Classification with R

Data Mining
Master Degree in Computer Science
University of Padova

a.y. 2017/2018

Annamaria Guolo

1 Mtcars dataset

Dataset mtcars contains information from 1974 *Motor Trend US* magazine about automobile design and performance for 32 automobiles in the period 1973-1974. The following analysis is inspired by the public results available online on the webpage *Cookbook for R*.

```
data(mtcars)
names(mtcars)

##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear" "carb"
```

Among the different variables, we will consider

- vs: V engine (vs=0) or a straight engine (vs=1)
- mpg: Fuel efficiency, Miles/(US) gallon
- am: Transmission indicator (0 = automatic, 1 = manual)

```
## extract the information of interest
cars.data <- mtcars[, c('mpg', 'vs', 'am')]
cars.data[1:3,]

##                mpg vs am
## Mazda RX4     21.0  0  1
## Mazda RX4 Wag 21.0  0  1
## Datsun 710    22.8  1  1

dim(cars.data)

## [1] 32  3
```

Is variable `am` a factor?

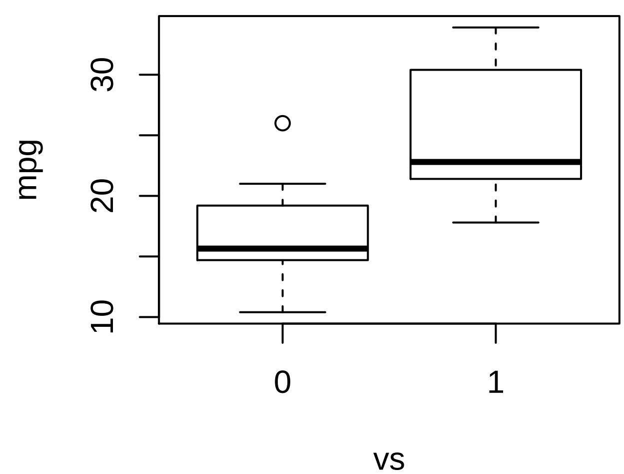
```
is.factor(cars.data$am)

## [1] FALSE
```

Make it a factor

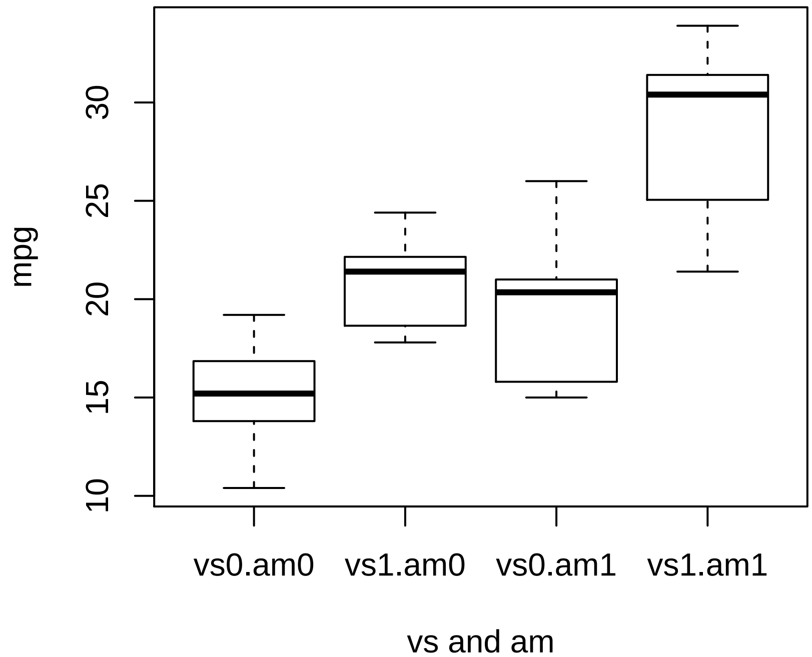
```
cars.data$am <- as.factor(cars.data$am)
```

Some graphical evaluations of the relationships between variables

```
## relationship between vs and mpg
boxplot(cars.data$mpg~cars.data$vs,  xlab='vs', ylab='mpg')
```




```
## add on am
boxplot(cars.data$mpg~cars.data$vs*cars.data$am,  ylab='mpg', xlab='vs and am',
        names=c('vs0.am0','vs1.am0','vs0.am1', 'vs1.am1'))
```



Comments?

1.1 Logistic regression model

Fit the logistic regression model with variables `mpg`, `am` and their interaction, that is, the complete model with all the covariates.

```
model.cars <- glm(vs ~ mpg*am, data=cars.data, family=binomial)
```

The fit is done using command `glm` with option `family=binomial`. By default, the function used is logit. Alternatives need to be specified, but we will not see how it.

```
summary(model.cars)

##
## Call:
## glm(formula = vs ~ mpg * am, family = binomial, data = cars.data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.70566  -0.31124  -0.04817   0.28038   1.55603
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -20.4784    10.5525  -1.941   0.0523 .
## mpg           1.1084     0.5770   1.921   0.0547 .
## am1          10.1055    11.9104   0.848   0.3962
## mpg:am1      -0.6637     0.6242  -1.063   0.2877
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43.860  on 31  degrees of freedom
## Residual deviance: 19.125  on 28  degrees of freedom
## AIC: 27.125
##
## Number of Fisher Scoring iterations: 7
```

The output from `summary` contains:

- information about residuals
- estimates, standard error, significance test on all the parameters (with normal variable as reference distribution)
- information about null deviance and model deviance
- AIC (more later)
- number of iterations of the algorithm needed to compute the maximum likelihood estimates

Comments on the output?
Model without interaction

```r
model.cars2 <- glm(vs ~ mpg+am, data=cars.data, family=binomial)
summary(model.cars2)

##
## Call:
## glm(formula = vs ~ mpg + am, family = binomial, data = cars.data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.05888  -0.44544  -0.08765   0.33335   1.68405
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -12.7051     4.6252  -2.747  0.00602 **
```

```
## mpg           0.6809     0.2524   2.698  0.00697 **
## am1          -3.0073     1.5995  -1.880  0.06009 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43.860  on 31  degrees of freedom
## Residual deviance: 20.646  on 29  degrees of freedom
## AIC: 26.646
##
## Number of Fisher Scoring iterations: 6
```

The significance of the parameter associated to `am` is questionable.
Extract the estimates of the coefficients

```
estimate <- coef(model.cars2)
estimate

## (Intercept)         mpg         am1
## -12.7051158   0.6809205  -3.0072739
```

and the associated standard error

```
se <- sqrt(diag(vcov(model.cars2)))
se

## (Intercept)         mpg         am1
##   4.6252123   0.2523749   1.5994800
```

Compute a confidence interval of level 0.90 for the coefficient associated to `mpg`

```
c(estimate[2]-qnorm((1+0.90)/2)*se[2], estimate[2]+qnorm((1+0.90)/2)*se[2])

##       mpg       mpg
## 0.2658008 1.0960402
```

where we used the quantiles from a standard normal provided by function `qnorm`. The interval does not contain 0, so the parameter is significantly far from zero. Note that command

```
confint(model.cars2, level=0.90)

## Waiting for profiling to be done...

##                     5 %      95 %
## (Intercept) -22.5194716 -6.665149
## mpg           0.3517087  1.216123
## am1          -6.0959520 -0.690963
```

provides a different confidence interval using a different likelihood quantity, called profile likelihood.
We can carry out hypothesis testing as done for linear regression model and using the standard normal distribution as reference.
Evaluate the accuracy of model `model.cars2` on the basis of the deviance

```
1-pchisq(20.646, 29)

## [1] 0.8717172
```

