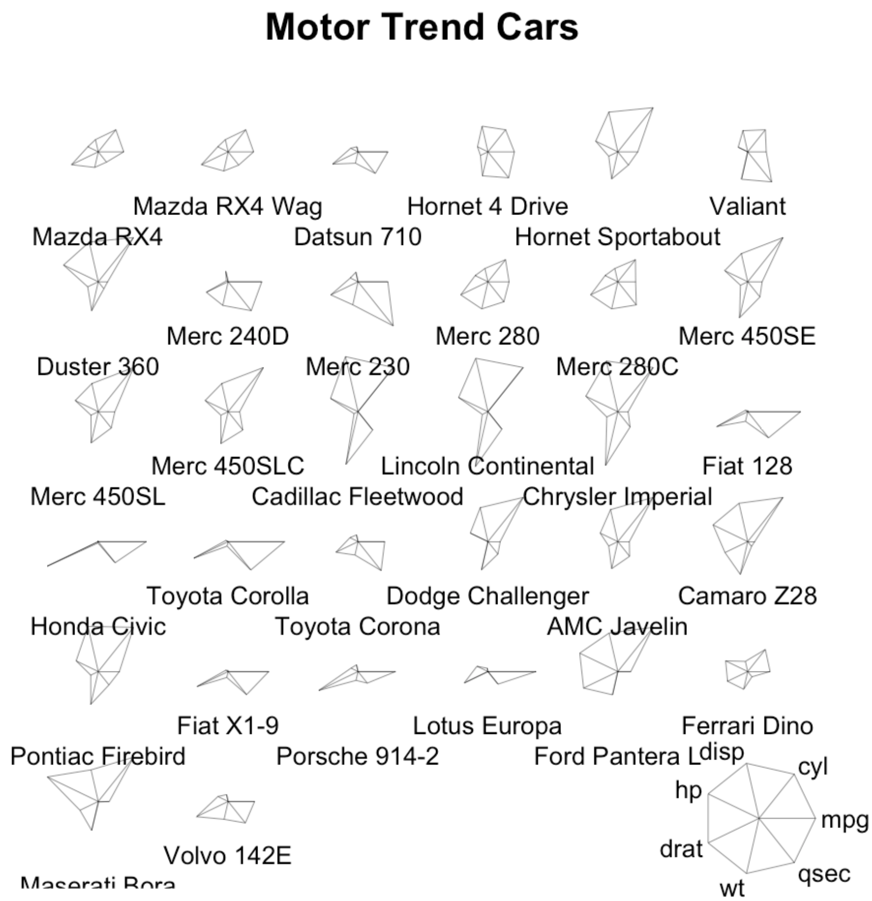
We can add even more variables with rectangles and colors:

```
plot(mtcars$hp, mtcars$mpg, pch = 20,
     xlab = "Horse power", ylab = "Miles per Gallon",
     ylim=c(0,40))
symbols(mtcars$hp, mtcars$mpg,
        rectangles = cbind(mtcars$wt,mtcars$drat),
        fg=mtcars$cyl, lwd=2, add = TRUE)
title("Width: weight, height: rear axel ratio")
legend("topright", c("4 cyl", "6 cyl", "8 cyl"),
       lty=1, col=c(4,6,8), lwd=2)
```

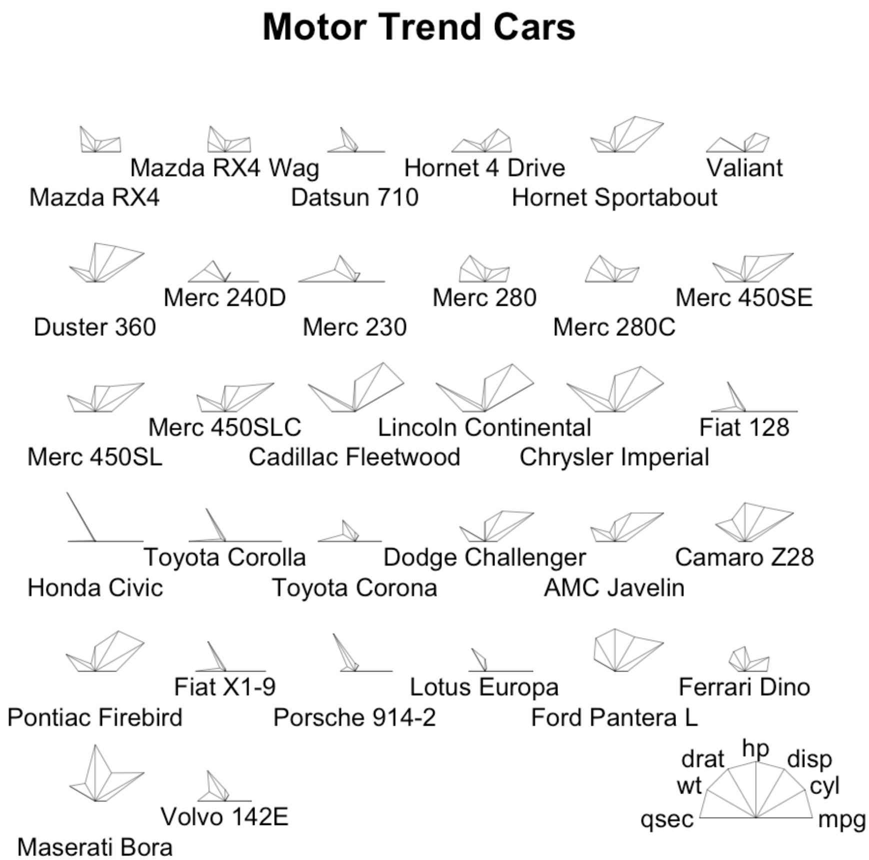


1.2.4 Star plots

```
stars(mtcars[, 1:7], key.loc = c(14, 2), scale=T,
      main = "Motor Trend Cars")
```

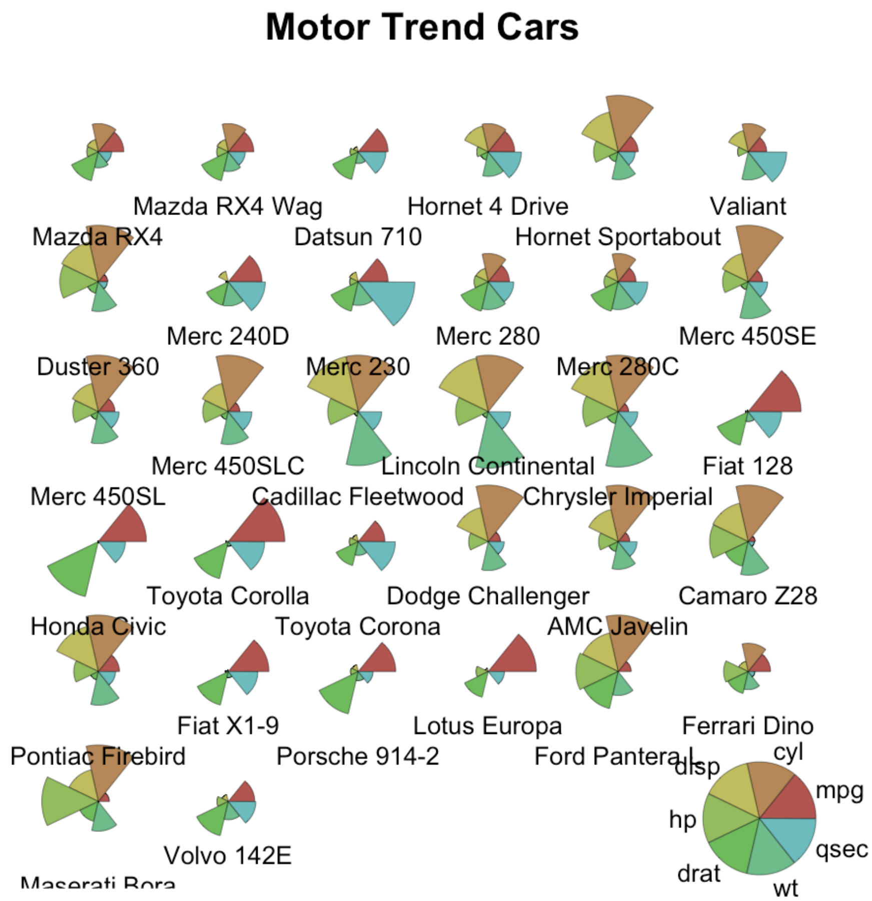



```
stars(mtcars[, 1:7], key.loc = c(14, 2), scale=T,
      main = "Motor Trend Cars", full = FALSE)
```

