Applied Statistics and Data Analysis
Lab 6: Predictive and classification methods

Luca Grassetti and Paolo Vidoni
Department of Economics and Statistics, University of Udine

September, 2019

1 Example: advertising data

The present statistical analysis regards a data set considered in the book *An Introduction to Statistical Learning: with Applications in R* by G. James, D. Witten, T. Hastie and R. Tibshirani. This data set includes information on the values, in thousands, of

- the sales of a certain product (`Sales`);
- the advertising budgets for three different media: `TV`, `Radio` and `Newspaper`;

related to 200 different markets. The data set is saved in the `csv` (comma separated values) file `Advertising.csv` and it can be loaded using the function `read.csv`. The data set (with the first column excluded, since it corresponds to the row numbers) is assigned to the data frame `Advertising`. The file `Advertising.csv` is in the working directory and then we do not set the file path option.

```
Advertising<-read.csv(file="Advertising.csv",header=TRUE)[,-1]
str(Advertising)

'data.frame': 200 obs. of  4 variables:
 $ TV       : num  230.1 44.5 17.2 151.5 180.8 ...
 $ Radio    : num  37.8 39.3 45.9 41.3 10.8 48.9 32.8 19.6 2.1 2.6 ...
 $ Newspaper: num  69.2 45.1 69.3 58.5 58.4 75 23.5 11.6 1 21.2 ...
 $ Sales    : num  22.1 10.4 9.3 18.5 12.9 7.2 11.8 13.2 4.8 10.6 ...
```

A new function `panel.hist` is defined in order to represent the histograms, of the three interest variables, on the main diagonal of the multiple scatterplot matrix representation (see the examples in the help of function `pairs`).

Firstly, the default user specification for the extreme coordinates of the plotting region is saved in the object `usr` with the command `par("usr") = c(0,1,0,1)` and, with the further command `on.exit(par(usr))`, we assure that these graphical parameters will be recovered when the function exits. Then, the new graphical parameters are specified as an argument of function `par`. Secondly, we consider the function `hist` which, with the option `plot=FALSE`, returns a list with the breaks and the counts that can be used to obtain the histogram (in this case, the histogram is not plotted). This object is then used to obtain:

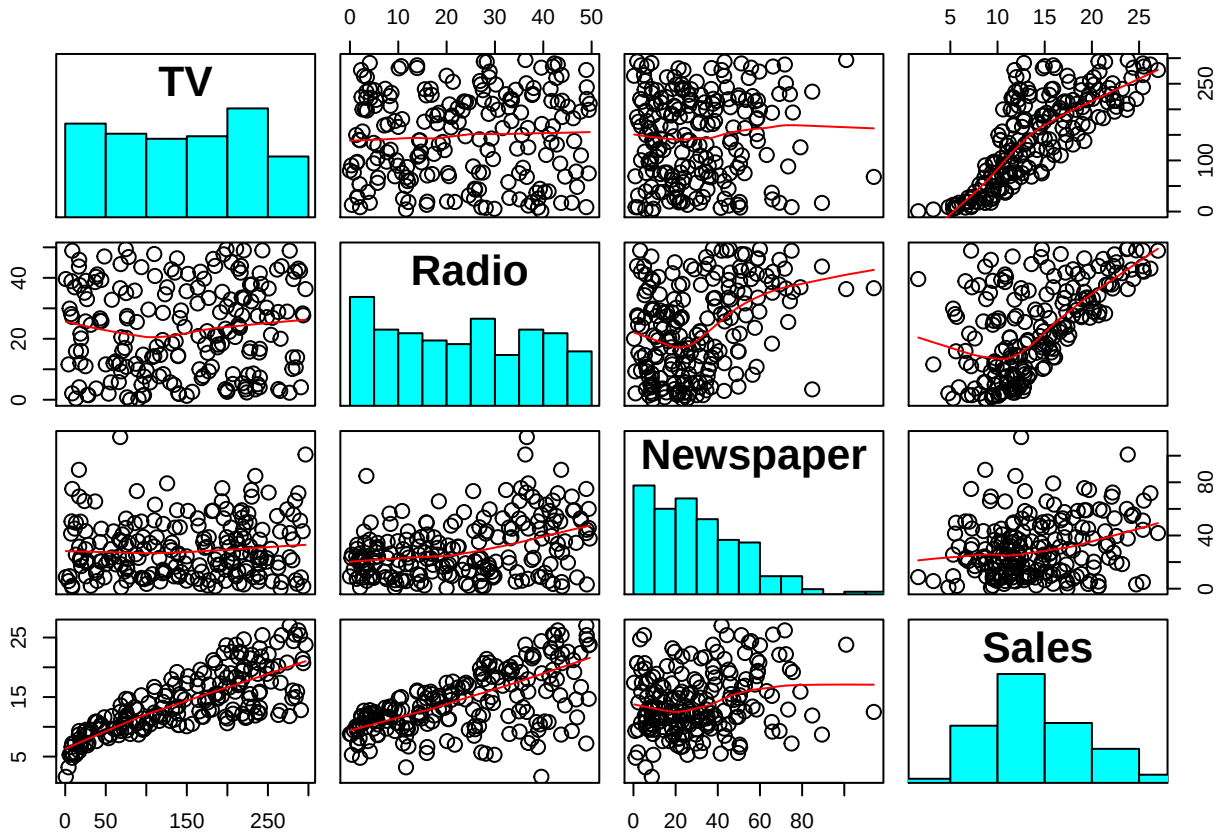
- the break points `h$breaks` of the histogram;
- the number of break points `length(breaks)` of the histogram;
- the counts `h$counts` of the observations in the classes, standardized by dividing the given values with their maximum.

Finally, the `rect` function is used to plot the histogram, with specified fill and border colors. This function draws a sequence of rectangles with the coordinates given by the options `xleft`, `ybottom`, `xright` and `ytop`; the last argument is set equal to `y`, namely the "standardized" count values for the histogram classes.

```
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", ...)
}
```

The scatterplot matrix is obtained using function `pairs`. The option `diag.panel = panel.hist` is used to plot the histograms on the diagonal and the argument `panel` is specified for defining the contents of each panel of the display (the `panel.smooth` option gives the scatterplot including the smoothing regression curve).

```
pairs(Advertising, panel = panel.smooth, cex = 1.5, pch = 1, bg = "light blue",
      diag.panel = panel.hist, cex.labels = 2, font.labels = 2)
```



The scatterplots suggest that there is a clear relationship between `Sales` and `Radio` and between `Sales` and `TV`, whereas `Sales` and `Newspaper` do not seem related. We consider a multiple linear regression model for the response `Sales`, with all the available regressors `TV`, `Radio` and `Newspaper`. The inferential results confirm that the effect of `Newspaper` on `Sales` is not statistically significant.

```
mod.adv <- lm(Sales~TV+Radio+Newspaper, Advertising)
summary(mod.adv)
```

```
Call:
lm(formula = Sales ~ TV + Radio + Newspaper, data = Advertising)

Residuals:
    Min      1Q  Median      3Q     Max
-8.8277 -0.8908  0.2418  1.1893  2.8292

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.938889   0.311908   9.422   <2e-16 ***
TV           0.045765   0.001395  32.809   <2e-16 ***
```

```
Radio        0.188530   0.008611  21.893   <2e-16 ***
Newspaper   -0.001037   0.005871  -0.177     0.86
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.686 on 196 degrees of freedom
Multiple R-squared:  0.8972,Adjusted R-squared:  0.8956
F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

```
summary(mod.adv)$sigma^2 # estimated error variance
```

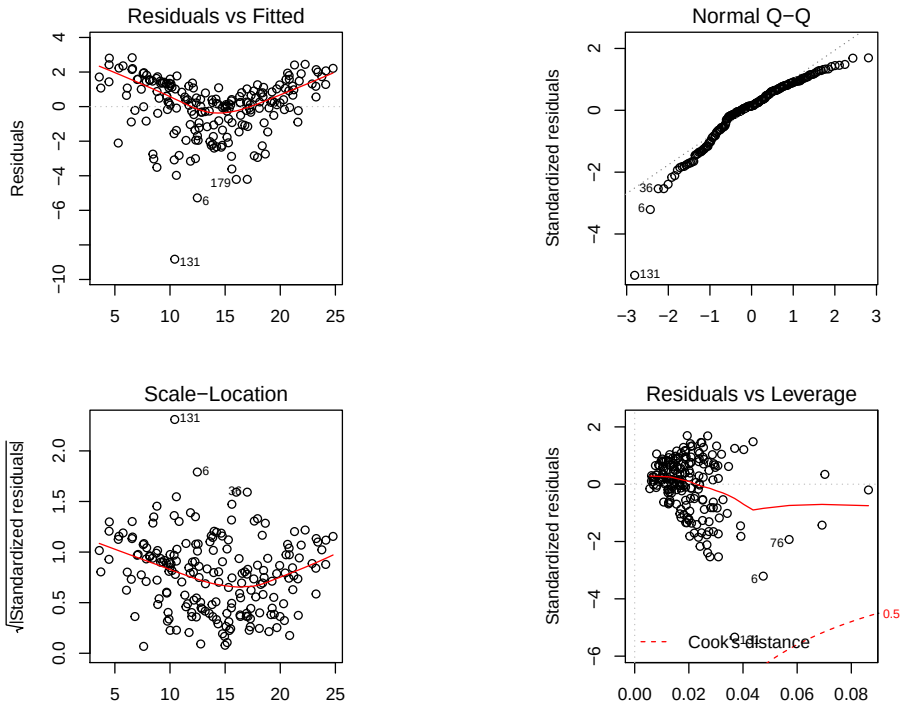
```
[1] 2.840945
```

```
AIC(mod.adv) # AIC criterion
```

```
[1] 782.3622
```

Thus, this linear regression model does not provide an adequate description for the response `sales`. This conclusion is also supported by the diagnostic plots given below, which suggest some possible nonlinear effect of the covariates.

```
par(mfrow=c(2,2), pty="s", mar=c(3,2,3,2))
plot(mod.adv)
```



```r
par(mfrow=c(1,1))
```

In order to improve the inferential description of the interest variable `Sales`, we specify a multiple regression model with the additive effect of the two regressors `TV` and `Radio`, with their interaction effect and with the quadratic effect of `TV`. The quadratic term is included in the model `formula` by using `I(TV^2)` and the interaction between `TV` and `Radio` is specified by `TV:Radio`.