The p-value associated to the `Residual deviance` indicates that the model is a good simplification of the saturated model (which is associated to the maximum value fo the likelihood), so it is acceptable.
Can we even more simplify the model by eliminating `am`?

```
model.cars3 <- glm(vs ~ mpg, data=cars.data, family=binomial)
summary(model.cars3)

##
## Call:
## glm(formula = vs ~ mpg, family = binomial, data = cars.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2127  -0.5121  -0.2276   0.6402   1.6980
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.8331     3.1623  -2.793  0.00522 **
## mpg           0.4304     0.1584   2.717  0.00659 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 43.860  on 31  degrees of freedom
## Residual deviance: 25.533  on 30  degrees of freedom
## AIC: 29.533
##
## Number of Fisher Scoring iterations: 6
```

We can compare the deviances using function `anova`

```r
anova(model.cars3, model.cars2, test='Chisq')

## Analysis of Deviance Table
##
## Model 1: vs ~ mpg
## Model 2: vs ~ mpg + am
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        30     25.533
## 2        29     20.646  1    4.887  0.02706 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Given the output, we maintain model `model.cars2`. We can calculate the statistic without using function `anova`. Consider that the models are nested and that they differ for 1 parameter. The deviance of `model.cars2` is smaller than that of `model.cars3`, as the first model is nested in the second model. The difference of deviances follows a χ^2_1 distribution and it is equal to

```r
25.533 - 20.646

## [1] 4.887
```

We reject the null hypothesis, that is moving to the simplified model `model.cars3`, for large values of the deviance comparison. Assuming a significance level equal to 0.05, we reject the null hypothesis if the observed values is larger than

```r
qchisq(0.95, 1)

## [1] 3.841459
```

There is empirical evidence against the simplification of the model. The associated p-value is

```r
1-pchisq(4.887, 1)

## [1] 0.02705967
```

as reported in the output of function `anova()`.
Compute the estimated value from `model.cars2` on the training set

```r
est.values <- predict(model.cars2)
est.values[1:4]

##      Mazda RX4  Mazda RX4 Wag     Datsun 710 Hornet 4 Drive
##     -1.4130585     -1.4130585     -0.1874016      1.8665835
```

The function with no option specifications provides the predictions on the training set for logit transformation. While using the option type='response'

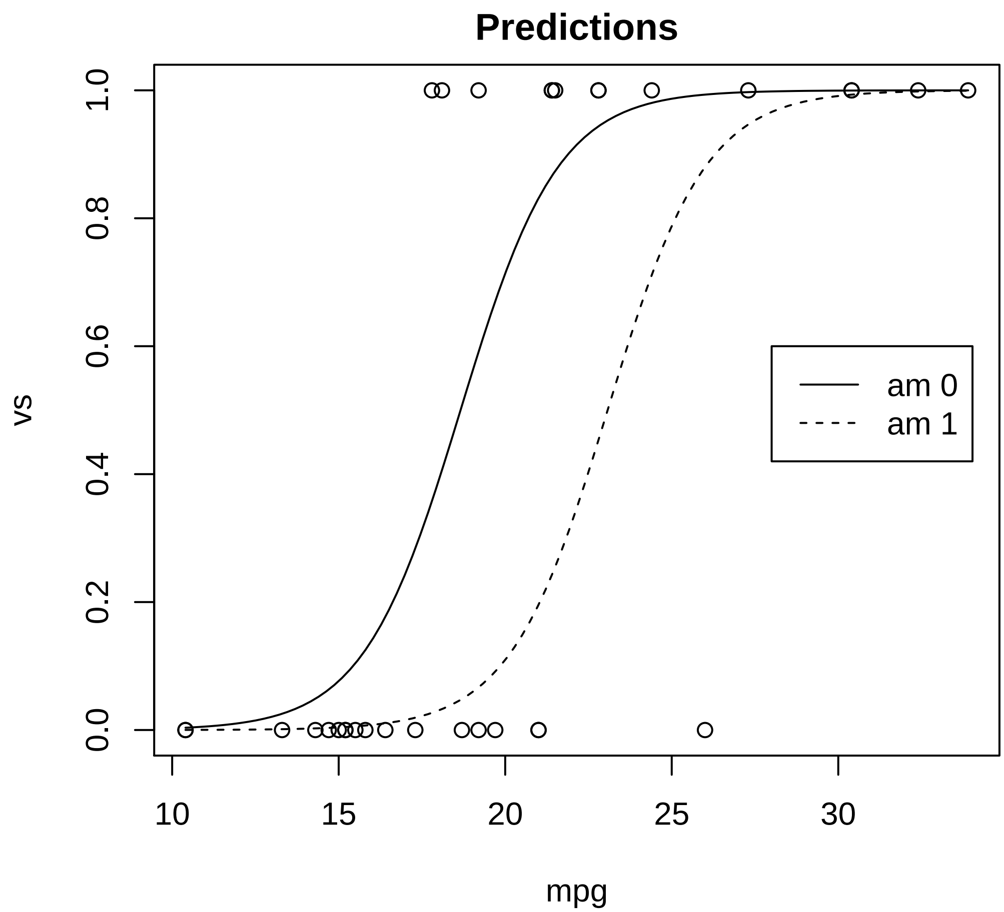
```
est.probs <- predict(model.cars2, type='response')
est.probs[1:4]

##      Mazda RX4  Mazda RX4 Wag     Datsun 710 Hornet 4 Drive
##      0.1957521      0.1957521      0.4532862      0.8660625
```

it provides the predictions in terms of probability. The same predictions can be obtained also as

```
exp(est.values)/(1+exp(est.values))

##           Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive
##         0.195752093         0.195752093         0.453286237         0.866062468
##   Hornet Sportabout             Valiant          Duster 360           Merc 240D
##         0.507024063         0.406017350         0.048894865         0.980340613
##            Merc 230            Merc 280           Merc 280C          Merc 450SE
##         0.943740285         0.591110583         0.357844862         0.176823422
##          Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental
##         0.283901447         0.086659370         0.003598826         0.003598826
##   Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla
##         0.063234437         0.998255316         0.993224169         0.999371042
##       Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28
##         0.873766034         0.104252045         0.086659370         0.025360551
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa
##         0.591110583         0.946684602         0.879906401         0.993224169
##      Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E
##         0.007006782         0.091267566         0.004075891         0.242193639
```

using the expression of the logistic function $P(Y = 1|x) = e^x/(1 + e^x)$.
Plot the estimated probabilities by distinguishing the levels of am

```
plot(cars.data$mpg, cars.data$vs, xlab='mpg', ylab='vs', main='Predictions')
curve(predict(model.cars2, newdata=data.frame(mpg=x, am="0"),
        type='response'), add=TRUE)
curve(predict(model.cars2, newdata=data.frame(mpg=x, am="1"),
        type='response'), add=TRUE, lty=2)
legend(28, 0.6, lty=c(1,2), legend=c('am 0','am 1'))
```



Predicted values of vs are

```
preds <- rep(0, nrow(cars.data))
preds[est.probs > 0.5] <- 1
```

Note that the prediction is 0 or 1 if the probability is smaller or larger than 0.5.
Evaluate the prediction capability of the model by computing the confusion matrix, also called misclassification matrix

```
addmargins(table(preds, vs=cars.data$vs))

##       vs
## preds  0  1 Sum
##   0   15  4  19
##   1    3 10  13
##   Sum 18 14  32
```

Note that `table()` provides the table, while `addmargins()` adds on the sums over rows and columns.
The total error rate is 7/32=21.875% (training error rate)

Evaluate the model on a test set constructed from the original data. Choose randomly 60% of the original data as training set and the remaining as test set.

```
n <- nrow(cars.data)
set.seed(222)
selection <- sample(n, 0.60*n, replace=FALSE)
selection

##  [1] 30  3 15  1 26 28 10 11 14  4  9  2 31 24 17 29 25 18 21
```

Function `sample` sample from a set of `n` objects a number of `0.6*n` objects, without re-sampling. As the sample is random, we fix the seed using `set.seed` in order to obtain the same selection every time the command is running.