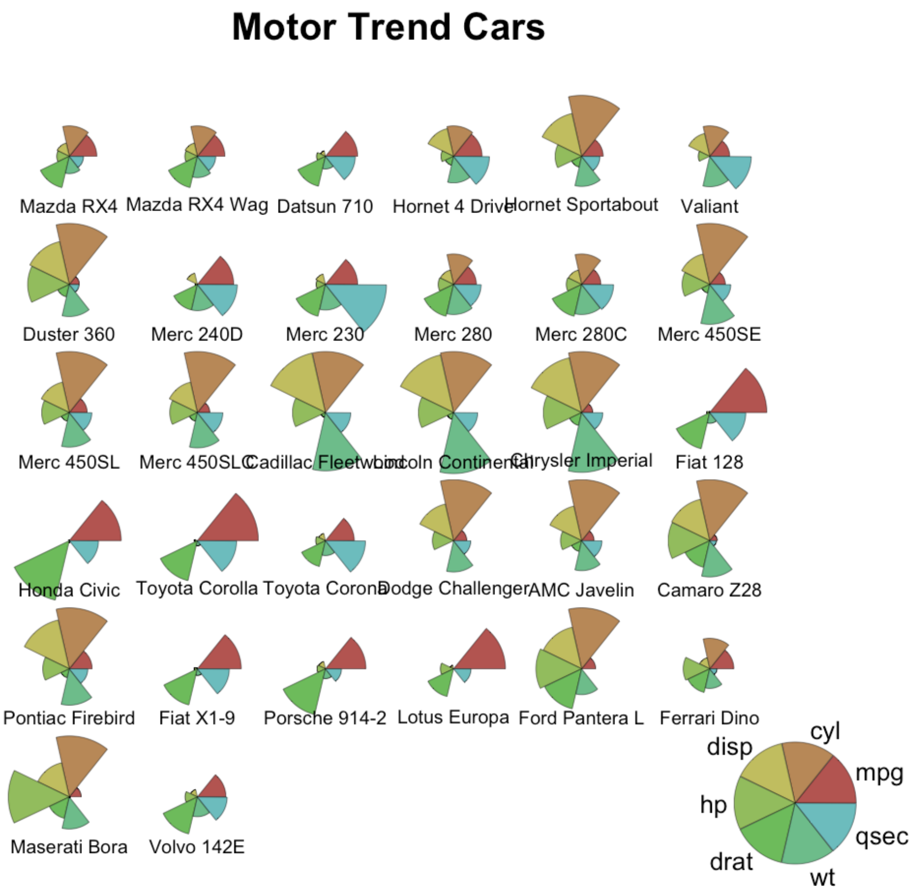


Better approach, segment plot with colors.

```
palette(rainbow(12, s = 0.6, v = 0.75))
stars(mtcars[, 1:7], key.loc = c(14,2), scale=T,
      main = "Motor Trend Cars",
      draw.segment=TRUE)
# with more control over position of labels:
loc <- stars(mtcars[, 1:7], key.loc = c(14,2), scale=T,
       main = "Motor Trend Cars",
       draw.segment=TRUE)
```




```
loc <- stars(mtcars[, 1:7], key.loc = c(14,2), scale=T,
      labels=NULL, main = "Motor Trend Cars",
      draw.segment=TRUE)
# loc contains the centers of the segment plots
# write name of car in the middle of each segment plot
text(loc[,1],loc[,2]-.8,
     row.names(mtcars),
     col = "black", cex = 0.6, xpd = TRUE)
```

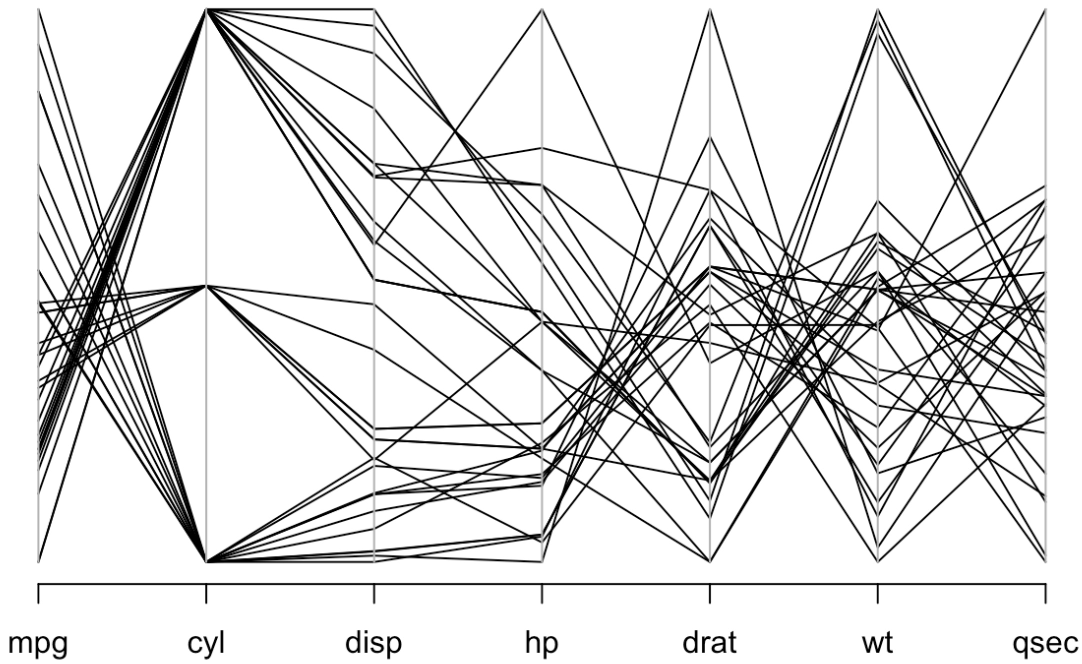



```
palette("default") # set colors back to default
```

1.2.5 Parallel coordinates plot

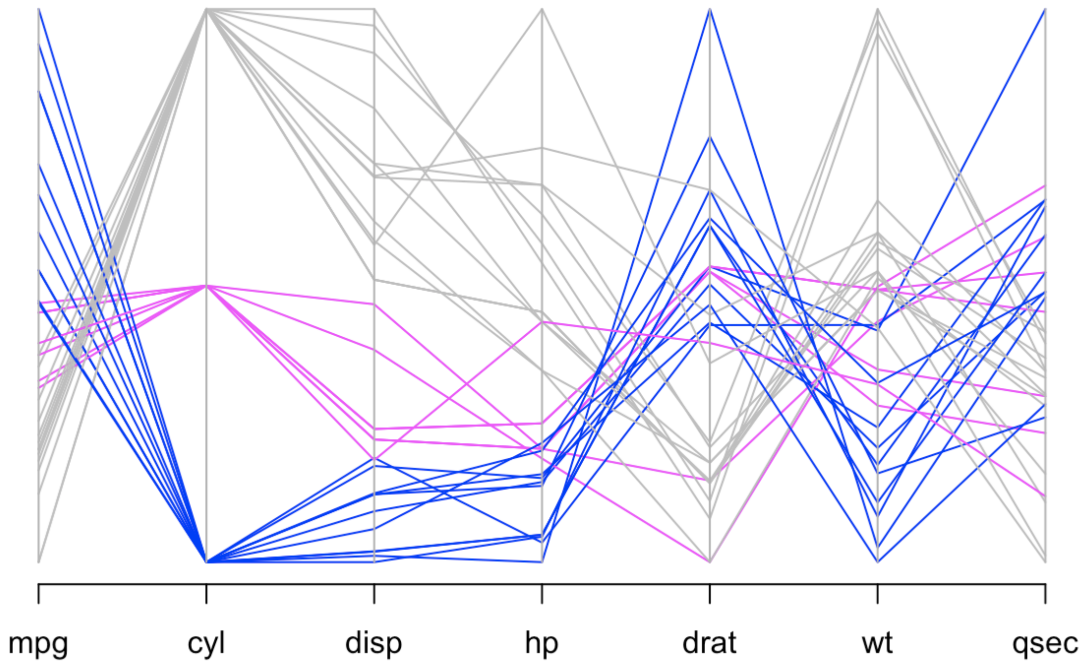
This is one of my favourites for high-dimensional data.

```
library(MASS)
parcoord(mtcars[,1:7])
```