```r
mod.adv1 <- lm(Sales~TV+Radio+I(TV^2)+TV:Radio, Advertising)
summary(mod.adv1)
```

```
Call:
lm(formula = Sales ~ TV + Radio + I(TV^2) + TV:Radio, data = Advertising)

Residuals:
    Min      1Q  Median      3Q     Max
-4.9949 -0.2969 -0.0066  0.3798  1.1686

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.137e+00  1.927e-01  26.663  < 2e-16 ***
TV           5.092e-02  2.232e-03  22.810  < 2e-16 ***
Radio        3.516e-02  5.901e-03   5.959 1.17e-08 ***
I(TV^2)     -1.097e-04  6.893e-06 -15.920  < 2e-16 ***
TV:Radio     1.077e-03  3.466e-05  31.061  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6238 on 195 degrees of freedom
Multiple R-squared:  0.986,Adjusted R-squared:  0.9857
F-statistic:  3432 on 4 and 195 DF,  p-value: < 2.2e-16
```

```r
summary(mod.adv1)$sigma^2 # estimated error variance
```

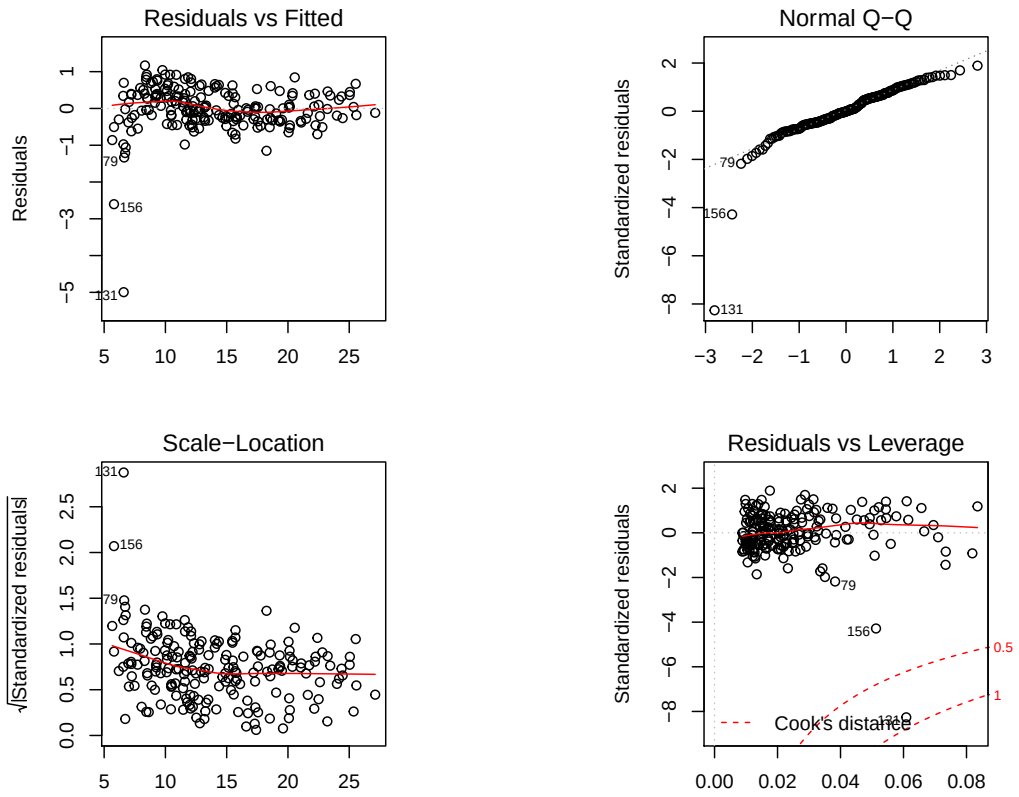
```
[1] 0.3890814
```

```r
AIC(mod.adv1) # AIC criterion
```

```
[1] 385.7185
```

We conclude that, in this case, all the regression terms are statistically significant. Indeed, with respect to the previous linear model, the multiple (adjusted) R^2 coefficient is higher, while the AIC statistic and the estimated variance are both smaller. Furthermore, the diagnostic plots assure that the present model improves the previous one, though there are some worrisome local deviations.

```
par(mfrow=c(2,2), pty="s", mar=c(3,2,3,2))
plot(mod.adv1)
```




```
par(mfrow=c(1,1))
```

Whereas in the inference framework one may be interested, for example, in finding which media contributes significantly to sales, in the prediction framework one could be interested, for example, in predicting the amount of sales given a fixed budget for the relevant media. Thus, once the multiple regression model is fitted, it can be interesting to predict the observed value for the response on the basis of a set of fixed values for the predictors. Under this respect, confidence intervals quantify the uncertainty surrounding the average sales over a large number of markets and prediction intervals can be used to asses the uncertainty surrounding sales for a particular market.

Using the function `predict`, it is almost immediate to compute both the prediction and the confidence intervals, by considering suitable fixed values for the predictors. For example, given that 100 (thousands \$) is spent on TV advertising and 20 (thousands \$) is spent on radio advertising in each market, the 95% confidence interval for the mean value of `Sales` is given by the following command.

```
intc <- predict(mod.adv1, newdata=data.frame(TV=100,Radio=20),
                interval="confidence")
intc

       fit      lwr      upr
1 11.98811 11.86426 12.11196
```

On the other hand, assuming the same spent amount for TV and radio advertising, the 95% prediction interval for the `Sales` is obtained using the option `interval="prediction"`.

```
intp <- predict(mod.adv1, newdata=data.frame(TV=100,Radio=20),
                interval="prediction")
intp

       fit     lwr      upr
1 11.98811 10.7517 13.22452
```

Note that both intervals are centered on the same value 11.988, but the prediction interval is wider than the confidence interval, reflecting the there is an increased uncertainty about sales for a given market in comparison to the average sales over many locations.

2 Example: automobile bodily injury claims

We consider a data set collected in 2002 by the Insurance Research Council, US, and regarding automobile bodily injury claims. The data are also presented in the book *Regression Modeling with Actuarial and Financial Applications* by E.W. Frees. We have 1340 observations on the following variables:

- `ATTORNEY`: claimant represented by an attorney, with values `1` (`yes`) and `2` (`no`);
- `CLMSEX`: claimant gender, with values `1` (`male`) and `2` (`female`);
- `MARITAL`: claimant marital status, with values `1` (married `M`), `2` (single `S`), `3` (widowed `W`) and `4` (divorced `D`);
- `CLMINSUR`: driver of the claimant's vehicle uninsured, with values `1` (`yes`) and `2` (`no`);
- `SEATBELT`: claimant wearing a seatbelt/child restraint, with values `1` (`yes`) and `2` (`no`);
- `CLMAGE`: claimant's age;
- `LOSS`: claimant's total economic loss (in thousands $).

The data set is saved in the file `AutoBI.csv`, which can be loaded using the function `read.csv` and assigned to the data frame `autobi`. The variables are originally specified as numerical variables and then the columns of the data frame correspond to numerical vectors, as pointed out by the output of function `str`.

```
autobi=read.csv(file="AutoBI.csv",header=TRUE)[,-1]
str(autobi)

'data.frame': 1340 obs. of  7 variables:
 $ ATTORNEY: int  1 2 2 1 2 1 1 1 2 2 ...
 $ CLMSEX  : int  1 2 1 1 1 2 1 2 2 1 ...
 $ MARITAL : int  NA 2 2 1 4 1 2 2 2 2 ...
 $ CLMINSUR: int  2 1 2 2 2 2 2 2 2 2 ...
 $ SEATBELT: int  1 1 1 2 1 1 1 1 1 1 ...
 $ CLMAGE  : int  50 28 5 32 30 35 19 34 61 NA ...
 $ LOSS    : num  34.94 10.892 0.33 11.037 0.138 ...
```

Using the functions `factor` (which encodes a vector as a factor), `levels` (which specifies the values of the factor levels) and `cut` (which divides a range into intervals and classifies the values according to which interval they fall), the first five variables are redefined as factor variables. Furthermore, the additional variable `AGECLASS` is introduced, by considering the claimant's age classified into the classes `(0,18]`, `(18,26]`, `(26,36]`, `(36,47]` and `(47,95]`, which are interpreted as the levels of a further factor variable. With the functions `str` and `summary`, we may display the structure of the new data frame and some summary statistics on the eight observed variables.

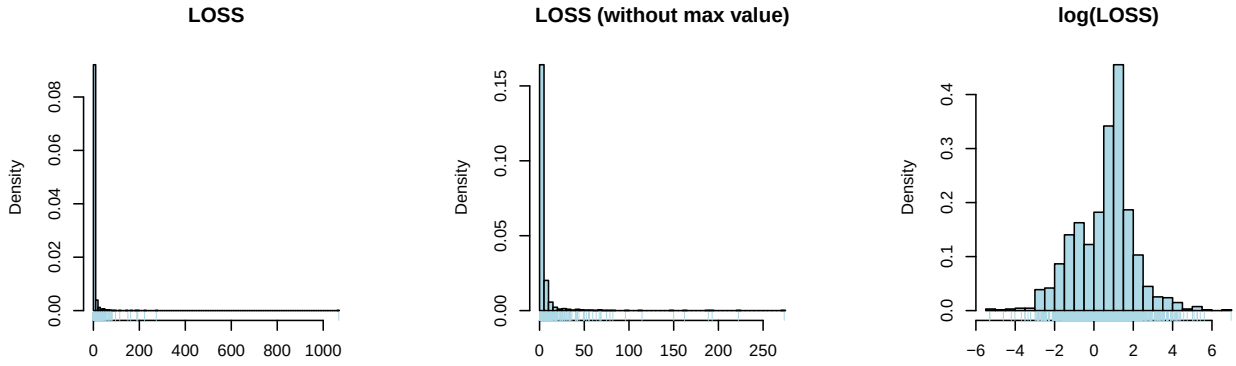
```
autobi$ATTORNEY <- factor(autobi$ATTORNEY)
levels(autobi$ATTORNEY) <- c("yes","no")
autobi$CLMSEX <- factor(autobi$CLMSEX)
levels(autobi$CLMSEX) <- c("male","female")
autobi$MARITAL <- factor(autobi$MARITAL)
levels(autobi$MARITAL) <- c("M","S","W","D")
autobi$CLMINSUR <- factor(autobi$CLMINSUR)
levels(autobi$CLMINSUR) <- c("yes","no")
autobi$SEATBELT <- factor(autobi$SEATBELT)
levels(autobi$SEATBELT) <- c("yes","no")
# the new variable AGECLASS
autobi$AGECLASS <- cut(autobi$CLMAGE, breaks=c(0,18,26,36,47,95))
levels(autobi$AGECLASS) <- c("1","2","3","4","5")
str(autobi)

'data.frame': 1340 obs. of  8 variables:
 $ ATTORNEY: Factor w/ 2 levels "yes","no": 1 2 2 1 2 1 1 1 2 2 ...
 $ CLMSEX  : Factor w/ 2 levels "male","female": 1 2 1 1 1 2 1 2 2 1 ...
 $ MARITAL : Factor w/ 4 levels "M","S","W","D": NA 2 2 1 4 1 2 2 2 2 ...
 $ CLMINSUR: Factor w/ 2 levels "yes","no": 2 1 2 2 2 2 2 2 2 2 ...
 $ SEATBELT: Factor w/ 2 levels "yes","no": 1 1 1 2 1 1 1 1 1 1 ...
 $ CLMAGE  : int  50 28 5 32 30 35 19 34 61 NA ...
 $ LOSS    : num  34.94 10.892 0.33 11.037 0.138 ...
 $ AGECLASS: Factor w/ 5 levels "1","2","3","4",..: 5 3 1 3 3 3 2 3 5 NA ...
```

```
summary(autobi)
```

```
 ATTORNEY     CLMSEX    MARITAL    CLMINSUR    SEATBELT
 yes:685   male  :586   M   :624   yes : 120   yes :1270
 no :655   female:742   S   :650   no  :1179   no  :  22
           NA's  : 12   W   : 15   NA's:  41   NA's:  48
                        D   : 35
                        NA's: 16


     CLMAGE           LOSS          AGECLASS
 Min.   : 0.00   Min.   :   0.005   1   :246
 1st Qu.:19.00   1st Qu.:   0.640   2   :211
 Median :31.00   Median :   2.331   3   :222
 Mean   :32.53   Mean   :   5.954   4   :243
 3rd Qu.:43.00   3rd Qu.:   3.995   5   :215
 Max.   :95.00   Max.   :1067.697   NA's:203
 NA's   :189
```

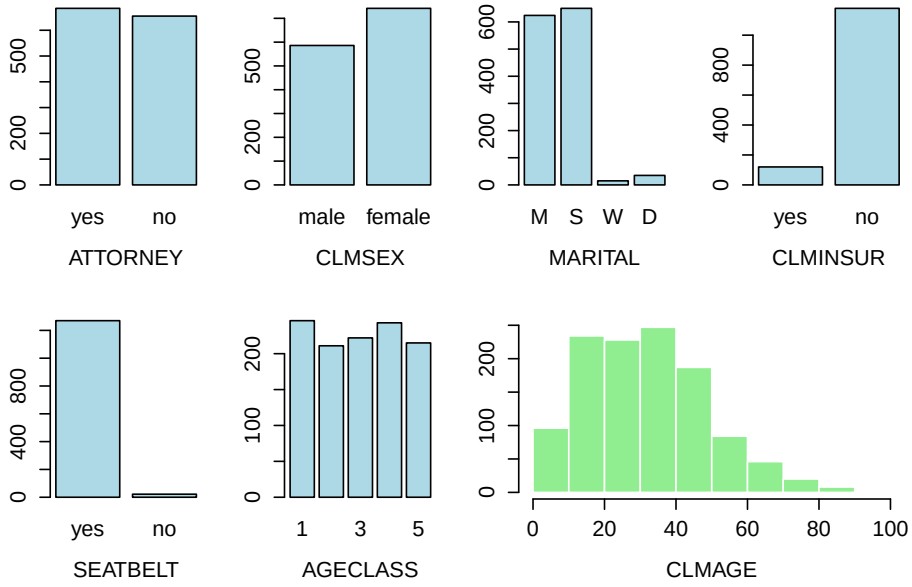
The aim of the analysis is to specify a statistical model for predicting claim amounts in future policies, based on a sample of claim amounts of past policies. The response variable is `LOSS`, namely the claimant's total economic loss, which is analyzed using the logarithmic transformation, as motivated by the following graphical analysis. In a single graphical device with three panels, we represent the histogram based on the original data set, the histogram based on the data set without the maximum value (which is an extremely large observation) and the histogram based on the log transformed data. The histograms are obtained using the function `hist.scott` of the library `MASS`, where an automatic bin width selection is performed using the Scott formula. In order to exclude the maximum value, we consider the reduced data set `autobi$LOSS[-which.max(autobi$LOSS)]`, where the `which.max` (`which.min`) function identifies the location of the maximum (minimum) inside a numeric vector. Indeed, the `rug` function adds to an existing plot the location of the data points on the x-axis. The `col` argument is used in the graphical functions for defining the color of the representation (a list of colors available in R can be found at www.research.stowers-institute.org/efg/R/Color/Chart/).