```
## training set and test set
training.set <- cars.data[selection, ]
test.set <- cars.data[-selection, ]
```

Pay attention to the specification `-selection` useful to include in the test set all the observations available in `cars.data` except those belonging to the selection.

```
## model fitted on the training set
model.cars.train <- glm(vs ~ mpg + am, data=training.set, family=binomial)
summary(model.cars.train)

##
## Call:
## glm(formula = vs ~ mpg + am, family = binomial, data = training.set)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.80226  -0.17226  -0.00836   0.10825   1.37418
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -24.0502    15.2087  -1.581   0.1138
## mpg           1.3258     0.8175   1.622   0.1048
## am1          -5.4999     3.2141  -1.711   0.0871 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26.2869  on 18  degrees of freedom
## Residual deviance:  7.2473  on 16  degrees of freedom
```

```
## AIC: 13.247
##
## Number of Fisher Scoring iterations: 8
```

Predictions on the test set

```
probs.test <- predict(model.cars.train, newdata=test.set, type='response')
preds.test <- rep(0, length(probs.test))
preds.test[probs.test>0.5] <- 1
```

Model evaluation

```
addmargins(table(preds.test, vs=test.set$vs))

##           vs
## preds.test  0  1 Sum
##        0    6  2   8
##        1    2  3   5
##        Sum  8  5  13
```

Test error rate

```
4/13

## [1] 0.3076923
```

Reasons for such a behaviour? What happens changing the seed?

1.2 Discriminant analysis

Consider the linear discriminant analysis on the training set using variables `mpg` and `am` without interaction. We fit the model using function `lda()` with the same syntax used for the logistic regression model, without specifying `family=binomial`. Function `lda()` is implemented inside library `MASS`

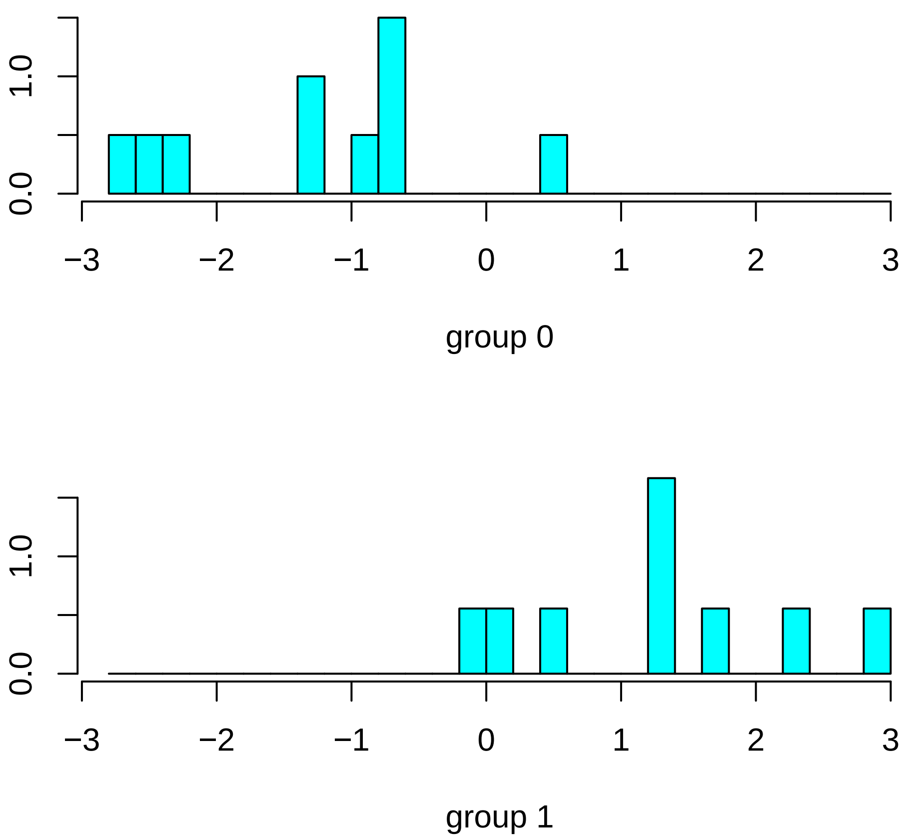
```
## upload the library
library(MASS)
```

```
model.cars.lda <- lda(vs ~ mpg + am, data=training.set)
model.cars.lda

## Call:
## lda(vs ~ mpg + am, data = training.set)
##
```

```
## Prior probabilities of groups:
##         0         1 
## 0.5263158 0.4736842 
## 
## Group means:
##        mpg       am1
## 0 16.53000 0.5000000
## 1 23.95556 0.4444444
## 
## Coefficients of linear discriminants:
##            LD1
## mpg  0.3263393
## am1 -1.9399213
```

Function `plot()` applied to the fitted model provides a graphical representation of the results, through a histogram of the values from the discriminant function for the observations from each group.

```
plot(model.cars.lda)
```



Groups are not well differentiated, as histograms partially overlap.
Predictions on the test set

```
preds.lda <- predict(model.cars.lda, test.set)
```

Function `predict()` applied to the fitted model provides a list with components including

- `posterior`: the posterior probabilities that each observation belongs to group "0" or group "1"
- `x`: the value of the linear discriminant function

Misclassification table

```
preds.lda1 <- rep(0, nrow(test.set))
preds.lda1[preds.lda$posterior[,2]>0.5] <- 1
addmargins(table(predictions=preds.lda1, vs=test.set$vs))

##            vs
## predictions  0  1 Sum
##         0    6  1   7
##         1    2  4   6
##         Sum  8  5  13

## test error rate
3/13

## [1] 0.2307692
```

Predictions at cutoff different from 50%

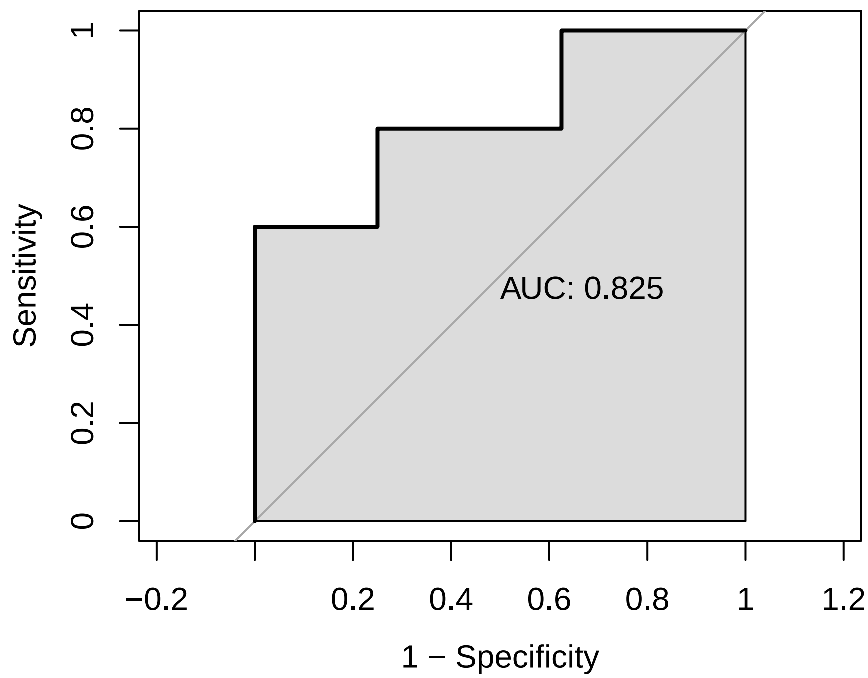
```
preds.lda2 <- rep(0, nrow(test.set))
preds.lda2[preds.lda$posterior[,2]>0.2] <- 1
addmargins(table(predictions=preds.lda2, vs=test.set$vs))

##            vs
## predictions  0  1 Sum
##         0    4  1   5
##         1    4  4   8
##         Sum  8  5  13
```

The ROC curve can be obtained using functionalities inside library `pROC`. Function `roc` requires as input the observed values and the predictions and it provides sensitivity and specificity at different thresholds, among others.