Color can help.

```
parcoord(mtcars[,1:7], col=mtcars$cyl)
```



1.3 Random vectors and matrices

1.3.1 Random matrix

```
x<-matrix(rnorm(6), ncol=2)
x
```

```
##            [,1]       [,2]
## [1,]  2.4868938 -0.5839087
## [2,] -0.7161627  0.1765573
## [3,]  0.1950095 -0.4694244
```

Notice that `mean(x)` DOES NOT produce what we want.

```
mean(x)
```

```
## [1] 0.1814941
```

Empirical mean.

```
n<-dim(x)[1]
ones<-matrix(rep(1,n),ncol=1)
mu<-t(x) %*% ones / n
print(mu)
```

```
##            [,1]
## [1,]  0.6552469
## [2,] -0.2922586
```

1.3.2 Variance and standard deviation of a vector

```
x
```

```
##            [,1]       [,2]
## [1,]  2.4868938 -0.5839087
## [2,] -0.7161627  0.1765573
## [3,]  0.1950095 -0.4694244
```

```
var(x[,1])
```

```
## [1] 2.723757
```

```
var(x[,2])
```

```
## [1] 0.1681179
```

```
sd(x[,1])
```

```
## [1] 1.650381
```

```
sd(x[,2])
```

```
## [1] 0.4100219
```

```
# covariance
var(x[,1], x[,2])
```

```
## [1] -0.5478001
```

Variance-covariance matrix.

```
var(x)
```

```
##            [,1]       [,2]
## [1,]  2.7237566 -0.5478001
## [2,] -0.5478001  0.1681179
```

Correlation matrix.

```
cor(x)
```

```
##            [,1]       [,2]
## [1,]  1.0000000 -0.8095263
## [2,] -0.8095263  1.0000000
```

1.3.3 Sample variance-covariance

3x3 matrix of 1s.

```
ones %*% t(ones)
```

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

Identity matrix.

```
diag(3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

Matrix computation of S (unbiased)

```
(1/(n-1)) * t(x) %*% (diag(3)-(1/n)*ones %*% t(ones)) %*% x
```

```
##            [,1]       [,2]
## [1,]  2.7237566 -0.5478001
## [2,] -0.5478001  0.1681179
```

Produces the same result

```
var(x)
```

```
##            [,1]       [,2]
## [1,]  2.7237566 -0.5478001
## [2,] -0.5478001  0.1681179
```

2 Salient features of Big Data

2.1 Outliers in higher dimensions are not obvious

2.1.1 Outliers in univariate case

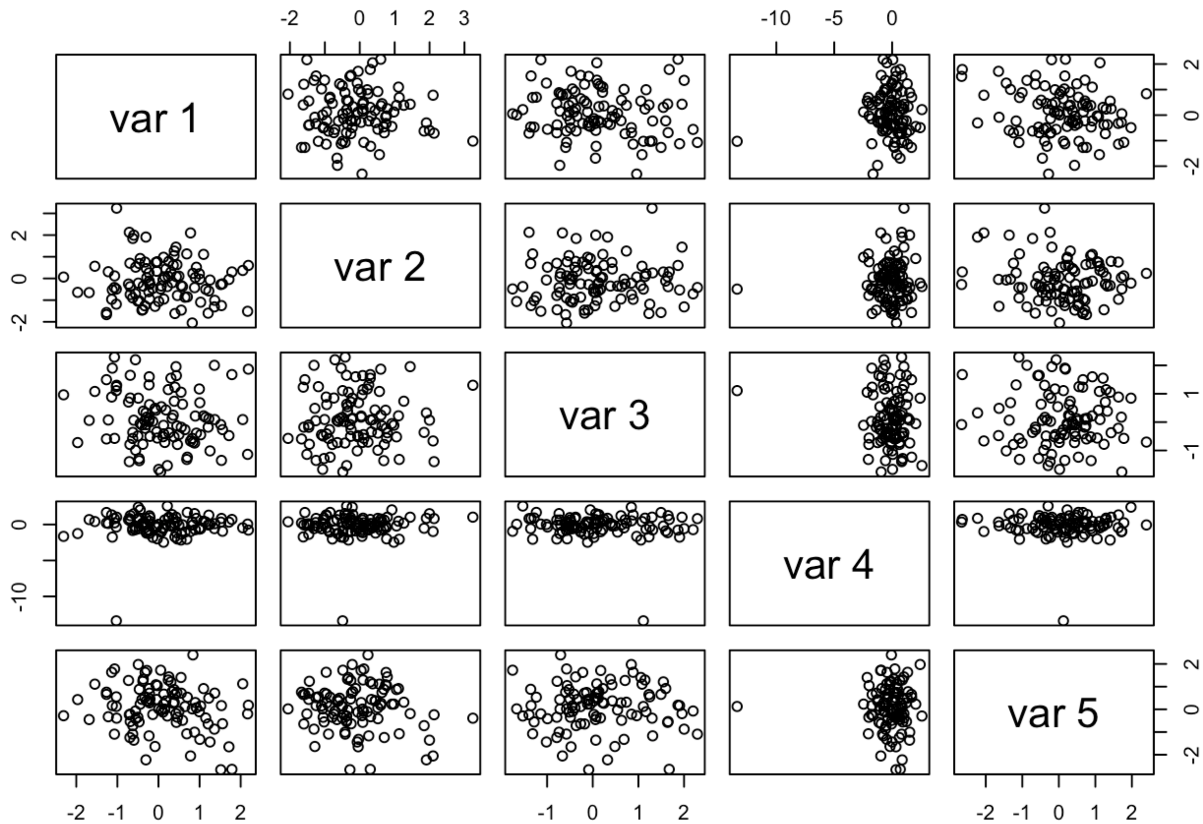
```
set.seed(123)
dat <- matrix(rnorm(5*100),100,5)
summary(dat)
```

```
##        V1                  V2                 V3
##  Min.   :-2.30917   Min.   :-2.0532   Min.   :-1.75653
##  1st Qu.:-0.49385   1st Qu.:-0.8011   1st Qu.:-0.53131
##  Median : 0.06176   Median :-0.2258   Median : 0.03591
##  Mean   : 0.09041   Mean   :-0.1075   Mean   : 0.12047
##  3rd Qu.: 0.69182   3rd Qu.: 0.4678   3rd Qu.: 0.76363
##  Max.   : 2.18733   Max.   : 3.2410   Max.   : 2.29308
##        V4                   V5
##  Min.   :-2.465898   Min.   :-2.6609
##  1st Qu.:-0.729376   1st Qu.:-0.3964
##  Median :-0.003509   Median : 0.1651
##  Mean   :-0.036223   Mean   : 0.1059
##  3rd Qu.: 0.688690   3rd Qu.: 0.7216
##  Max.   : 2.571458   Max.   : 2.3975
```

```
dat[23,4] <- dat[23,4] * 10
summary(dat)
```

```
##        V1                  V2                 V3
##  Min.   :-2.30917   Min.   :-2.0532   Min.   :-1.75653
##  1st Qu.:-0.49385   1st Qu.:-0.8011   1st Qu.:-0.53131
##  Median : 0.06176   Median :-0.2258   Median : 0.03591
##  Mean   : 0.09041   Mean   :-0.1075   Mean   : 0.12047
##  3rd Qu.: 0.69182   3rd Qu.: 0.4678   3rd Qu.: 0.76363
##  Max.   : 2.18733   Max.   : 3.2410   Max.   : 2.29308
##        V4                    V5
##  Min.   :-13.387743   Min.   :-2.6609
##  1st Qu.: -0.729376   1st Qu.:-0.3964
##  Median : -0.003509   Median : 0.1651
##  Mean   : -0.156713   Mean   : 0.1059
##  3rd Qu.:  0.688690   3rd Qu.: 0.7216
##  Max.   :  2.571458   Max.   : 2.3975
```