```
library(MASS)
par(mfrow=c(1,3), pty="s")
hist.scott(autobi$LOSS, main="LOSS", xlab="", col="lightblue")
rug(autobi$LOSS, col="lightblue")
hist.scott(autobi$LOSS[-which.max(autobi$LOSS)], main="LOSS (without max value)",
           xlab="",col="lightblue")
rug(autobi$LOSS[-which.max(autobi$LOSS)],col="lightblue")
hist.scott(log(autobi$LOSS), main="log(LOSS)",xlab="",col="lightblue")
rug(log(autobi$LOSS),col="lightblue")
```



The frequency distribution of the potential explanatory variables is represented in a graphical device with seven panels. The `layout` function divides the device into as many rows and columns as there are in the matrix given in the argument. Here, the last plot occupies the last two panels, which have the same label number 7.

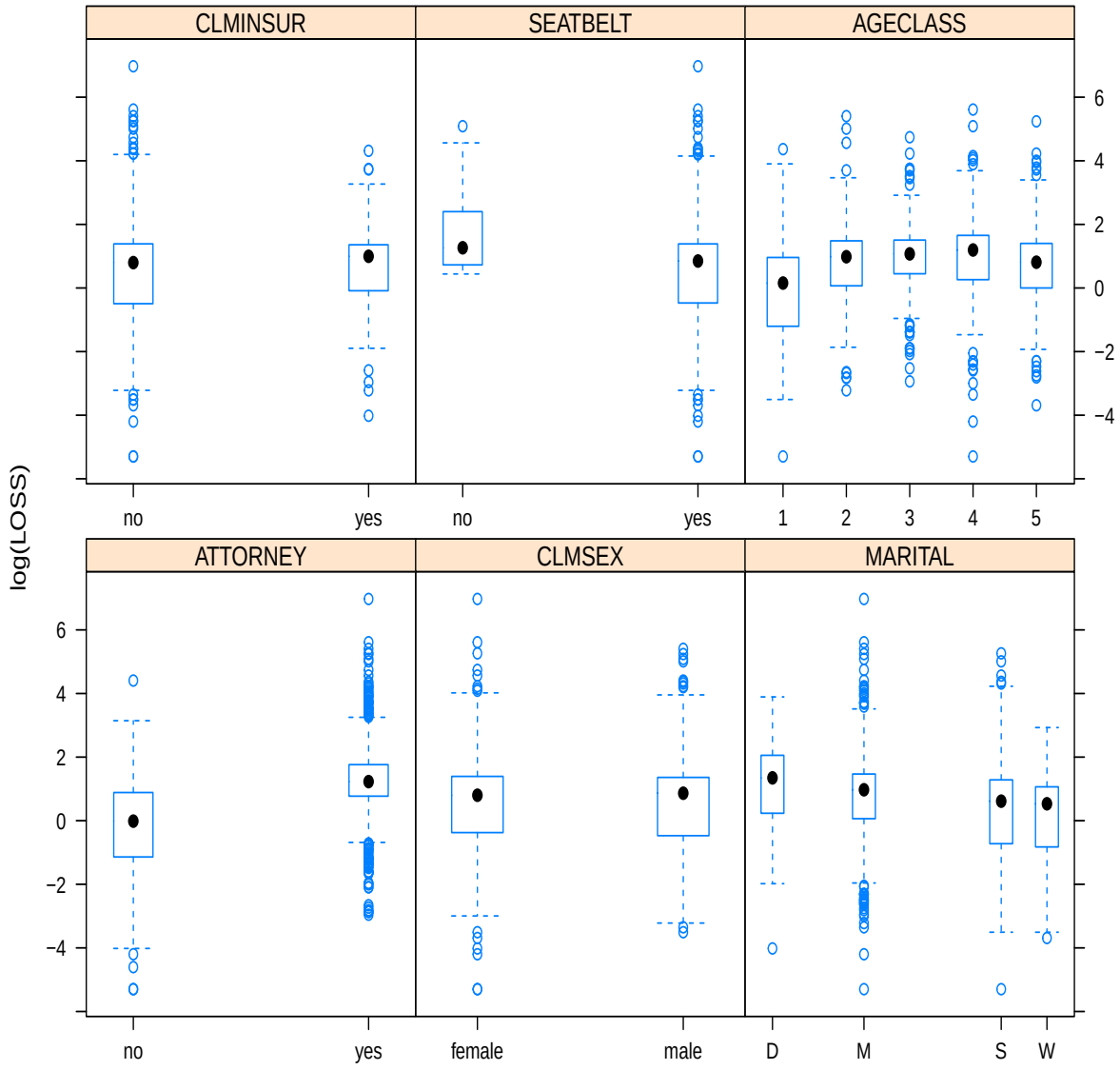
```
par(cex.axis=1.3,cex.lab=1.3, mar=c(5,3,1.5,1))
layout(matrix(c(1:7,7),byrow=TRUE,nrow=2))
ind <- c(1,2,3,4,5,8)
for (i in 1:6)
   barplot(table(autobi[,ind[i]]) , xlab=names(autobi)[ind[i]],
                col="lightblue")
hist(autobi$CLMAGE,col="lightgreen",xlab="CLMAGE",ylab="",main="",border="white")
```



In order to investigate the relationship between the response variable and the factor explanatory variables, the frequency distribution of `log(LOSS)` is represented conditionally on each factor variable. The `bwplot` function of the library `lattice` is used to represent a matrix with the corresponding conditional boxplots. The first argument is defined as a `formula` object. Indeed, the

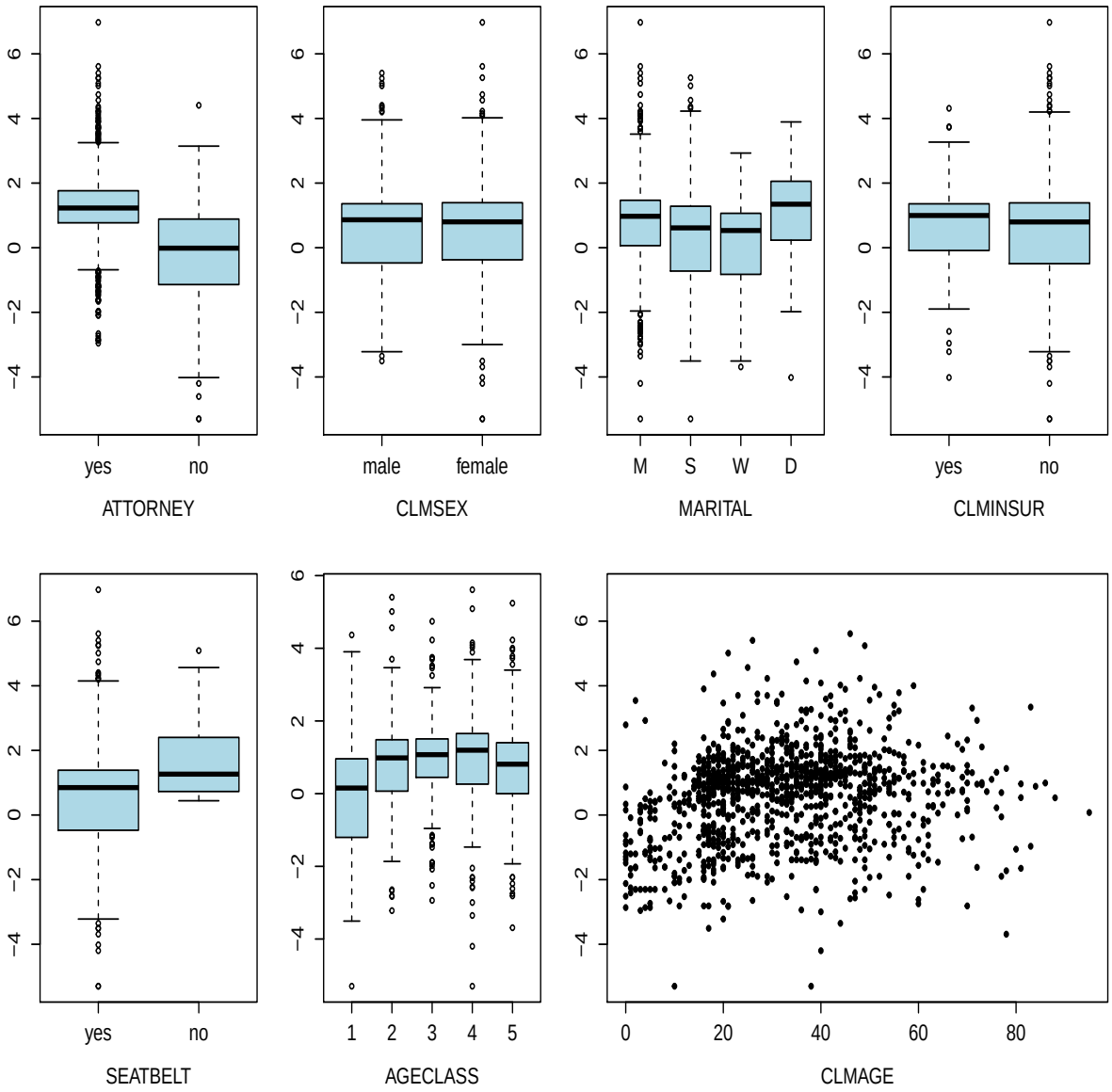
`layout` argument defines the number of rows and columns in the graphical device, the `aspect` argument controls the aspect ratio of each panel, the option `outer=TRUE` allows to obtain separated plots for each factor and the `scales` argument is used to determine the axes labels.