```r
## in order to install the library, if not already installed in R
## type install.packages('pROC') and choose the mirror
## for downloading
library(pROC)
values.roc <- roc(test.set$vs, preds.lda$posterior[,2] )
values.roc

##
## Call:
## roc.default(response = test.set$vs, predictor = preds.lda$posterior[,     2])
##
## Data: preds.lda$posterior[, 2] in 8 controls (test.set$vs 0) < 5 cases (test.set$vs 1
## Area under the curve: 0.825

names(values.roc)

##  [1] "percent"            "sensitivities"      "specificities"      "thresholds"
##  [5] "direction"          "cases"              "controls"           "fun.sesp"
##  [9] "auc"                "call"               "original.predictor" "original.respons
## [13] "predictor"          "response"           "levels"

values.roc$sensitivities

##  [1] 1.0 1.0 1.0 1.0 0.8 0.8 0.8 0.8 0.6 0.6 0.6 0.4 0.2 0.0

values.roc$specificities

##  [1] 0.000 0.125 0.250 0.375 0.375 0.500 0.625 0.750 0.750 0.875 1.000 1.000 1.000 1.

values.roc$thresholds

##  [1]       -Inf 0.03296835 0.09379518 0.13676911 0.15156013 0.21532193 0.36129096
##  [8] 0.52746112 0.66397474 0.80284912 0.94164726 0.99656282 0.99822712        Inf
```

Plot of the ROC curve

```r
plot(values.roc, legacy.axes=TRUE, xlim=c(1.0, 0.0), print.auc=TRUE,
        auc.polygon=TRUE)
```



Consider the quadratic discriminant analysis. The analysis can be performed using function `qda()`, with a syntax similar to `lda()`

```
model.cars.qda <- qda(vs ~ mpg + am, data=training.set)
model.cars.qda

## Call:
## qda(vs ~ mpg + am, data = training.set)
##
## Prior probabilities of groups:
##         0         1
## 0.5263158 0.4736842
##
## Group means:
##        mpg       am1
## 0 16.53000 0.5000000
## 1 23.95556 0.4444444
```

Predictions on the test set

```
preds.qda <- predict(model.cars.qda, test.set)
preds.qda

## $class
##  [1] 1 1 0 1 0 0 0 1 1 0 0 1 0
```

```
## Levels: 0 1
##
## $posterior
##                                0           1
## Hornet Sportabout   3.077033e-01 0.692296719
## Valiant             4.056445e-01 0.594355516
## Duster 360          9.087183e-01 0.091281690
## Merc 240D           6.944431e-03 0.993055569
## Merc 450SE          6.952336e-01 0.304766426
## Merc 450SL          5.467173e-01 0.453282671
## Lincoln Continental 9.931985e-01 0.006801521
## Honda Civic         3.134058e-04 0.999686594
## Toyota Corolla      4.793255e-06 0.999995207
## Dodge Challenger    8.113011e-01 0.188698914
## AMC Javelin         8.414384e-01 0.158561581
## Porsche 914-2       5.278851e-02 0.947211489
## Volvo 142E          9.197698e-01 0.080230178
```

Function `predict()` applied to the model provides a list of two elements

- `class`: the predictions
- `posterior`: the posterior probabilities that each observation belongs to group "0" or group "1"

Misclassification table

```
addmargins(table(predictions=preds.qda$class, vs=test.set$vs))

##             vs
## predictions  0  1 Sum
##         0    6  1   7
##         1    2  4   6
##         Sum  8  5  13

## test error rate
3/13

## [1] 0.2307692
```

On the basis of this test set the performance of DLA and that of QDA are similar in terms of test error rate. Thus, we prefer LDA given its simplicity if compared to QDA.
Compute the ROC curve

```
values.roc <- roc(test.set$vs, preds.qda$posterior[,2] )
values.roc

##
## Call:
## roc.default(response = test.set$vs, predictor = preds.qda$posterior[,     2])
##
## Data: preds.qda$posterior[, 2] in 8 controls (test.set$vs 0) < 5 cases (test.set$vs 1
## Area under the curve: 0.775
```

Conclusion?

2 Auto Dataset

This example is inspired by exercise number 11 in chapter 4 of the textbook Gareth J, Witten D, Hastie T, Tibshirani R. An Introduction to Statistical Learning with Applications in R.

```
library(ISLR)
data(Auto)
dim(Auto)

## [1] 392   9

Auto[1:3,]

##   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
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
```

Data refer to the characteristics of 392 cars. Consider the following variables:

- `mpg`: Fuel efficiency, Miles/(US) gallon
- `displacement`: Engine displacement (cu. inches)
- `horsepower`: Engine horsepower
- `origin`: Origin of car (1. America, 2. Europe, 3. Japan)

Create a new variable called `new.mpg` indicating whether the car has a high (value 1) or low (value 0) `mpg`, by distinguishing values under or below the median of `mpg`. Then, evaluate whether the fuel efficiency depends on the remaining variables and which variables are useful to predict the fuel efficiency.

```
median.mpg <- median(Auto$mpg)
## create a vector as long as the number of cars and composed by ones
new.mpg <- rep(1, length(Auto$mpg))
## substitute 0 to the ones corresponding to cars with mpg lower than the median
new.mpg[Auto$mpg < median.mpg] <- 0
## create the dataset we need
new.auto <- data.frame(new.mpg=new.mpg,
        Auto[,c('displacement', 'horsepower', 'origin')])
```

Check whether `origin` is considered as a factor

```
is.factor(new.auto$origin)

## [1] FALSE
```

Make it qualitative

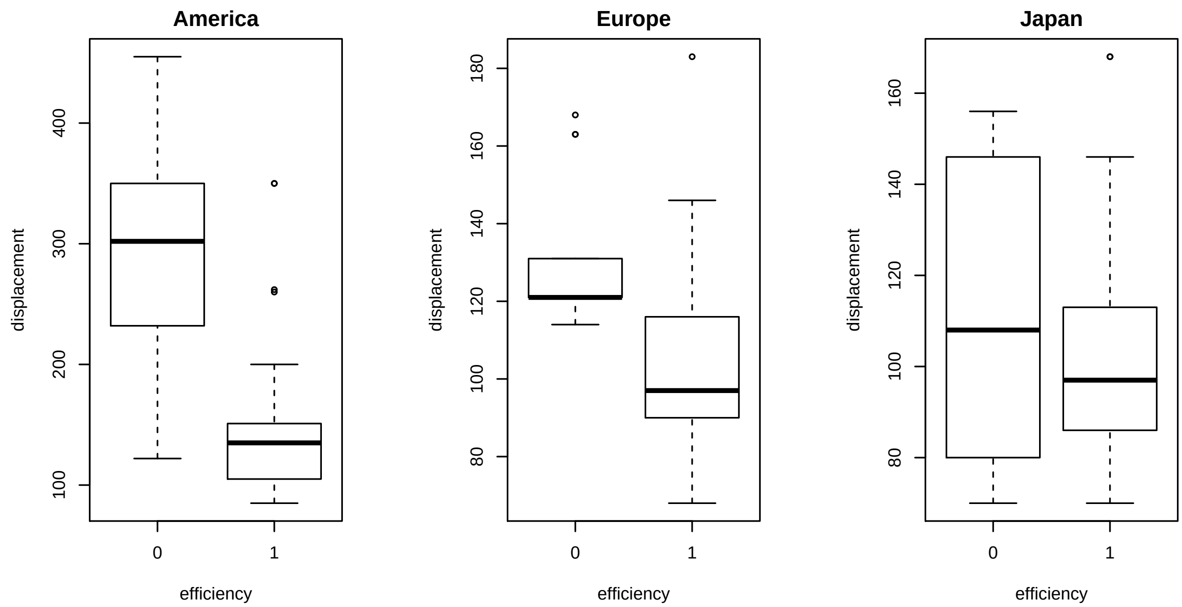
```
new.auto$origin <- as.factor(new.auto$origin)
```

Change the names of the levels with the name of the country