```
pairs(dat)
```



Extract its location

```
which.min(dat[,4])
```

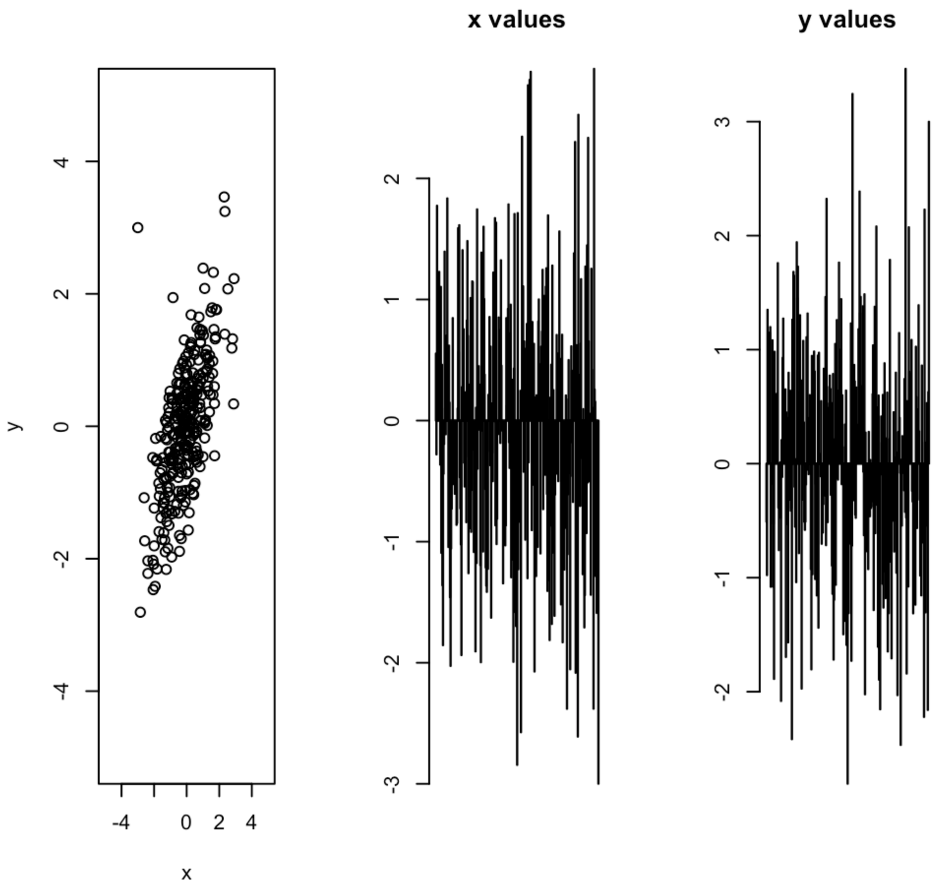
```
## [1] 23
```

2.1.2 Multivariate outliers

```
load(file = "simpleExample.rda")
```

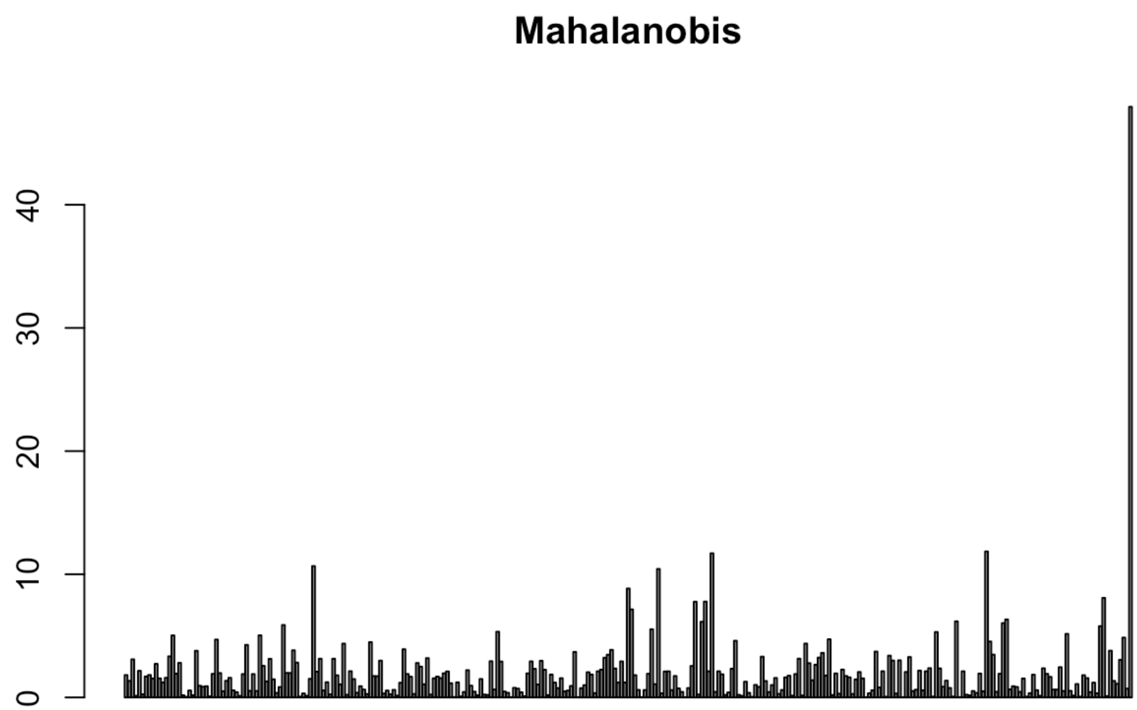
There are no clear univariate outliers here

```
par(mfrow=c(1,4))
plot(dat, xlim = c(-5,5), ylim = c(-5,5))
barplot(dat[,1], main="x values")
barplot(dat[,2], main="y values")
```



Using Mahalanobis distance

```
d <- mahalanobis(dat, colMeans(dat), cov(dat))
barplot(d, main="Mahalanobis")
```

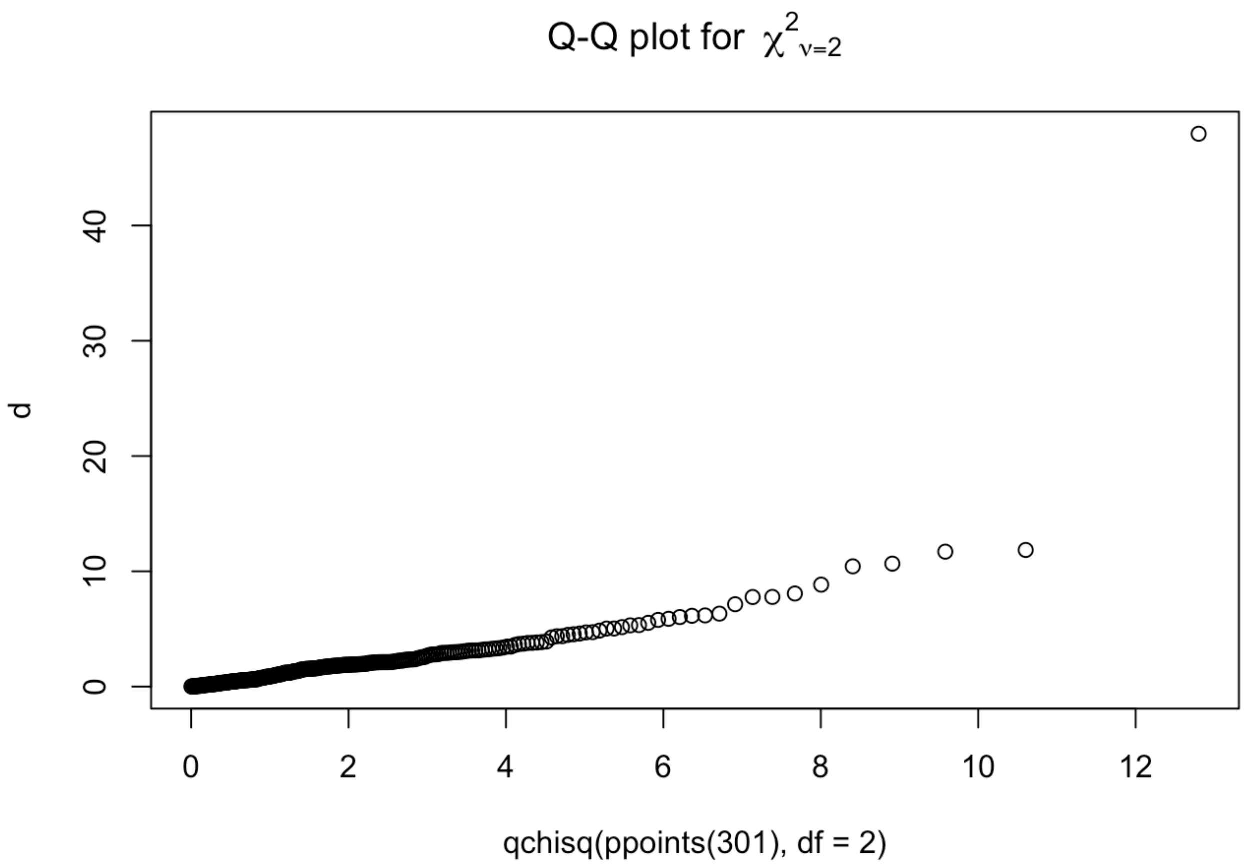