```
library(lattice)
bwplot(log(LOSS)~ ATTORNEY + CLMSEX + MARITAL + CLMINSUR + SEATBELT + AGECLASS,
       data=autobi, ylab="log(LOSS)", outer = TRUE, scales = list(x = "free"),
       xlab="", layout=c(3,2), main="", aspect="fill")
```



A better representation of the conditional distribution of `log(LOSS)` can be obtained with the following lines of code, which define a suitable graphical device with seven panels. The relationship between `CLMAGE` and `log(LOSS)` is described by the scatterplot occupying the last two panels, whereas the `boxplot` function is used to obtain the six conditional boxplots.

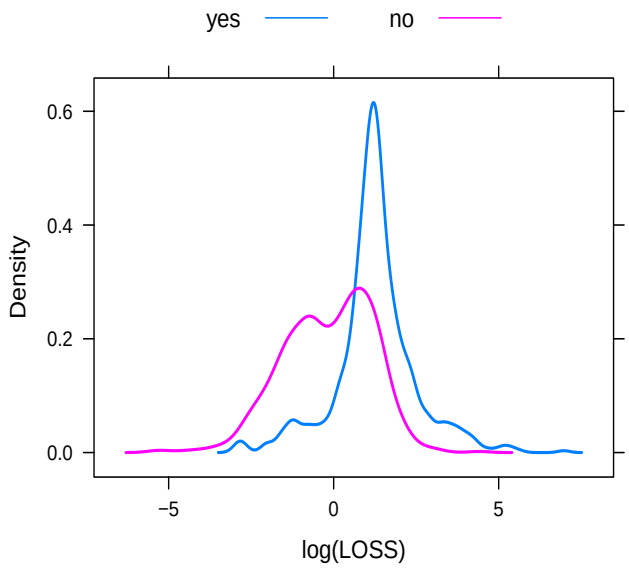
```
par(cex.axis=1.3,cex.lab=1.3, mar=c(5,3,1.5,1))
layout(matrix(c(1:7,7),byrow=TRUE,nrow=2))
ind <- c(1,2,3,4,5,8)
for (i in 1:6)
   boxplot(log(autobi$LOSS)~ autobi[,ind[i]] , xlab=names(autobi)[ind[i]],
                col="lightblue")
plot(autobi$CLMAGE,log(autobi$LOSS),xlab="CLMAGE",ylab="",main="",pch=20)
```



The effect of `CLMSEX` and `CLMINSUR` on `log(LOSS)` does not seem relevant and the potential effect of `CLMAGE` is not clear. On the other hand, the variables that may be relevant for `log(LOSS)` are `SEATBELT`, `MARITAL`, `AGECLASS` and `ATTORNEY`. In particular, being represented by an attorney seems to be very important, as confirmed by the following plot where the probability distribution of `log(LOSS)` is represented by considering the kernel density estimates conditioned on the values

of the factor `ATTORNEY`. We use the function `densityplot` of the library `lattice`, where the first argument indicates the variable on which the procedure is applied, the `group` argument specifies the grouping variable and the option `plot.points=FALSE` excludes the representation of the data points on the x-axis.

```
library(lattice)
densityplot(~log(LOSS), group=ATTORNEY, data=autobi, lwd=2,
       xlab="log(LOSS)",  plot.points=FALSE, auto.key=list(columns=2))
```



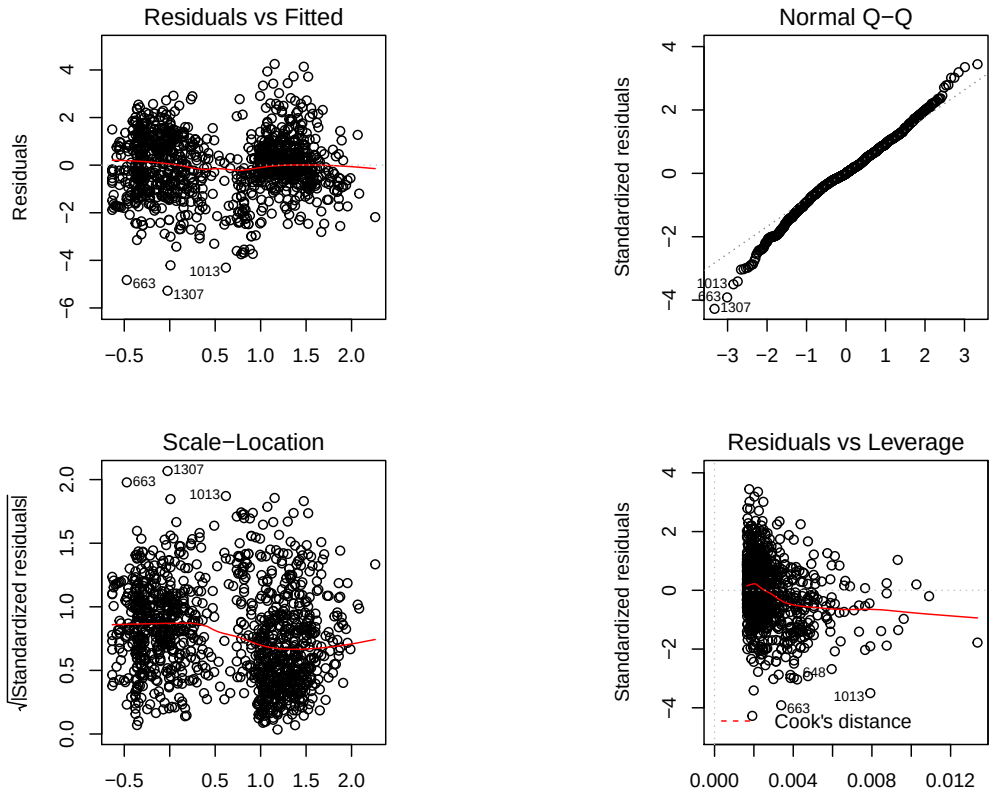
The effects of `CLMAGE` and `ATTORNEY` (factor variable coded with a single dummy variable and assuming the level `yes` as reference level) on the response `logLOSS` can be described using the following multiple linear regression model. The linear model estimation for the `log(LOSS)` as a function of `CLMAGE` and `ATTORNEY` is obtained and the results are summarized below.

```
mod <- lm( log(LOSS) ~ ATTORNEY + CLMAGE, autobi)
summary(mod)


Call:
lm(formula = log(LOSS) ~ ATTORNEY + CLMAGE, data = autobi)

Residuals:
    Min      1Q  Median      3Q     Max
-5.2750 -0.6596  0.0059  0.7757  4.2502

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.737636   0.085129   8.665  < 2e-16 ***
ATTORNEYno  -1.369938   0.072940 -18.782  < 2e-16 ***
```

```
CLMAGE       0.016025   0.002132   7.517 1.13e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.235 on 1148 degrees of freedom
  (189 observations deleted due to missingness)
Multiple R-squared:  0.2607,Adjusted R-squared:  0.2594
F-statistic: 202.4 on 2 and 1148 DF,  p-value: < 2.2e-16
```

The model fitting assures that both the regression coefficients are strongly significant, pointing to a relevant effect of both the explanatory variables on the mean response. The dummy variable flags those observations that have the level of `ATTORNEY=no` and, since its estimated coefficient is negative, subjects without an attorney have a smaller mean response. Notice that 189 observations were deleted due to missingness. Indeed, the diagnostic plots obtained using the function `plot` are moderately good and, in particular, emphasize the clustering induced by the factor regressor `ATTORNEY`.

```
par(mfrow=c(2,2), pty="s", mar=c(3,2,3,2))
plot(mod)
```



```
par(mfrow=c(1,1))
```

The effect of the factor `ATTORNEY` may be easily described by considering the estimate of the mean total economic loss and the associated 95% confidence interval, both in the log and in the original scale, for an individual of age 30, in case of `ATTORNEY=yes` and `ATTORNEY=no`, respectively.

```
ci_yes<-predict(mod, newdata=data.frame(ATTORNEY="yes",CLMAGE=30,SEATBELT="yes"),
                interval="confidence")
ci_yes # age=30 and attorney=yes, log scale

       fit      lwr      upr
1 1.218397 1.119567 1.317228

exp(ci_yes) # age=30 and attorney=yes, original scale

       fit      lwr     upr
1 3.381764 3.063526 3.73306

ci_no<-predict(mod, newdata=data.frame(ATTORNEY="no",CLMAGE=30,SEATBELT="yes"),
               interval="confidence")
ci_no # age=30 and attorney=no, log scale

         fit        lwr         upr
1 -0.1515402 -0.2561221 -0.04695834

exp(ci_no) # age=30 and attorney=no, original scale

        fit       lwr       upr
1 0.8593833 0.7740474 0.9541271
```

With the aim of improving the model specification, we consider also the factor regressor `SEATBELT` (coded with a single dummy variable and assuming the level `yes` as reference level) and the quadratic effect of `CLMAGE`. We use the `lm` function with the argument `subset`, in order to select only the complete observations in the data frame. The function `complete.cases` returns a logical vector indicating which cases are complete, namely which observations are without missing values.

```
mod2s <- lm(log(LOSS) ~ATTORNEY + CLMAGE + I(CLMAGE^2) + SEATBELT,
            data=autobi,  subset = complete.cases(autobi))
summary(mod2s)


Call:
lm(formula = log(LOSS) ~ ATTORNEY + CLMAGE + I(CLMAGE^2) + SEATBELT,
    data = autobi, subset = complete.cases(autobi))
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-5.5581 -0.6537 -0.0012  0.7394  4.0894

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.249e-01  1.376e-01  -1.635  0.10244
ATTORNEYno  -1.352e+00  7.247e-02 -18.659  < 2e-16 ***
CLMAGE       8.283e-02  7.498e-03  11.046  < 2e-16 ***
I(CLMAGE^2) -9.076e-04  9.537e-05  -9.517  < 2e-16 ***
SEATBELTno   9.241e-01  2.681e-01   3.446  0.00059 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.185 on 1073 degrees of freedom
Multiple R-squared:  0.3233,Adjusted R-squared:  0.3208
F-statistic: 128.2 on 4 and 1073 DF,  p-value: < 2.2e-16

AIC(mod2s)

[1] 3432.361

AIC(mod)

[1] 3757.804
```

The values for the (adjust) multiple R^2 coefficient, the residual standard error (square root of the error variance) and the AIC model selection statistic are smaller in the new model.

In order to evaluate the predictive accuracy of a given model, we may compute the training mean squared error (MSE), which corresponds to the residual sample variance, obtained by considering the total number of observations instead of the degrees of freedom. However, since this measure gives an overoptimistic predictive assessment of the model, it is better to compute the test MSE, where the model is fitted using the training observations and the prediction accuracy is evaluated by considering the test observations. An estimate for the test MSE can be obtained using a suitable cross-validation procedure. Using the function `cv.glm` of the library `boot` we may compute the estimated K-fold cross-validation prediction error (test MSE). The required first argument is an object of class `"glm"`, and for this reason the two multiple linear regression models are now refitted using function `glm`. Indeed, the optional argument `K` is used to specify the number of groups (folds); the default value is the sample size, corresponding to the leave-one-out cross-validation procedure. The result is a list, where the element `delta` is two-dimensional, which second component is the adjusted cross-validation estimate.

```
library(boot)
# data frame with complete observations
autobi_complete<-autobi[complete.cases(autobi),]
mod_bis <- glm(log(LOSS) ~ATTORNEY + CLMAGE, data=autobi_complete)
mod2s_bis <- glm(log(LOSS) ~ATTORNEY + CLMAGE + I(CLMAGE^2) + SEATBELT,
                 data=autobi_complete)
sum(resid(mod_bis)^2)/length(autobi_complete[,1]) # training MSE for mod_bis

[1] 1.532974

cv.glm(autobi_complete,mod_bis)$delta[2] # cv estimate test MSE for mod_bis

[1] 1.542337

sum(resid(mod2s_bis)^2)/length(autobi_complete[,1]) # training MSE for mod2s_bis

[1] 1.39794

cv.glm(autobi_complete,mod2s_bis)$delta[2] # cv estimate test MSE for mod2s_bis

[1] 1.412208
```

Using the new fitted model, we may determine the 95% confidence interval for the mean total economic loss and the 95% prediction interval for the total economic loss, both in the log and in the original scale, for an individual of age 30, with `ATTORNEY=yes` and `SEATBELT="yes"`.