```
levels(new.auto$origin) <- c('America', 'Europe', 'Japan')
```

Graphics to evaluate relationships between `new.mpg` and other variables

```
## Relationship with displacement and origin
par(mfrow=c(1,3))
boxplot(displacement~new.mpg, data=new.auto,
        subset=new.auto$origin=='America', main='America', xlab='efficiency',
                ylab='displacement')
boxplot(displacement~new.mpg, data=new.auto,
        subset=new.auto$origin=='Europe', main='Europe', xlab='efficiency',
                ylab='displacement')
boxplot(displacement~new.mpg, data=new.auto,
        subset=new.auto$origin=='Japan', main='Japan', xlab='efficiency',
                ylab='displacement')
```



As an alternative...

```
by(new.auto, new.auto$origin, function(x) boxplot(x[,2] ~ x[,1]))
```

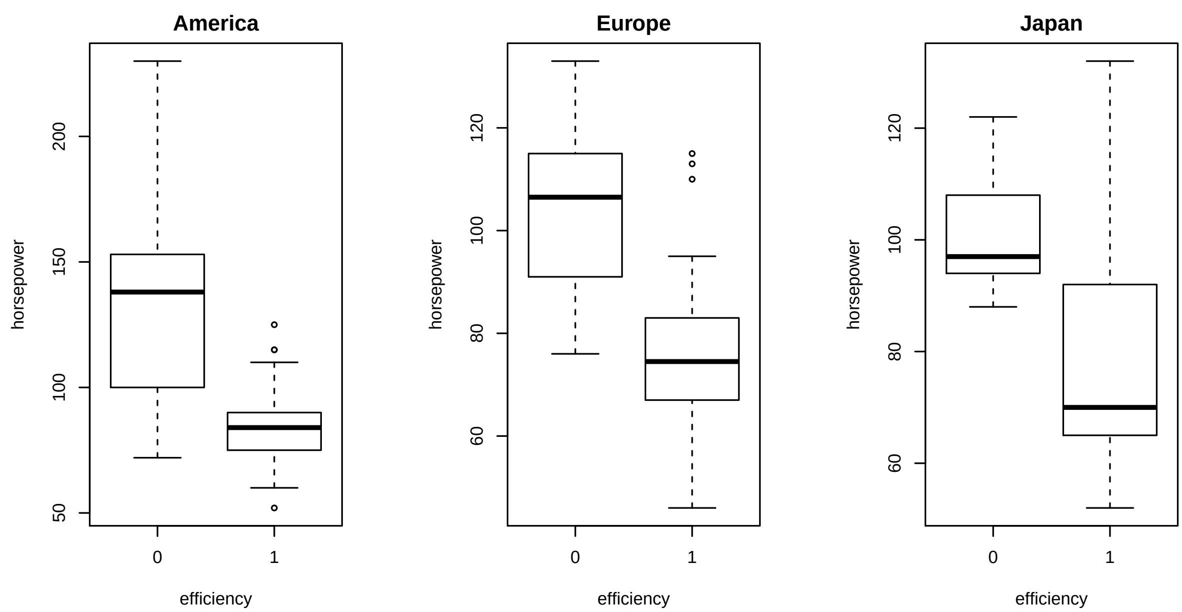
just to obtain the three boxplot quickly. Function `by()` applies a function specified as third argument to a data set (first argument) according to a subdivision provided by the second argument....in our case, it constructs a boxplot for each subgroup of `new.auto` identified by `origin`.
Command

```
by(new.auto, new.auto$origin, function(x) boxplot(x[,2] ~ x[,1], main=x[1,4]))
```

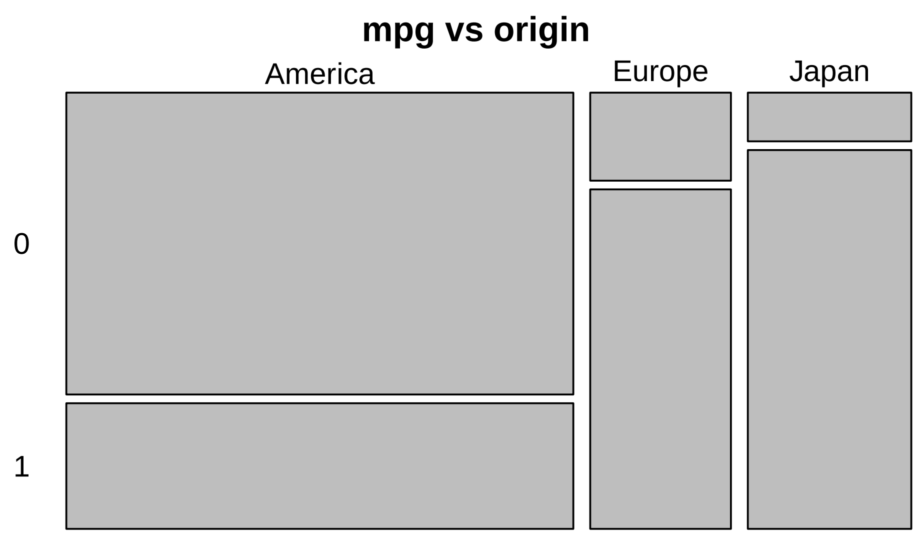
adds the name of the country to each boxplot. The name is repeated inside each block identified by `origin`, so taking the first element (of the fourth column, that corresponding to `origin`) is sufficient.
The different behaviour of the boxplots of displacement with respect to origin suggests that there could be an interaction between `displacement` and `origin`. A similar suggestion comes from boxplots of `horsepower` with respect to `origin`.

```
## Relationship between horsepower and origin
par(mfrow=c(1,3))
boxplot(horsepower~new.mpg, data=new.auto,
        subset=new.auto$origin=='America', main='America', xlab='efficiency',
                ylab='horsepower')
boxplot(horsepower~new.mpg, data=new.auto,
        subset=new.auto$origin=='Europe', main='Europe', xlab='efficiency',
                ylab='horsepower')
boxplot(horsepower~new.mpg, data=new.auto,
```

```
        subset=new.auto$origin=='Japan', main='Japan', xlab='efficiency',
                ylab='horsepower')
```




```
## Relationship with origin...it is a graph based on the distribution
## of two qualitative variables
mosaicplot(table(new.auto$origin, new.auto$new.mpg), las=1,
        cex.axis=1, main='mpg vs origin')
```



Option `las=1` plots labels corresponding to the levels of the variable in horizontal direction. The plot is the graphical translation of

```
table(new.auto$origin, new.auto$new.mpg)

##
##             0   1
##   America 173  72
##   Europe   14  54
##   Japan     9  70
```

Estimate a logistic regression model with all the covariates plus the interaction of the quantitative covariates with `origin`

```
m.auto <- glm(new.mpg ~ displacement*origin + horsepower*origin,
        data=new.auto, family=binomial)
summary(m.auto)