```
which.max(d)
```

```
## [1] 301
```

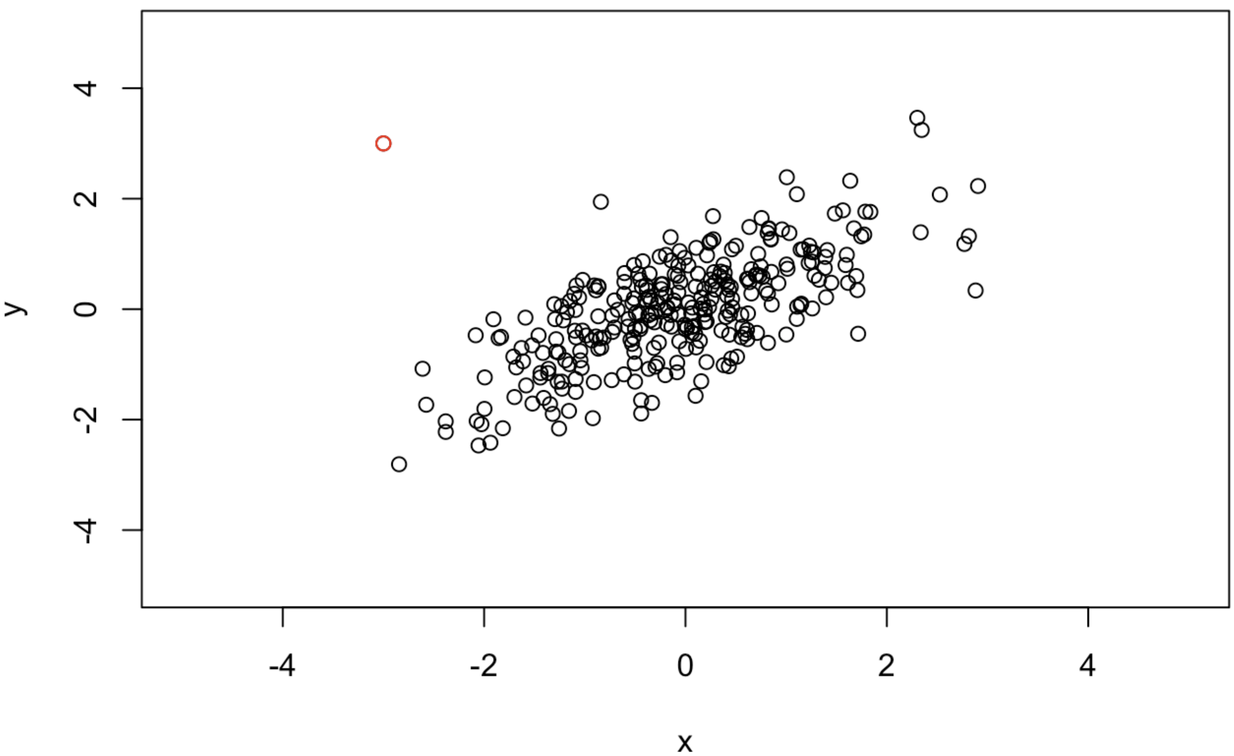
Create chi-squared QQ-plot.

```
par(mfrow=c(1,1))
qqplot(qchisq(ppoints(301), df = 2), d,
       main = expression("Q-Q plot for" ~~ {chi^2}[nu == 2]))
```



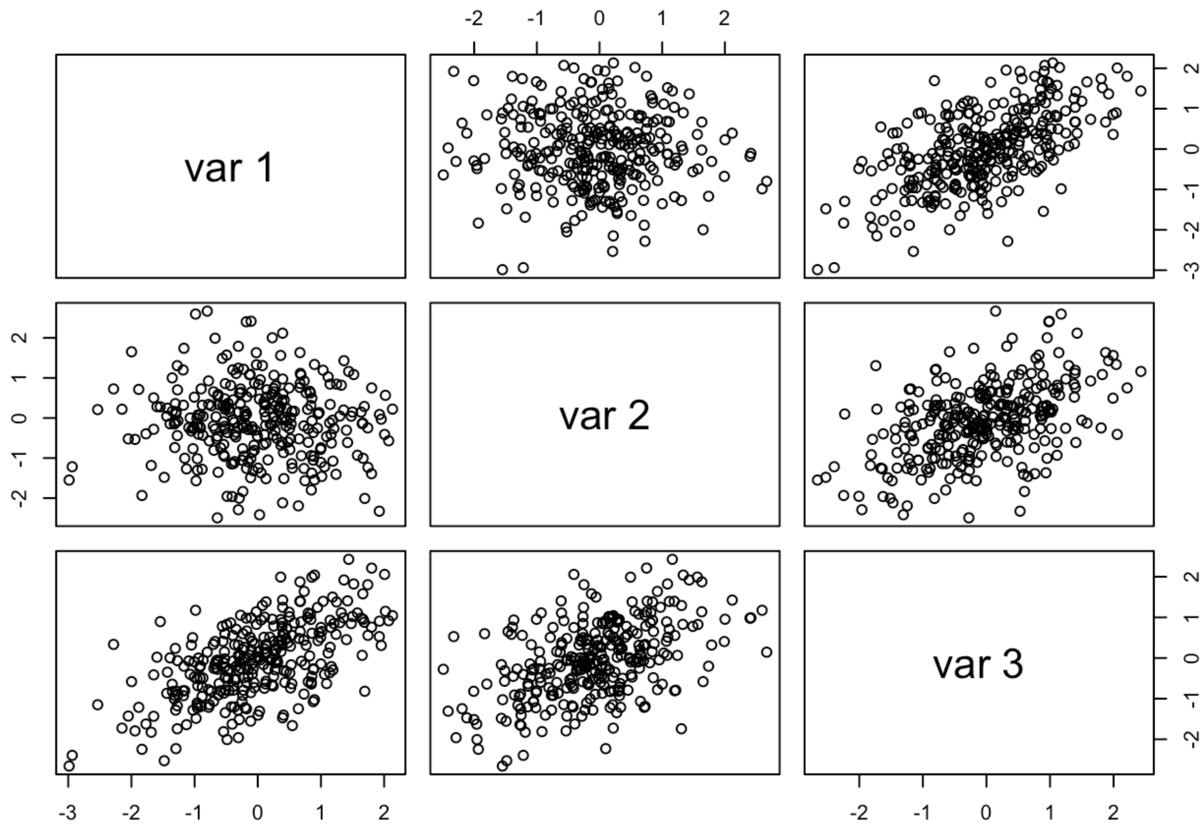
Now the outlier is clearly visible.