```
intc <- predict(mod2s, newdata=data.frame(ATTORNEY="yes",CLMAGE=30,
                SEATBELT="yes"), interval="confidence")
intc # log scale

       fit      lwr      upr
1 1.443131 1.334347 1.551915

exp(intc) # original scale

       fit      lwr      upr
1 4.233932 3.797515 4.720502

intp <- predict(mod2s, newdata=data.frame(ATTORNEY="yes",CLMAGE=30,
                SEATBELT="yes"), interval="prediction")
intp # log scale

       fit        lwr      upr
1 1.443131 -0.8847818 3.771044

exp(intp) # original scale

       fit       lwr      upr
1 4.233932 0.4128042 43.42537
```

With the additional argument `pred.var=1.4122`, the prediction intervals are computed by considering the prediction variance obtained using the cross-validation procedure.

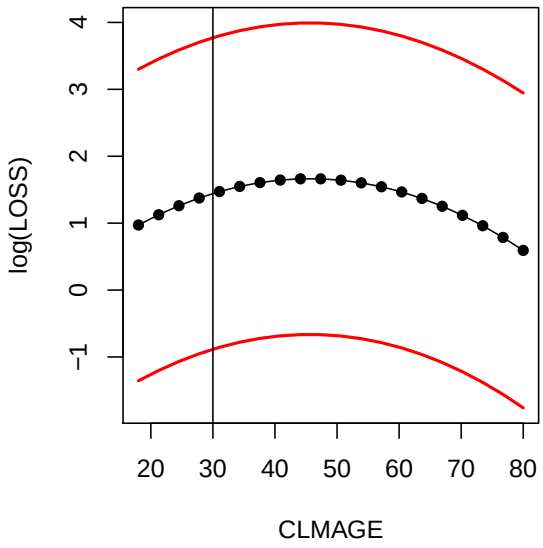
```
intp <- predict(mod2s, newdata=data.frame(ATTORNEY="yes",CLMAGE=30,
                SEATBELT="yes"), interval="prediction", pred.var=1.4122)
intp # log scale

       fit        lwr      upr
1 1.443131 -0.8911786 3.777441

exp(intp) # original scale

       fit      lwr      upr
1 4.233932 0.410172 43.70404
```

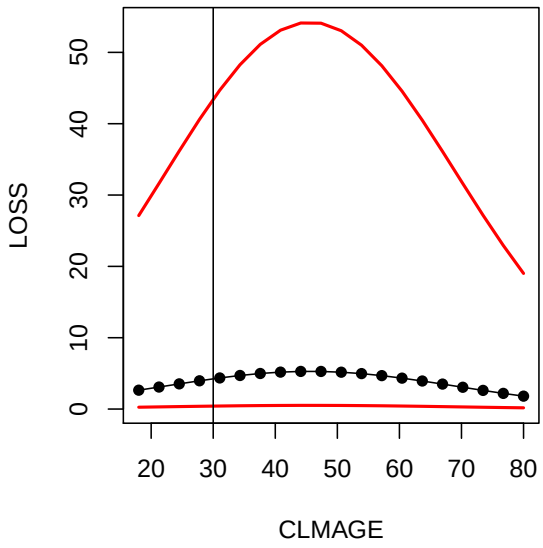
Finally, it is possible to compute the 95% prediction intervals for `log(LOSS)` and `LOSS`, for an insured with `ATTORNEY=yes`, `SEATBELT="yes"` and age ranging from 18 to 80 years. To this end, a sequence of prediction intervals is computed for a set of 20 different ages between 18 and 80 and it is saved in the matrix `matout`.

```
age <- seq(18, 80, l=20)
matout <- matrix(0, nrow=20, ncol=3)
for(i in 1:length(age))
matout[i,] <- predict(mod2s, newdata=data.frame(ATTORNEY="yes",CLMAGE=age[i],
                SEATBELT="yes"), interval="prediction")
```

In order to represent, in two different panels, the results in the logarithmic and in the original measurement scales, we use function `plot` (with the option `type="l"`) to draw the estimated mean value as a function of `CLMAGE`, function `points` to add the points corresponding to the 20 estimated means and function `lines` to add the lines identifying the upper and the lower prediction limits. The vertical line, obtained with the command `abline(v=30)`, identifies the prediction interval given before for an insured of age 30, represented by an attorney and wearing a seatbelt.

```
par(mfrow=c(1,2), pty="s")
plot(age, matout[,1], pch=16, type="l", ylim=range(matout), xlab="CLMAGE",
     ylab="log(LOSS)")
points(age, matout[,1], pch=16)
lines(age, matout[,2], col=2, lwd=2)
lines(age, matout[,3], col=2, lwd=2)
abline(v=30)
plot(age, exp(matout[,1]), pch=16, type="l", ylim=range(exp(matout)),
     xlab="CLMAGE", ylab="LOSS")
points(age, exp(matout[,1]), pch=16)
lines(age, exp(matout[,2]), col=2, lwd=2)
lines(age, exp(matout[,3]), col=2, lwd=2)
abline(v=30)
```





3 Example: credit scoring

We consider a data set, presented in the book *Regression: Models, Methods and Applications* by L. Fahrmeir, Th. Kneib, S. Lang and B. Marx, on 1000 private credits issued by a German bank. The following variables are considered:

- `y`: the client paid back his loan, with values `1` (`no`) and `0` (`yes`);
- `account`: the running account, with values `0` (`no account`), `1` (`bad running account`) and `2` (`good running account`);
- `duration`: duration of the credit (in months);
- `amount`: credit amount (in thousands euros);
- `moral`: previous payment behavior, with values `0` (`bad`) and `1` (`good`);
- `intuse`: intended use, with values `0` (`business`) and `1` (`private`).

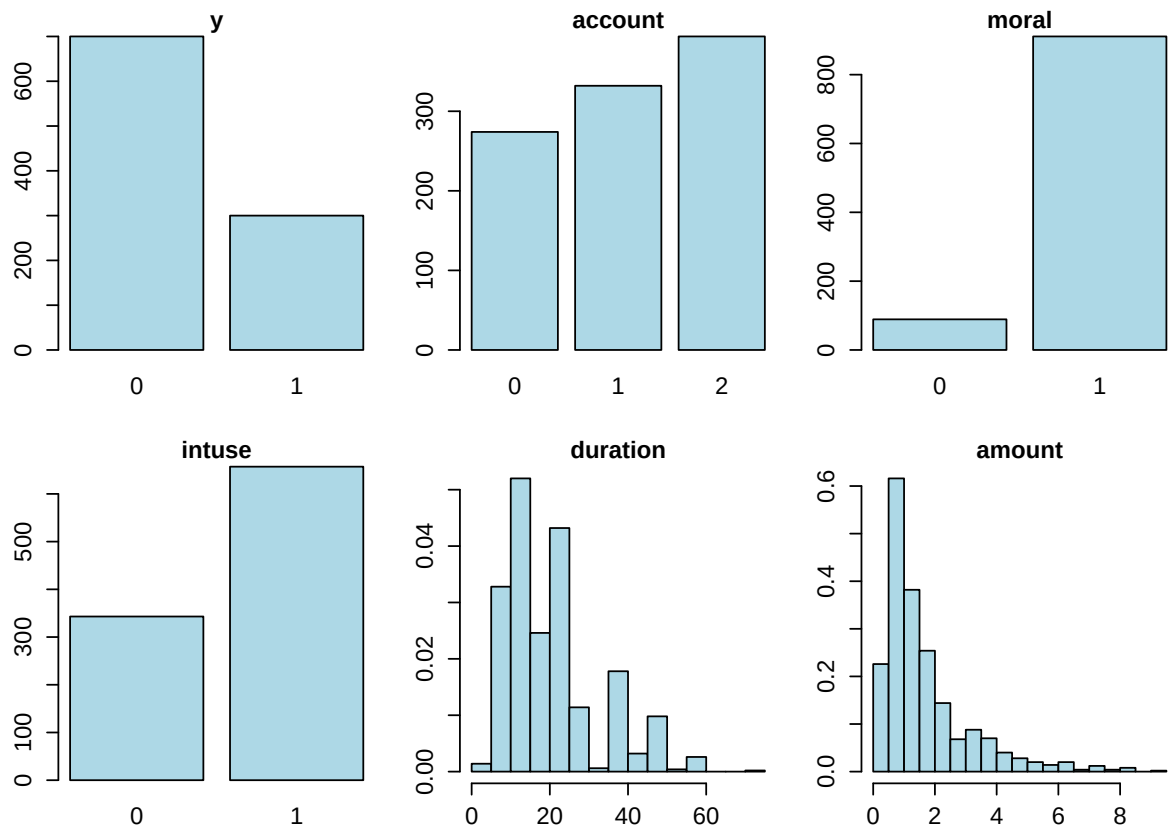
A typical problem arising in the credit scoring setting, and of great interest for a bank loan service, is to evaluate the possible loan insolvency for a customer, and then decide whether the customer will default or the customer will pay back. With this aim, we analyze the available training data, where every client is associated with a binary response variable `y` and five explanatory variables.

The data set is saved in the file `Scoring.txt`, which can be loaded using the function `read.table` and assigned to the data frame `Scoring`. Since also the categorical variables are specified as numeric variables, they are transformed into factors. Indeed, the new factor variable `account` is defined.

```
Scoring <- read.table(file="Scoring.txt",header=TRUE)
# acc1=1 (no runnig account) acc1=0 (good or bad running account)
# acc2=1 (good running account) acc2=0 (no or bad running account)
Scoring$account <- 1 - Scoring$acc1 + Scoring$acc2
Scoring$account <- factor(Scoring$account)
Scoring$moral <- factor(Scoring$moral)
Scoring$intuse <- factor(Scoring$intuse)
Scoring$y <- factor(Scoring$y)
str(Scoring)

'data.frame': 1000 obs. of  8 variables:
 $ y       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ acc1    : int  1 0 0 0 0 0 0 0 1 0 ...
 $ acc2    : int  0 0 1 1 1 1 1 1 0 1 ...
 $ duration: int  24 12 18 12 24 24 36 24 21 10 ...
 $ amount  : num  1.512 0.728 1.049 2.39 1.523 ...
 $ moral   : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
 $ intuse  : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 2 1 2 ...
 $ account : Factor w/ 3 levels "0","1","2": 1 2 3 3 3 3 3 3 1 3 ...
```

Firstly, the frequency distributions of the six variables involved in the study are represented using barplots and histograms, in a graphical device with six panels. Here we consider the `with` function which is adopted in order to define the data frame (given as first argument) where the R expression (given as second argument) has to be evaluated.

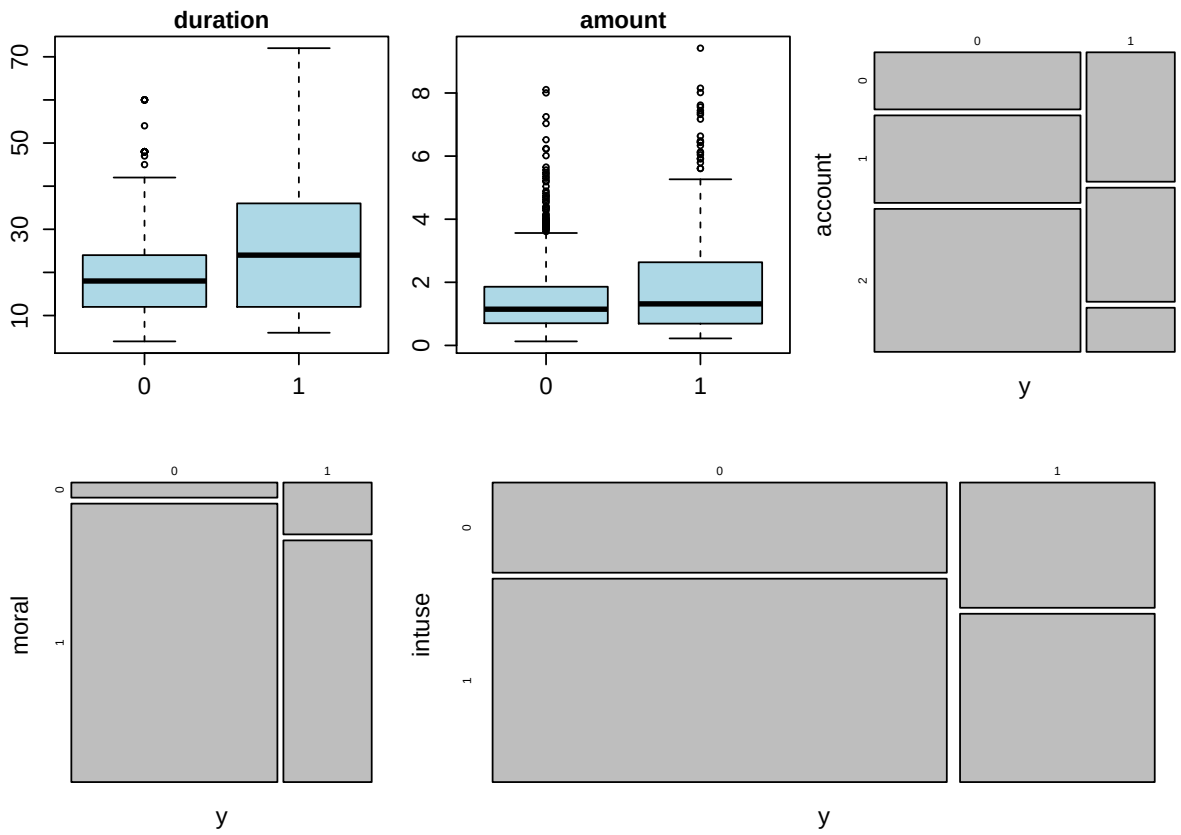
```
par(mfrow=c(2,3), pty="s")
par(cex.axis=1.3,cex.lab=1.3,mar=c(3.5,2,1.5,1))
with(Scoring, barplot(table(y),main="y",col="lightblue"))
with(Scoring, barplot(table(account),main="account",col="lightblue"))
with(Scoring, barplot(table(moral),main="moral",col="lightblue"))
with(Scoring, barplot(table(intuse),main="intuse",col="lightblue"))
with(Scoring, hist.scott(duration,main="duration", xlab="", col="lightblue") )
with(Scoring, hist.scott(amount,main="amount", xlab="", col="lightblue") )
```