##
## Call:
## glm(formula = new.mpg ~ displacement * origin + horsepower *
##     origin, family = binomial, data = new.auto)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2641  -0.2329   0.0196   0.3525   3.5625
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)               6.953352   1.537970   4.521 6.15e-06 ***
## displacement             -0.035372   0.006123  -5.777 7.61e-09 ***
## originEurope              6.932632   4.039343   1.716 0.086111 .
## originJapan               1.357005   2.910867   0.466 0.641083
## horsepower               -0.007337   0.020693  -0.355 0.722903
## displacement:originEurope  0.001796   0.023289   0.077 0.938522
## displacement:originJapan   0.067139   0.019893   3.375 0.000738 ***
## originEurope:horsepower   -0.089248   0.035440  -2.518 0.011793 *
## originJapan:horsepower    -0.102000   0.039648  -2.573 0.010094 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 543.43  on 391  degrees of freedom
## Residual deviance: 195.72  on 383  degrees of freedom
```

```
## AIC: 213.72
##
## Number of Fisher Scoring iterations: 7
```

Comments?
Despite `origin` and `horsepower` are non-significant, they are maintained inside the model and they are not eliminated as they appear in interaction with `displacement` and the interaction is significant (principle of hierarchy).
Evaluate the accuracy of the model using the deviance

```
1-pchisq(205.04, 386)

## [1] 1
```

What can we conclude?
Is it possible to simplify the model?

```
m.auto2 <- glm(new.mpg ~ displacement+origin + horsepower,
        data=new.auto, family=binomial)
summary(m.auto2)

##
## Call:
## glm(formula = new.mpg ~ displacement + origin + horsepower, family = binomial,
##     data = new.auto)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5074  -0.2000   0.0612   0.4299   3.5272
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   9.32907    1.21877   7.654 1.94e-14 ***
## displacement -0.02630    0.00484  -5.434 5.52e-08 ***
## originEurope -0.53263    0.51581  -1.033 0.301793
## originJapan   0.17794    0.56595   0.314 0.753216
## horsepower   -0.05074    0.01361  -3.728 0.000193 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 543.43  on 391  degrees of freedom
## Residual deviance: 213.36  on 387  degrees of freedom
## AIC: 223.36
```

```
##
## Number of Fisher Scoring iterations: 7
```

Comparison of the two models

```
anova(m.auto2, m.auto, test='Chisq')

## Analysis of Deviance Table
##
## Model 1: new.mpg ~ displacement + origin + horsepower
## Model 2: new.mpg ~ displacement * origin + horsepower * origin
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       387     213.36
## 2       383     195.72  4   17.645 0.001448 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Conclusion?
Compute the estimated values for the logit on the training set

```
est.values <- predict(m.auto)
```

and the predictions in terms of probability

```
est.probs <- predict(m.auto, type='response')
preds <- rep(0, nrow(new.auto))
preds[est.probs>0.5] <- 1
```

Training error rate

```
addmargins(table(predictions=preds, mpg=new.auto$new.mpg))

##            mpg
## predictions   0   1 Sum
##         0   176  15 191
##         1    20 181 201
##         Sum 196 196 392

35/392

## [1] 0.08928571
```

Calculate (without `R` functionalities) the probability of fuel efficiency and the predicted fuel efficiency for a Japanese car with `displacement=350` and `horsepower=170`

```
estimates <- coef(m.auto)
pred.japan <- estimates[1]+estimates[2]*350+ estimates[4]+estimates[5]*170+
        estimates[7]*350+estimates[9]*170
prob.japan <- exp(pred.japan)/(1+exp(pred.japan))
prob.japan

## (Intercept)
##   0.6988298
```

Using a classical cutoff at 50% we assign the observation to group `new.mpg=1` (efficiency over the median).
Check whether the computation is correct

```
predict(m.auto, newdata=data.frame(horsepower=170, displacement=350,
        origin="Japan"), type='response')

##         1
## 0.6988298

## ok!
```

For an American car? Pay attention to the fact that `America` is the basic level of `origin`

```
pred.america <- prob.japan[1]+estimates[2]*350+ estimates[5]*170
prob.america <- exp(pred.america)/(1+exp(pred.america))
prob.america

##  (Intercept)
## 2.427592e-06

## check
predict(m.auto, newdata=data.frame(horsepower=170, displacement=350,
        origin="America"), type='response')

##           1
## 0.001261632

## ok!
```

We assign the observation to group `new.mpg=0` (efficiency under the median).

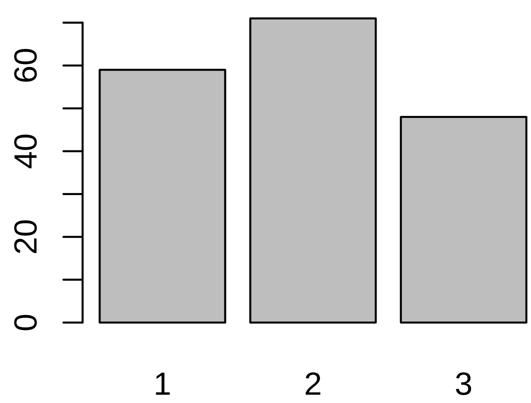
3 Wine dataset

Dataset `wine` in library `rattle.data` contains the results of 13 chemical analyses of three types of wines grown in a specific area of Italy.

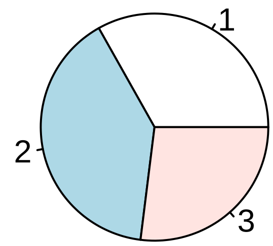
```
data(wine, package='rattle.data')
## upload the data without installing the library
dim(wine)

## [1] 178  14

names(wine)

##  [1] "Type"            "Alcohol"         "Malic"           "Ash"
##  [5] "Alcalinity"      "Magnesium"       "Phenols"         "Flavanoids"
##  [9] "Nonflavanoids"   "Proanthocyanins" "Color"           "Hue"
## [13] "Dilution"        "Proline"
```

Variable `Type` distinguishing the wine grown. The analysis wants to construct a model to predict the wine grown on the basis of the chemical characteristics of the wine. We will use the discriminant analysis.
First look at the data