```
par(mfrow = c(1,1))
plot(dat, xlim=c(-5,5), ylim=c(-5,5))
points(dat[301,1],dat[301,2],col="red")
```



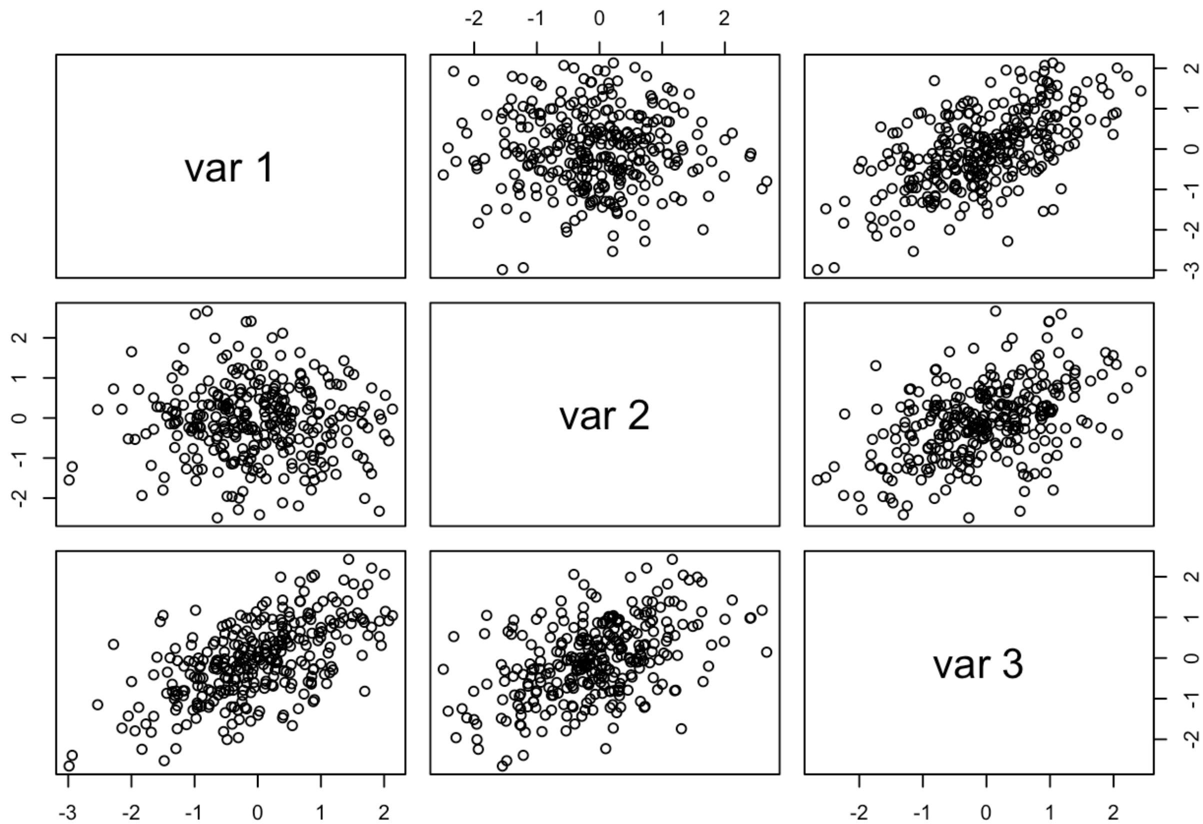
2.1.3 More dimensions

```
load(file = "3dExample.rda")
pairs(dat)
```

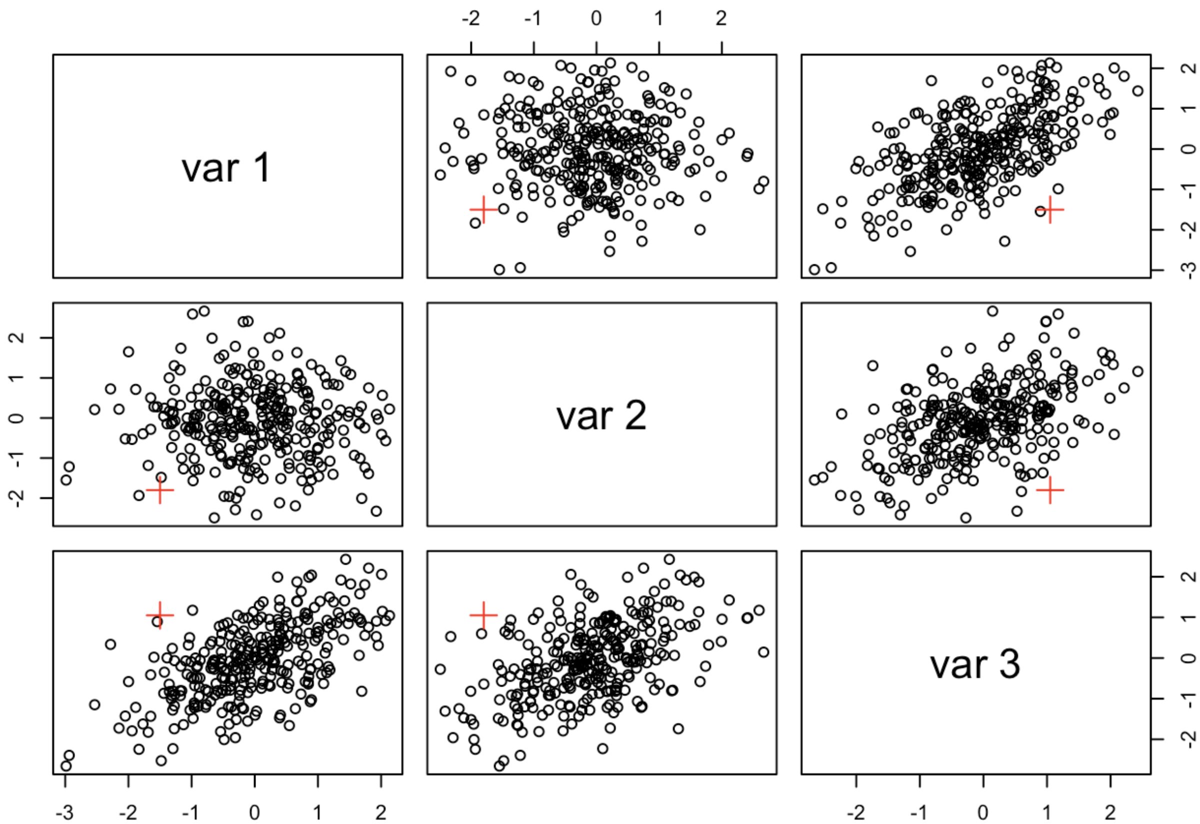


Introduce an outlier.

```
outFactor <- 1.5
dat <- rbind(dat, outFactor*c(-1,-1.2,0.7))
pairs(dat)
```

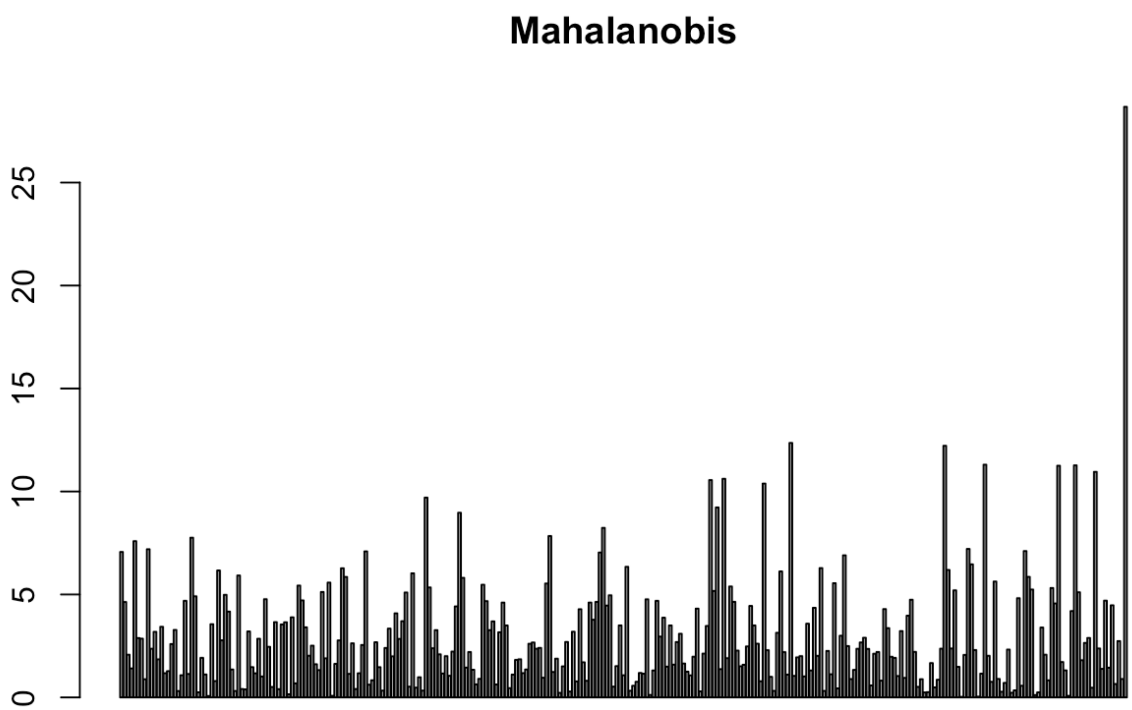



```
pairs(dat, col = c(rep(1,300), 2), pch = c(rep(1,300), 3), cex = c(rep(1,300), 2))
```



In none of the plots, the point is an outlier.