```
par(mfrow=c(1,1))
```

Secondly, with the following lines of code, we aim at analyzing the potential relationships between the binary response and the explanatory variables: the distributions of `duration` and `amount` conditional to the pay-back response variable are represented using boxplots and the joint distribution of the pay-back and the other three factors is described using mosaic-plots (obtained with the `plot` function applied to a contingency table).

```
layout(matrix(c(1:5,5),byrow=TRUE,nrow=2))
par(cex.axis=1.3,cex.lab=1.3,mar=c(3.5,2,1.5,1))
with(Scoring, boxplot(duration~y, main="duration", col="lightblue"))
with(Scoring, boxplot(amount~y, main="amount", col="lightblue"))
with(Scoring, plot(table(y, account),main=""))
with(Scoring, plot(table(y, moral),main=""))
with(Scoring, plot(table(y, intuse),main=""))
```




```
par(mfrow=c(1,1))
```

A multiple logistic regression model, including all the five predictors, is fitted using the `glm` function with the option `family=binomial` and it is used for classification on the same training data. The estimated probabilities obtained with the function `predict` are transformed into dichotomous prediction by considering the threshold 0.5, so that values greater than 0.5 correspond to `y=1` (`not creditworthy`) and values equal or lower than 0.5 to `y=0` (`creditworthy`). To summarize the predictive results and to evaluate the predictive performance, a confusion matrix is calculated in order to obtain the cross-classification of the observed and the predicted frequencies.

```
mod1 <- glm(y~account+duration+amount+moral+intuse, family=binomial, data=Scoring)
prediction1 <- ifelse(predict(mod1, Scoring, type='response') > 0.5, 1, 0)
observed <- Scoring$y
confusion <- table(prediction1, observed)
colnames(confusion) <- c("creditworthy","not creditworthy")
rownames(confusion) <- c("creditworthy","not creditworthy")
confusion
```

```
                observed
prediction1      creditworthy not creditworthy
  creditworthy             645              192
  not creditworthy          55              108
```

The same results can be obtained using the function `confusionMatrix` of the library `crossval`, which computes, despite its name, the vector with the elements of the confusion matrix. Thus, it counts the number of false positives (`FP`), true positives (`TP`), true negatives (`TN`) and false negatives (`FN`). Notice that, in this context, the positivity corresponds to the not creditworthy state (`y=1`). The required arguments are, respectively, the vector containing the correct labels, the vector containing the predicted labels and the label of the negative state.

```
library(crossval)
cm <- confusionMatrix(Scoring$y, prediction1, negative="0")
cm

 FP  TP  TN  FN
 55 108 645 192
attr(,"negative")
[1] "0"
```

In order to evaluate the predictive ability of the model, we may compute the sensitivity (true positive rate) `sens=TP/(TP+FN)`, the specificity (true negative rate) `spec=TN/(FP+TN)` and the overall accuracy (total accuracy rate) `acc=(TP+TN)/(FP+TN+TP+FN)`. Indeed, also the positive predictive value `ppv = TP/(FP+TP)`, the negative predictive value `npv = TN/(TN+FN)` and the log-odds ratio `lor = log(TP*TN/(FN*FP))` can be calculated. These values may be readily obtained using the function `diagnosticErrors` of the library `crossval` by setting as argument a vector containing the `FP`, `TP`, `TN`, `FN` counts, as computed by `confusionMatrix`.

```
de <- diagnosticErrors(cm)
de

      acc      sens      spec       ppv       npv       lor
0.7530000 0.3600000 0.9214286 0.6625767 0.7706093 1.8865530
attr(,"negative")
[1] "0"
```

The model seems to be satisfactory only for predicting the customers not at risk of defaulting since the true negative rate (the proportion of negatives that are correctly identified) is 0.921, whereas the true positive rate (the proportion of positives that are correctly identified) is 0.360 and the total accuracy rate (the proportion of overall correct classifications) is 0.753. Moreover, since the classification is performed on the training data, these measures of accuracy might give an overoptimistic predictive assessment of the model. For this reason, a more reliable evaluation can be obtained using a suitable K-fold cross-validation procedure.

Firstly, the commands to be use in the procedure are collected in a single function `predfun.glm.`, which has to be a function of the form `predfun(Xtrain, Ytrain, Xtest, Ytest, ...)`. Secondly, the `crossval` function of the library `crossval` is used by considering the following arguments: the prediction function `predfun.glm`, the matrix of regressors `Scoring[,4:8]`, the values of the

response variable `Scoring$y`, the number of folds `K=10`, the option `B=1`, allowing a single repetition, and the option `verbose=FALSE`, to avoid the print of the status messages.

```
predfun.glm = function(train.x, train.y, test.x, test.y, negative)
{
  train.data <- data.frame(train.x, train.y)
  glm.fit <- glm(train.y~., binomial, train.data)
  ynew <- predict(glm.fit, newdata=data.frame(test.x, test.y), type="response")
  ynew <- as.numeric(ynew>0.5)
  out <- confusionMatrix(test.y, ynew, negative=negative)
  return(out)
}
set.seed(1992)
cv.out <- crossval(predfun.glm, Scoring[,4:8], Scoring$y, K=10, B=1,
                   negative="0", verbose=FALSE)
```

Finally, the cross-validation results, and in particular the values returned by function `predfun` averaged over all the cross-validation runs (`cv.out$stat`), are used to obtain the diagnostic errors using function `diagnosticErrors`.

```
de.cv<-diagnosticErrors(cv.out$stat)
de.cv

      acc      sens      spec       ppv       npv       lor
0.7490000 0.3500000 0.9200000 0.6521739 0.7675805 1.8233078
```

The results based on the training data analysis and on the cross-validation procedure are represented in the following table. In this case the differences are rather small.

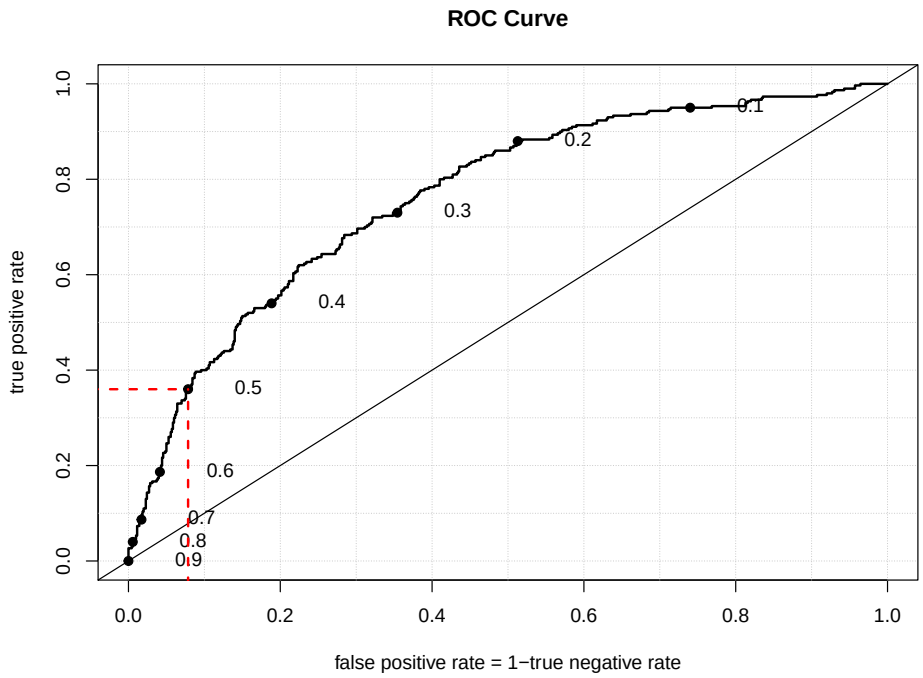
```
tabe <- rbind(de[c(1,3,2)],de.cv[c(1,3,2)])
rownames(tabe)<-c("training data","cross-validated")
colnames(tabe)<-c("tot. accuracy","true negative","true positive")
tabe

                tot. accuracy true negative true positive
training data           0.753     0.9214286          0.36
cross-validated         0.749     0.9200000          0.35
```

The threshold 0.5, for classifying an observation, is not necessarily the best choice, especially if the two classification errors have different costs. The ROC (Receiver Operating Characteristic) curve is a graphical plot that illustrates the performance of a binary classifier as its discrimination threshold varies. The curve is created by plotting the true positive rate (sensitivity) against the false positive rate (1−specificity) at various threshold settings. The area under the curve is viewed as a measure of classification accuracy, which can be used for comparing alternative classifiers. An area measure of 1 would indicate a perfect classifier, whereas a measure of 0.5 would indicate a

random classification.

The function `roc.plot` of the library `verification` generates the ROC curve representation, which may be useful for comparing the performances of alternative threshold settings. The main arguments are the binary observations coded as 0 and 1 (with the command `as.numeric(Scoring$y)-1` the factor `y` is transformed in a binary numeric vector) and the prediction probabilities. The true positive rate and the false positive rate associated to the threshold 0.5 are indicated using red segments, added to the plot with the function `segments`.