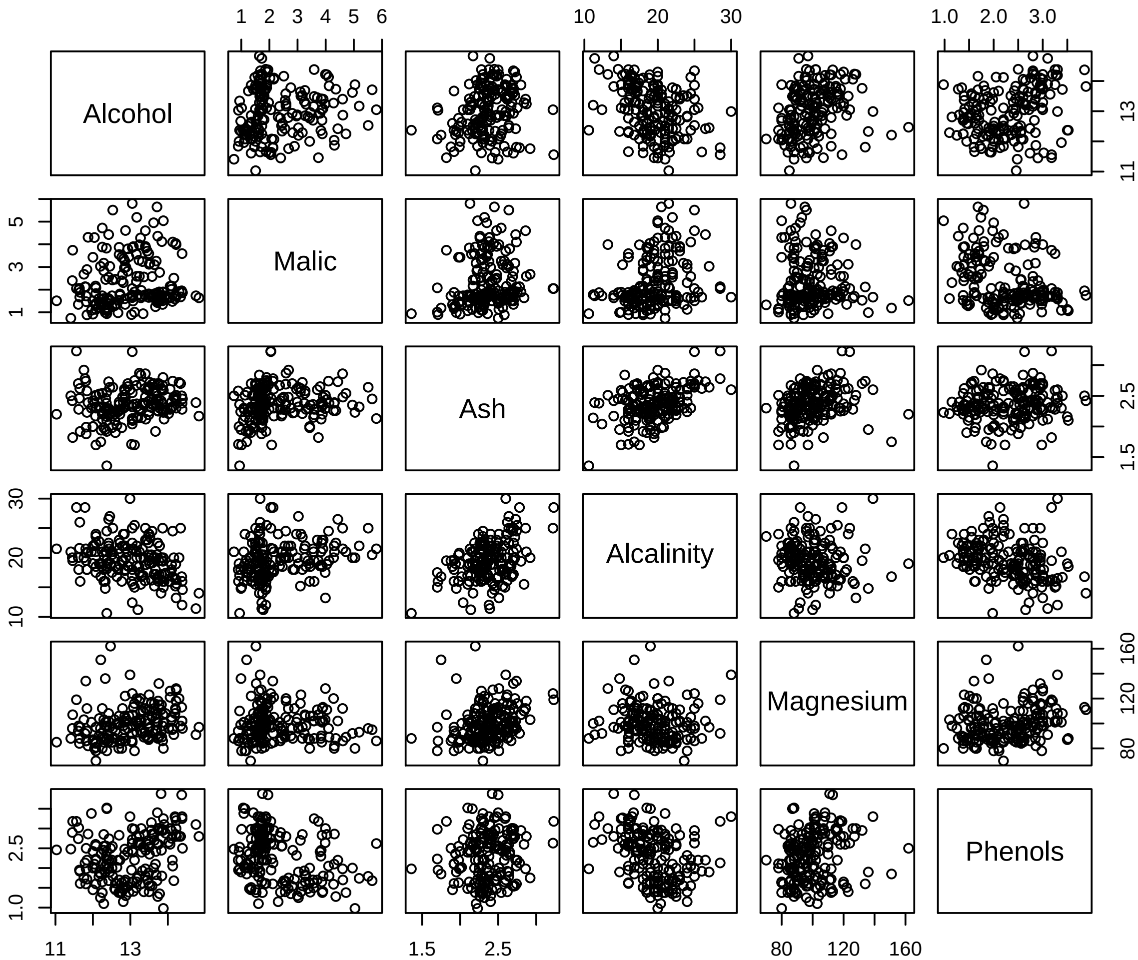
```
barplot(table(wine$Type))
```



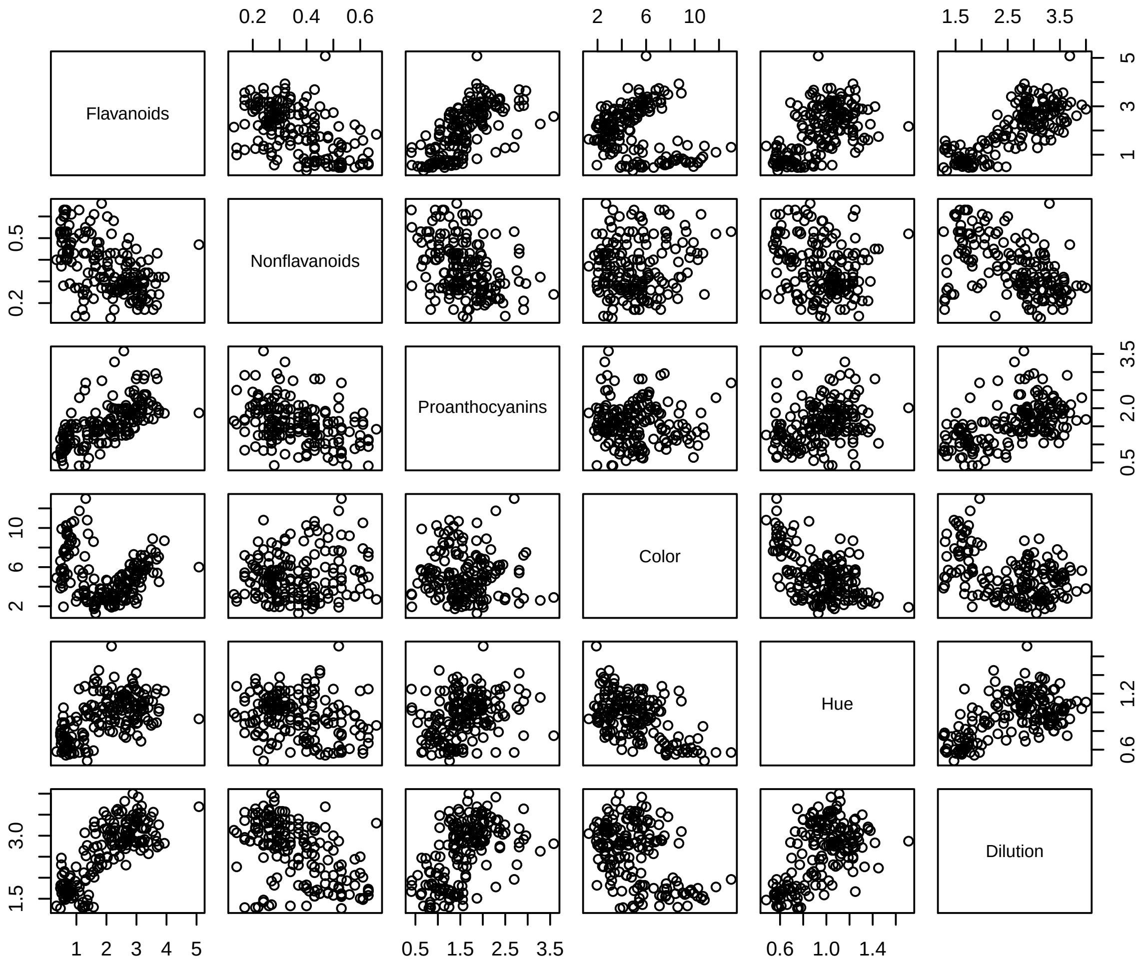

```
pie(table(wine$Type))
```




```
pairs(wine[,2:7])
```




```
pairs(wine[,8:13])
```



Linear discriminant analysis

```
wine.lda <- lda(Type ~ ., data=wine)
```

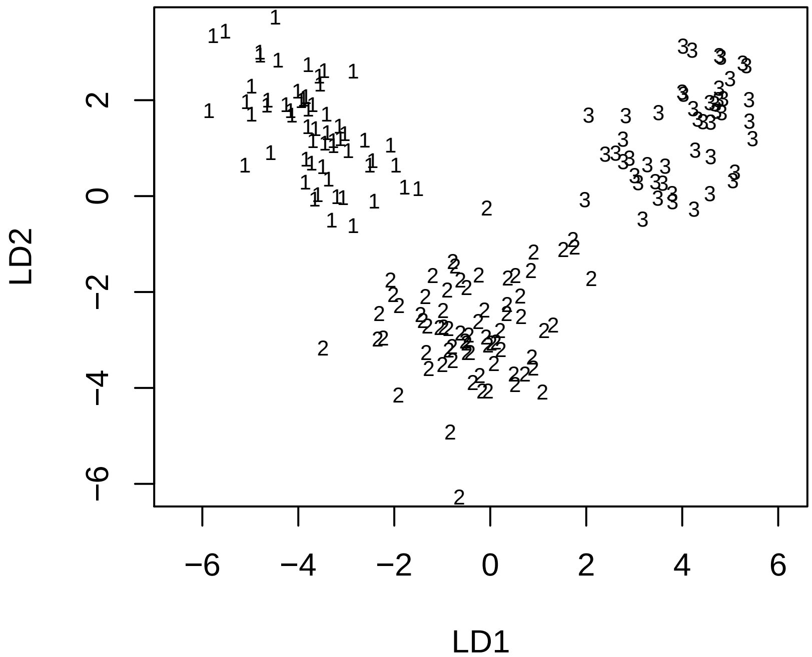
Specification . after ~ indicates to consider all the remaining variables in the dataset as covariates, without the need to insert them one by one.

```
wine.lda

## Call:
## lda(Type ~ ., data = wine)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3314607 0.3988764 0.2696629
##
## Group means:
```

```
##    Alcohol    Malic      Ash Alcalinity Magnesium  Phenols Flavanoids Nonflavanoids
## 1 13.74475 2.010678 2.455593   17.03729  106.3390 2.840169  2.9823729      0.290000
## 2 12.27873 1.932676 2.244789   20.23803   94.5493 2.258873  2.0808451      0.363662
## 3 13.15375 3.333750 2.437083   21.41667   99.3125 1.678750  0.7814583      0.447500
##   Proanthocyanins    Color       Hue Dilution   Proline
## 1        1.899322 5.528305 1.0620339 3.157797 1115.7119
## 2        1.630282 3.086620 1.0562817 2.785352  519.5070
## 3        1.153542 7.396250 0.6827083 1.683542  629.8958
##
## Coefficients of linear discriminants:
##                          LD1           LD2
## Alcohol         -0.403399781  0.8717930699
## Malic            0.165254596  0.3053797325
## Ash             -0.369075256  2.3458497486
## Alcalinity       0.154797889 -0.1463807654
## Magnesium       -0.002163496 -0.0004627565
## Phenols          0.618052068 -0.0322128171
## Flavanoids      -1.661191235 -0.4919980543
## Nonflavanoids   -1.495818440 -1.6309537953
## Proanthocyanins  0.134092628 -0.3070875776
## Color            0.355055710  0.2532306865
## Hue             -0.818036073 -1.5156344987
## Dilution        -1.157559376  0.0511839665
## Proline         -0.002691206  0.0028529846
##
## Proportion of trace:
##    LD1    LD2
## 0.6875 0.3125
```