```
d <- mahalanobis(dat, colMeans(dat), cov(dat))
barplot(d, main="Mahalanobis")
```

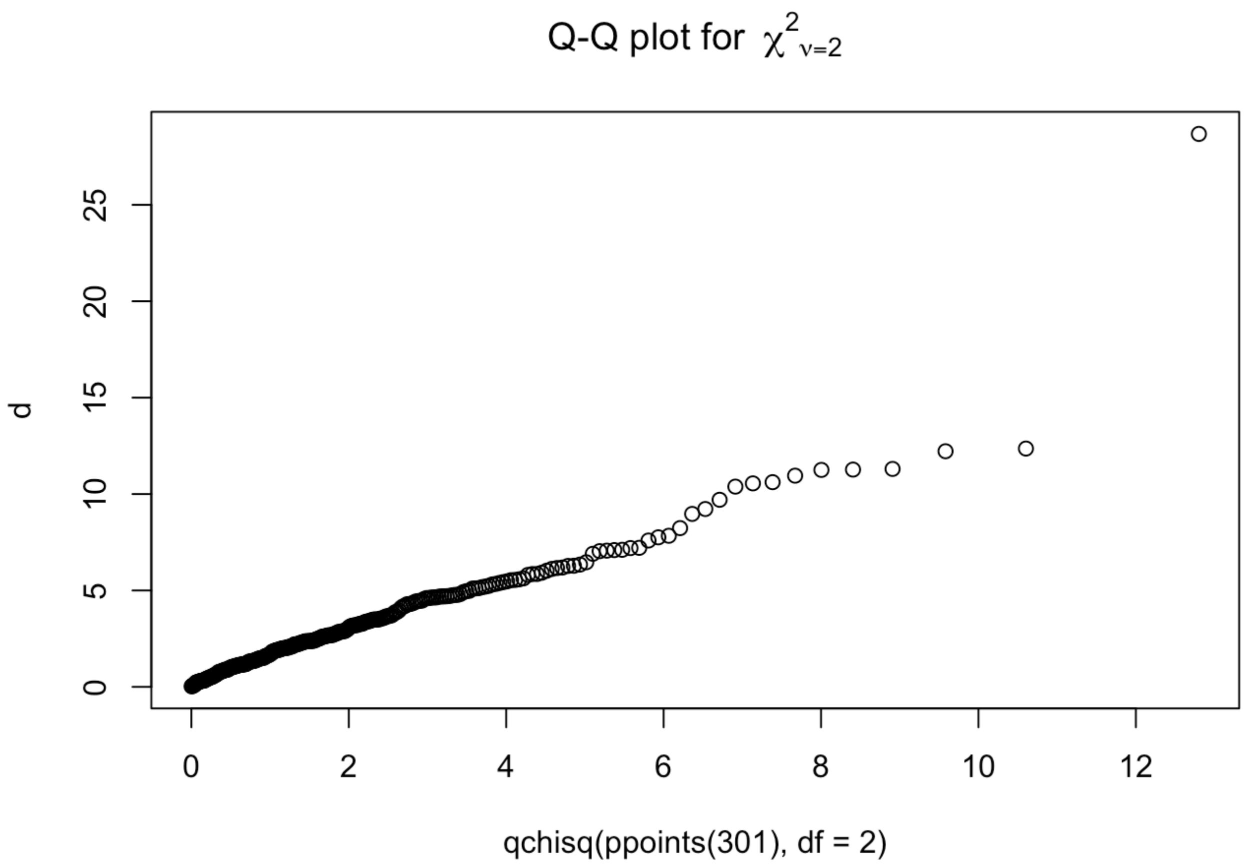



```
which.max(d)
```

```
## [1] 301
```

Create chi-squared QQ-plot, which will help show that it is a multivariate outlier.

```
par(mfrow=c(1,1))
qqplot(qchisq(ppoints(301), df = 2), d,
       main = expression("Q-Q plot for" ~~ {chi^2}[nu == 2]))
```



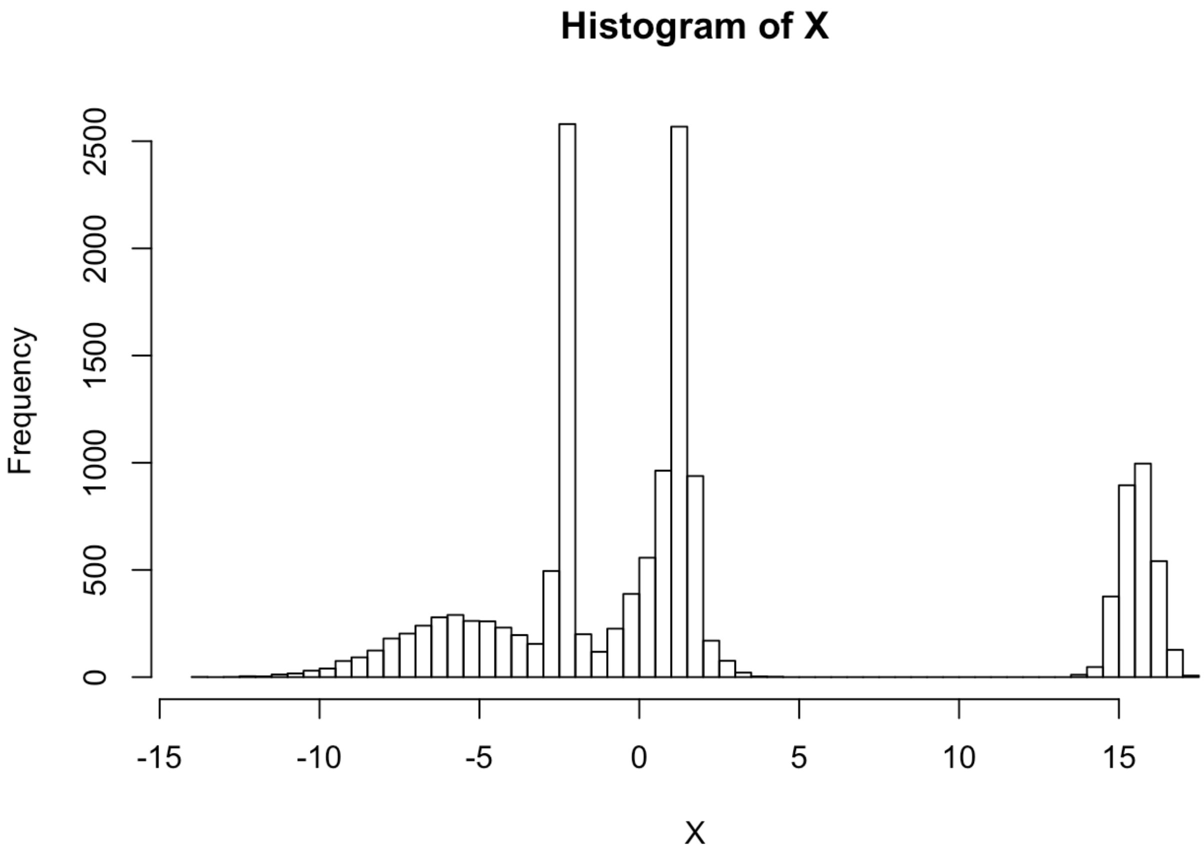
Let's do a fancy 3D plot. First run `install.packages('rgl')` to install the `rgl` package.