```
library(verification)
roc.plot(as.numeric(Scoring$y)-1, fitted(mod1),
    xlab='false positive rate = 1-true negative rate', ylab='true positive rate')
segments(1-de[3],-0.05,1-de[3],de[2],lty=2,col=2,lwd=2)
segments(-0.05,de[2],1-de[3],de[2],lty=2,col=2,lwd=2)
```



Since the ROC curves are useful for analyzing alternative classifiers, with the best classifier producing the highest curve, we will compare the classifier based on logistic regression, with those based on the LDA (Linear Discriminant Analysis), using only the numerical predictors, and on the kNN (k-Nearest Neighbors) method, with the optimal k value. The LDA procedure can be developed in R using the function `lda` of the library `MASS`. The main arguments are similar to those of function `glm` and the output is an object of class `lda`, which may be considered as argument of the `predict` function, in order to obtain the posterior probabilities of the two labels, for the considered data set.

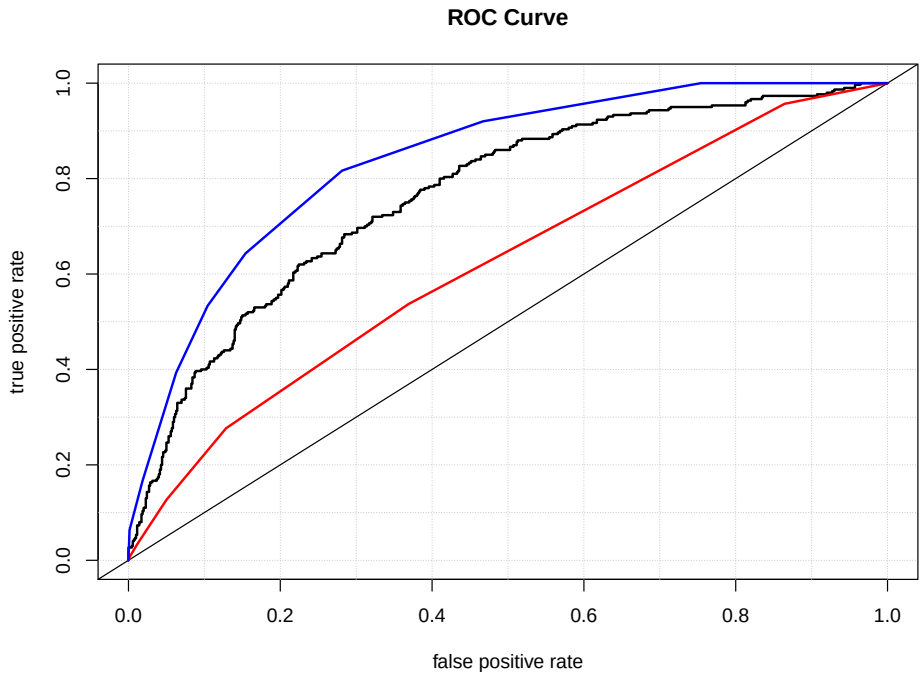
```
z <- lda(y ~ amount + duration, Scoring)
prob.lda <- predict(z)$posterior[,2]
```

Using the functions of the library `RWeka`, it is possible to apply the kNN method in order to obtain alternative estimates for the posterior probabilities. The library `RWeka` provides an R interface to Weka (www.cs.waikato.ac.nz/ml/weka/), which is an open source software collecting machine learning algorithms for data mining, written in Java and containing tools for data pre-processing, classification, regression, clustering, association rules, and visualization. In particular, we consider the function `IBk`, which provides a kNN classifier. The main arguments are similar to those of function `glm`, whereas the argument `control` gives the options to be passed to the Weka learner. In this case, RWeka automatically find the best value for k, between 1 and 20 (the optimal value, obtained using a leave-one-out cross-validation procedure, is $k = 9$). Finally, using function `predict`, we obtain the posterior probabilities of the two labels.

```
library(RWeka)
classifier <- IBk(y ~ moral + intuse+ account + amount + duration, data = Scoring,
                  control = Weka_control(K = 20, X= TRUE)) # selects K with CV n
prob.knn <- predict(classifier, type="probability")[,2]
```

The `verify` function, of the library `verification`, is used to compute verification statistics and skill scores for the three classification solutions. The arguments are the binary observations and the prediction probabilities, and the output is an object of class `verify`, which may be considered as the argument of function `roc.plot`. Then, the three classifiers are compared by considering the ROC curve representation, where the `roc.lines` function adds the LDA (in red) and the kNN (in blue) specific ROC curves to the plot with the curve related to the logistic model (in black). The kNN classification procedure presents the best classification accuracy.

```
ver_mod1<-verify(as.numeric(Scoring$y)-1, fitted(mod1), bins = FALSE,
            show = FALSE)
ver_lda<-verify(as.numeric(Scoring$y)-1, prob.lda, bins = FALSE, show = FALSE)
ver_knn<-verify(as.numeric(Scoring$y)-1, prob.knn, bins = FALSE, show = FALSE)
roc.plot(ver_mod1,xlab='false positive rate', ylab='true positive rate',lwd=2,
         show.thres = FALSE)
lines.roc(ver_lda, col = 2, lwd = 2)
lines.roc(ver_knn, col = 4, lwd = 2)
```



The function `evaluate_Weka_classifier` of the library `RWeka` can be used for computing model performance statistics of a fitted Weka classifier (defined as a `Weka_classifier` object). We obtain two different specifications of the confusion matrix: the first without cross-validation (`numFolds=0`) and the second using a cross-validation with 20 folds (`numFolds=20`).

```
tr.knn <- evaluate_Weka_classifier(classifier, numFolds=0)$confusionMatrix
cv.knn <- evaluate_Weka_classifier(classifier, numFolds=20,
              seed=1973)$confusionMatrix
```

The following function is defined in order to calculate the total accuracy, the true negative and the true positive rates associated to the matrices obtained using function `evaluate_Weka_classifier`. Finally, the results are organized in a table and then we may conclude that the assessments based on the training data are indeed over-optimistic with respect to those obtained with the cross-validation procedure.

```
ev.acc <- function(CM)
{
  acc <- (CM[1,1] + CM[2,2]) /sum(CM)
  tn <-  CM[1,1] / (CM[1,1]+CM[1,2])
   tp <-  CM[2,2] / (CM[2,1]+CM[2,2])
  return(c(acc,tn,tp))
}

tabe1 <- rbind(ev.acc(tr.knn), ev.acc(cv.knn))
rownames(tabe1)<-c("training data","cross-validated")
colnames(tabe1)<-c("tot. accuracy","true negative","true positive")
```

```
tabe1

                tot. accuracy true negative true positive
training data           0.787     0.8957143     0.5333333
cross-validated         0.749     0.8857143     0.4300000
```

4 Example: two-predictors simulated data

We consider the data set `mixture.example` of the package `ElemStatLearn`, associated to the book *The Elements of Statistical Learning, Data Mining, Inference, and Prediction* by T. Hastie, J.H. Friedman and R. Tibshirani. The data set concerns data from a simulated mixture example and it contains a list including:

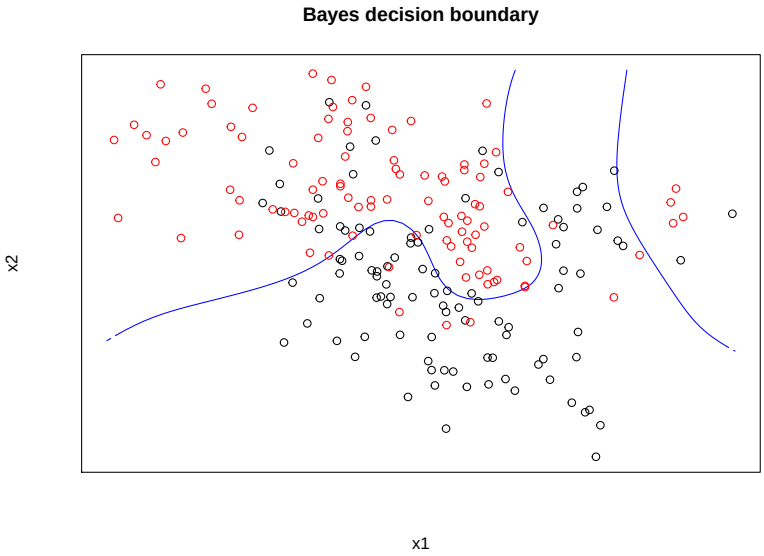
- `x`, a 200×2 matrix with values for the two predictors X_1 and X_2;
- `y`, a vector with the 200 observations of the binary response Y, taking values 0 or 1;
- `xnew`, a 6831×2 matrix of prediction points, on a 69×99 grid;
- `px1`, the grid values for first coordinate in `xnew`;
- `px2`, the grid values for second coordinate in `xnew`;
- `prob`, the vector with the 6831 true conditional probabilities of $Y = 1$ at each point in `xnew`.

These elements are saved in the objects `x`, `g`, `xnew`, `px1` and `px2`, respectively, and the true conditional probabilities are saved in the 69×99 matrix `prob.bayes`.

```
library(ElemStatLearn)
x <- mixture.example$x
g <- mixture.example$y
xnew <- mixture.example$xnew
px1 <- mixture.example$px1
px2 <- mixture.example$px2
prob.bayes <- matrix(mixture.example$prob, length(px1), length(px2))
```

Since the true model is known, the conditional probabilities are available and the Bayes classifier, which is a gold standard achieving the best classification, can be obtained. In the following plot we represent the Bayes decision boundary (in blue), which describes the points with a conditional probability of $Y = 1$ equal to 0.5. We use the function `contour`, which creates a contour plot. The first two arguments define the locations of the grid, the third contains the values to be plotted and the option `levels=0.5` gives the levels at which to draw contour lines. Indeed, with the function `points`, we add the points corresponding to the observed values of X_1 and X_2, in red if the value for Y is 1 and in black if the value for Y is 0. The `box()` function is finally used to draw a box around the plot.

```
contour(px1, px2, prob.bayes, levels=0.5, labels="", xlab="x1", axes=FALSE,
      ylab="x2", main="Bayes decision boundary", col="blue")
points(x, col=ifelse(g==1,"red", "black"))
box()
```



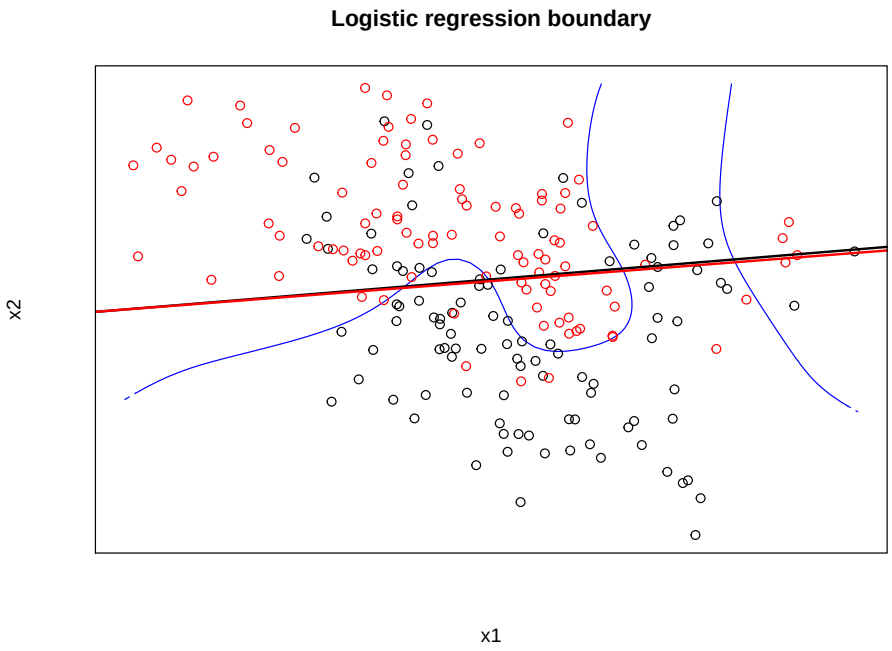
Using the observed values for the regressors X_1 and X_2 and for the binary response Y, we may fit a suitable logistic regression model, in order to estimate the conditional probability of $Y = 1$. The model is estimated using the `glm` function and the estimated regression coefficients are saved in the vector `beta`. In this case, the decision boundary, which describes the points with a conditional probability of $Y = 1$ equal to 0.5, corresponds to the line $\widehat{\beta}_0 + \widehat{\beta}_1 x_{i1} + \widehat{\beta}_2 x_{i2} = 0$. With the same data, we may fit a multiple linear regression model, which provides an unsuitable description for the conditional probability of the response and gives a decision boundary specified by the line $\widehat{\beta}_0 + \widehat{\beta}_1 x_{i1} + \widehat{\beta}_2 x_{i2} = 0.5$. In this case, the model is estimated using the `lm` function and the estimated regression coefficients are saved in the vector `beta.lm`.

```
mod <- glm(g~x, binomial)
beta <- coef(mod)
mod.lm <- lm(g~x)
beta.lm <- coef(mod.lm)
```

The `abline` commands are used to represent the logistic (in red) and linear (in black) classification boundaries. These two boundaries are very similar, and quite different from the Bayes decision boundary.