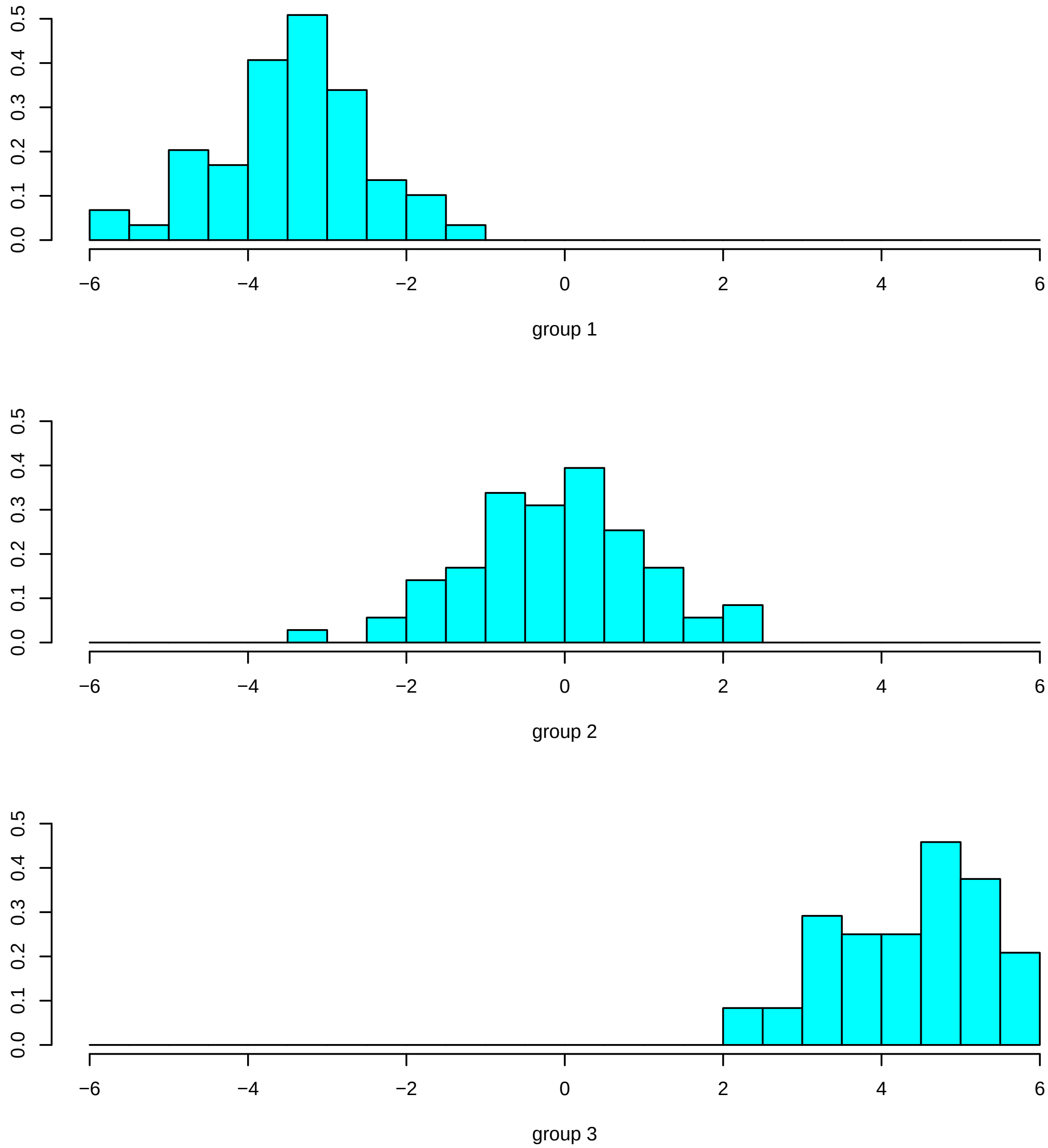
As `Type` includes three classes of wine grown, `R` computes two discriminant functions. The output includes the `proportion of trace`, that is the percentage of separation between the observations obtained from each discriminant function.

```
plot(wine.lda)
```

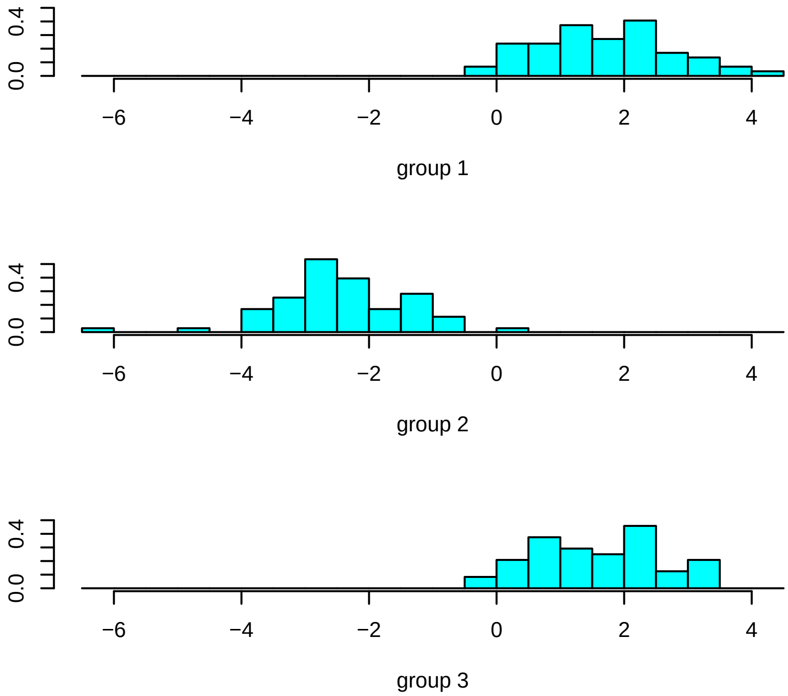


The plot shows the separation obtained by the discriminant functions: comments? Another way to check the ability of discrimination of the discriminant functions:

```
wine.previsioni <- predict(wine.lda)
ldahist(data = wine.previsioni$x[,1], g=wine$Type)
```




```r
ldahist(data = wine.previsioni$x[,2], g=wine$Type)
```



Can you relate the histograms to the dispersione plot of the two discriminant functions?