```
library(rgl)
plot3d(dat, col = c(rep(1,300), 2))
```

2.2 Heterogeneity

```
set.seed(123)
mus <- rnorm(5, mean=0, sd=10)
sds <- rchisq(5, 1)
```

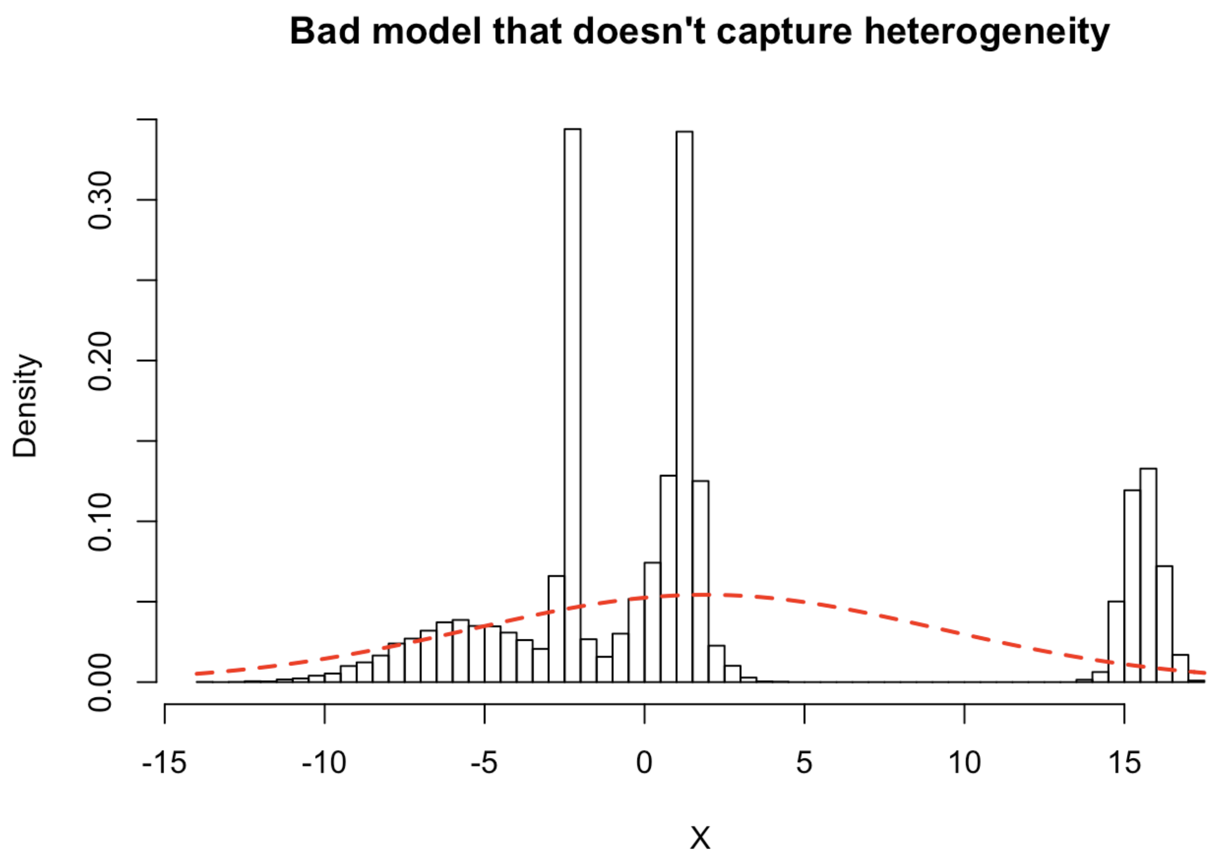
```
library(MASS)
Sigma <- diag(sds)
X <- as.vector(mvrnorm(n=3000,mus,Sigma))
hist(X, breaks=100)
```




```
summary(X)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -13.7700  -2.3920   0.6767   1.9200   1.6330  17.2600
```

```
mu <- mean(X)
sd <- sd(X)
hist(X, breaks=100, freq=FALSE, main="Bad model that doesn't capture heterogeneity")
curve(dnorm(x, mean=mu, sd=sd), col=2, lty=2, lwd=2, add=TRUE)
```



2.3 Noise accumulation

Try to replicate the example (Fig. 1) in the "Noise Accumulation" section of the paper *Challenges of Big Data analysis* by Fan, Han, and Liu. Useful functions are `prcomp` and `mvrnorm` (in the `MASS` package).

We setup the number of observations `n` and the number of dimensions `p` .

```
n <- 100
p <- 1000
```

The first class has a zero mean.

```
mu <- rep(0, p)
```

The second class has mean `eta` . It is a sparse mean with 0 entries everywhere except for the first 10 dimensions where the mean is 3.

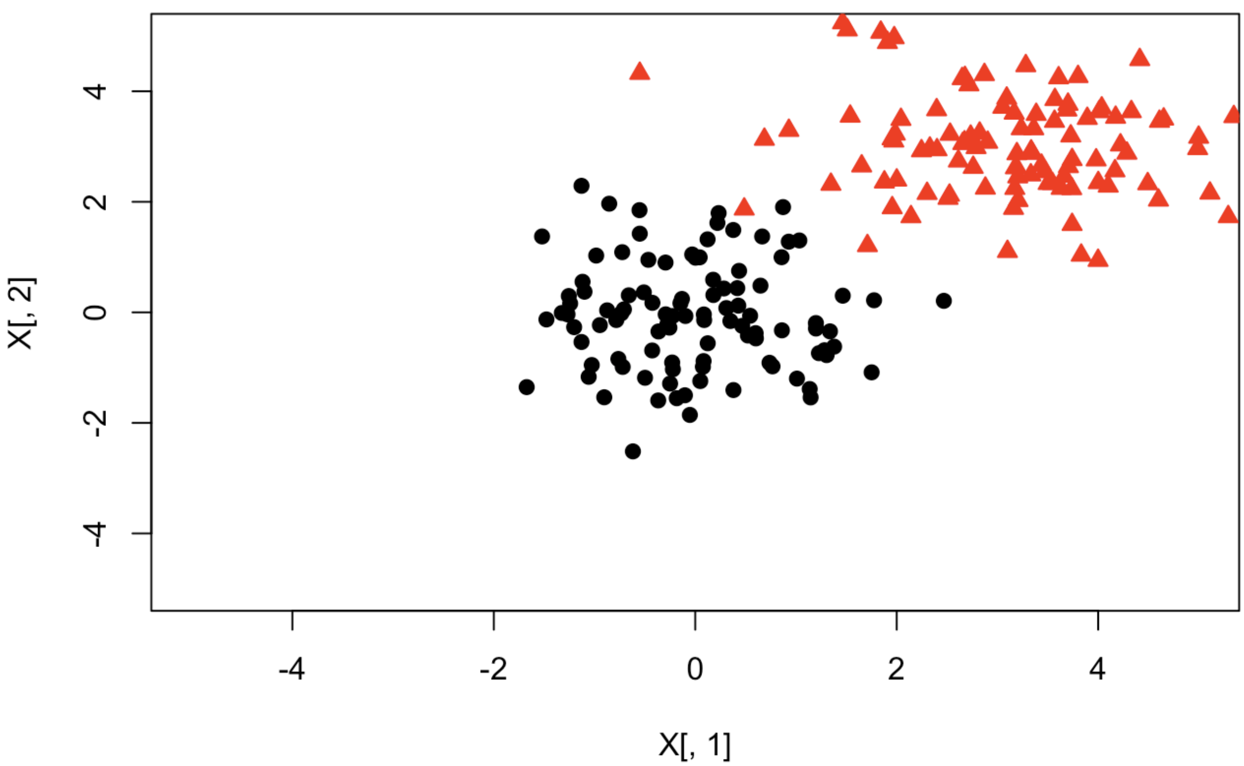
```
eta <- rep(0, p)
eta[1:10] <- 3
```

Sample from each class.

```
I <- diag(1,p,p)
X <- mvrnorm(n=n, mu, I)
Y <- mvrnorm(n=n, eta, I)
```

Plot the first two dimensions against each other for the two classes.

```
plot(X[,1], X[,2], bg=1, pch=19, xlim=c(-5,5), ylim=c(-5,5))
points(Y[,1], Y[,2], col=2, bg=2, pch=24)
```



Stack the data into one matrix.

```
Z <- rbind(X,Y)
```

```
#m <- 2
#pcaZ <- prcomp(t(Z), scale=TRUE)
#features <- pcaZ$rotation[1:m,]
#Zc <- t(features) %*% t(Z)
#Za <- t(features %*% Zc)
```

```
#dim(Za)

# pca <- prcomp(Z[,1:500], center=FALSE, scale=TRUE, retx=TRUE)
# pX <- pca$x[1:n,1:2]
# pY <- pca$x[(n+1):nrow(pZ),1:2]
# plot(pX[,1], pX[,2], bg=1, pch=19, xlim=c(-5,5), ylim=c(-5,5))
# points(pY[,1], pY[,2], col=2, bg=2, pch=24)
```