```
contour(px1, px2, prob.bayes, levels=0.5, labels="", xlab="x1", axes=FALSE,
       ylab="x2", main="Logistic regression boundary", col="blue", lwd=1)
points(x, col=ifelse(g==1,"red", "black"))
abline(a=-beta[1]/beta[3], b=-beta[2]/beta[3], lwd=2)
abline(a=-beta.lm[1]/beta.lm[3]+0.5/beta.lm[3],
```

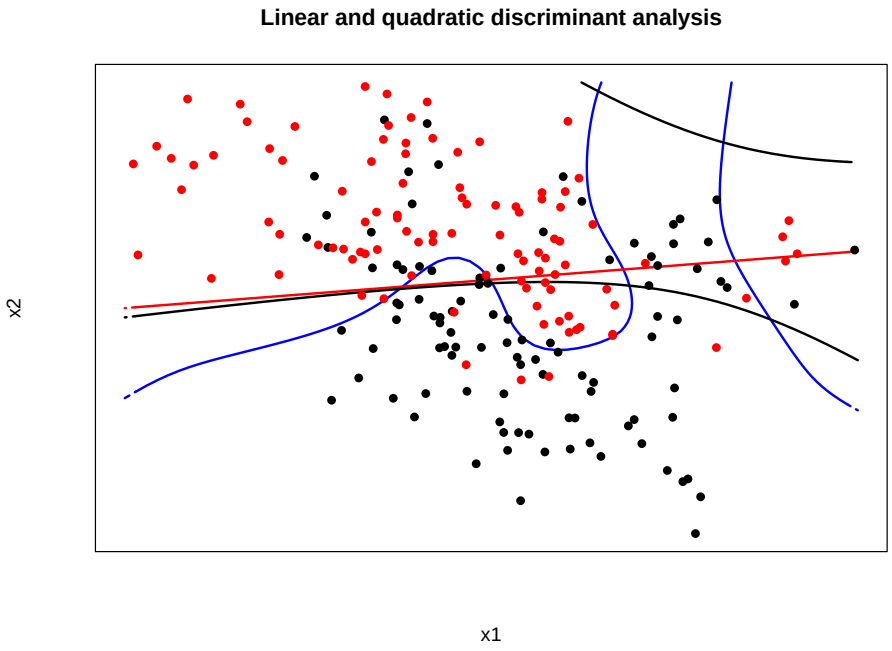
```
       b=-beta.lm[2]/beta.lm[3], col="red", lwd=2)
box()
```



The classification rules based on the linear discriminant analysis and on the quadratic discriminant analysis can be obtained using the functions `lda` and `qda` of the library `MASS`. The arguments of `qda` are similar to those of `lda`, presented in the previous section. The associated decision boundaries have, respectively, a linear (in red) and a quadratic (in black) form and, in this case, they are quite different from the Bayes decision boundary. Moreover, the LDA produces similar classifications to those obtained using a logistic regression model.

```
x1<-x[,1]
x2<-x[,2]
lda.fit<-lda(g~x1+x2)
lda.fit

Call:
lda(g ~ x1 + x2)

Prior probabilities of groups:
  0   1
0.5 0.5

Group means:
         x1        x2
0 1.1311358 0.2081589
```

```
1 0.4382081 1.3038738

Coefficients of linear discriminants:
           LD1
x1 -0.09935417
x2  1.09553973

# plot(lda.fit)
lda.pred<-predict(lda.fit,newdata=data.frame(x1=xnew[,1],x2=xnew[,2]))
problda<-matrix(1-lda.pred$posterior,length(px1),length(px2))
# 1-posterior probability, since the class labels are switched

qda.fit<-qda(g~x1+x2)
qda.fit

Call:
qda(g ~ x1 + x2)

Prior probabilities of groups:
  0   1
0.5 0.5

Group means:
         x1        x2
0 1.1311358 0.2081589
1 0.4382081 1.3038738

qda.pred<-predict(qda.fit,newdata=data.frame(x1=xnew[,1],x2=xnew[,2]))
probqda<-matrix(1-qda.pred$posterior,length(px1),length(px2))
# 1-posterior probability, since the class labels are switched

contour(px1, px2, prob.bayes, levels=0.5, labels="", xlab="x1", axes=FALSE,
        ylab="x2", main="Linear and quadratic discriminant analysis",
        col="blue",lwd=2)
contour(px1, px2, problda, levels=0.5,labels="", xlab="", axes=FALSE,
        ylab="", main="", co="red", lwd=2,add=T)
contour(px1, px2, probqda, levels=0.5,labels="", xlab="", axes=FALSE,
        ylab="", main="", lwd=2,add=T)
points(x, col=ifelse(g==1,"red", "black"),pch=16)
box()
```



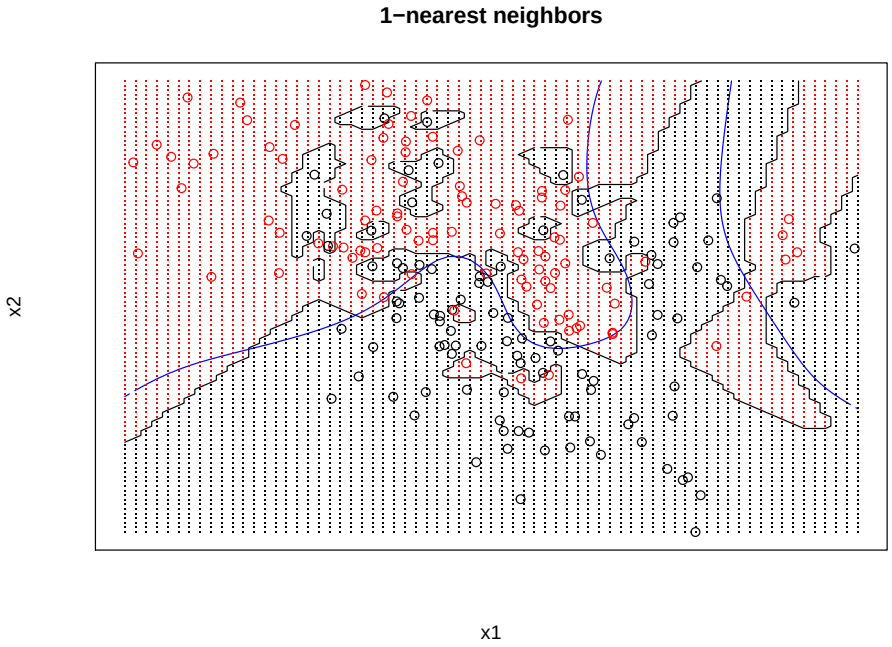
A better classification can be obtained by considering the kNN procedure. We use the function `knn` of the library `class`, which provides the k-Nearest Neighbor classification for a test set from a given training set. The main arguments are a matrix or a data frame of training set cases, a matrix or a data frame of test set cases, a factor of true classifications for the training set and the number `k` of neighbors to be considered; indeed, the option `prob=TRUE` assures that the proportion of the votes for the winning class are returned as the attribute `prob`. In the first analysis we set `k=1`, so that the classification rule is based only on the status of the single nearest "observation". For this reason, the object `prob` includes only 0 or 1 values, which are saved in the matrix `prob1`.

```
library(class)
mod1 <- knn(x, xnew, g, k=1, prob=TRUE)
prob <- attr(mod1, "prob")
prob <- ifelse(mod1=="1", prob, 1-prob)
prob1 <- matrix(prob, length(px1), length(px2))
```

The `contour` function is used to represent the classification boundary and the `expand.grid` function creates a data frame from all combinations of the values of the supplied vectors `px1` and `px2`. The plot includes also the Bayes decision boundary (in blue), the red points corresponding to $Y = 1$ and the black points corresponding to $Y = 0$. The classification is also performed for the points of the expanded grid of values. The classification appears even too detailed, with respect to that one given by the Bayes decision boundary.

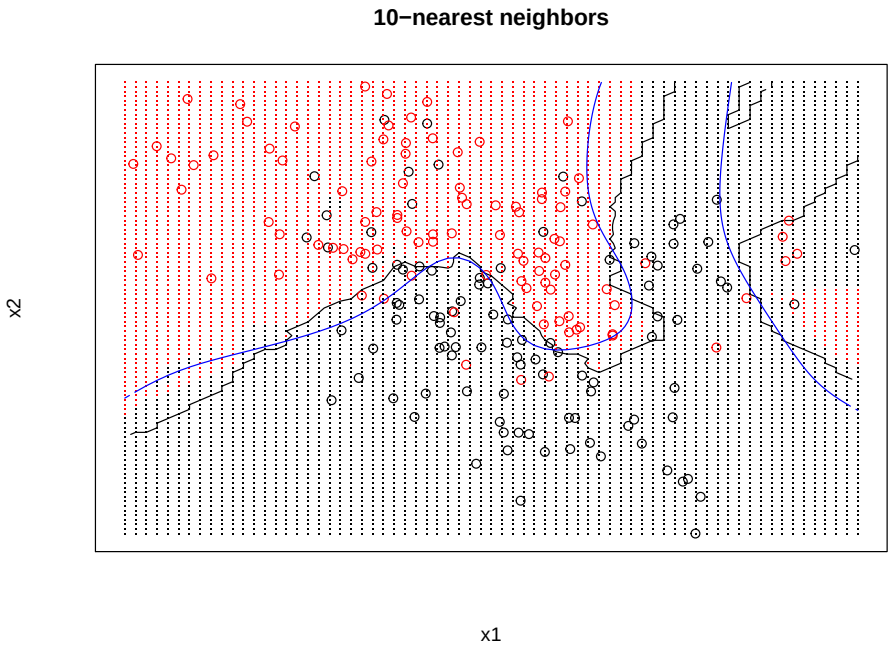
```
contour(px1, px2, prob1, levels=0.5, labels="", axes=FALSE,
        main="1-nearest neighbors", xlab="x1", ylab="x2")
gd <- expand.grid(x=px1, y=px2)
contour(px1, px2, prob.bayes, levels=0.5, labels="", xlab="x1",
```

```
    ylab="x2", col="blue", lwd=1,add=T)
points(x, col=ifelse(g==1, "red", "black"))
points(gd, pch=".", cex=1.2, col=ifelse(prob1>0.5, "red", "black"))
box()
```



A similar analysis is developed by considering `k=10` and then the estimated probability is obtained as the average status of the 10 nearest "observed" points. The obtained classification boundary is smoother and closer to the Bayesian one.

```
mod10 <- knn(x, xnew, g, k=10, prob=TRUE)
prob <- attr(mod10, "prob")
prob <- ifelse(mod10=="1", prob, 1-prob)
prob1 <- matrix(prob, length(px1), length(px2))
contour(px1, px2, prob1, levels=0.5, labels="", axes=FALSE,
        main="10-nearest neighbors", xlab="x1", ylab="x2")
points(x, col=ifelse(g==1, "red", "black"))
gd <- expand.grid(x=px1, y=px2)
points(gd, pch=".", cex=1.2, col=ifelse(prob1>0.5, "red", "black"))
contour(px1, px2, prob.bayes, levels=0.5, labels="", xlab="x1", ylab="x2",
        col="blue", lwd=1,add=T)
box()
```



Finally, we consider a classification procedure with the option `k=50`. It is immediate to see that the higher is the number of points included in the neighborhood, the smoother is the classification boundary. Then, we obtain a rough classification, since the set of neighbors of each point is enlarged too much. However, as emphasized in the following plot, the classifier is, even now, closer to the Bayes classifier (in blue) than the linear boundary obtained using a logistic regression model (in red).

```
mod50 <- knn(x, xnew, g, k=50, prob=TRUE)
prob <- attr(mod50, "prob")
prob <- ifelse(mod50=="1", prob, 1-prob)
prob1 <- matrix(prob, length(px1), length(px2))
contour(px1, px2, prob1, levels=0.5, labels="", axes=FALSE,
        main="50-nearest neighbors",xlab="x1", ylab="x2")
points(x, col=ifelse(g==1, "red", "black"))
gd <- expand.grid(x=px1, y=px2)
points(gd, pch=".", cex=1.2, col=ifelse(prob1>0.5, "red", "black"))
contour(px1, px2, prob.bayes, levels=0.5, labels="", xlab="x1", ylab="x2",
        col="blue", lwd=1,add=T)
abline(a=-beta[1]/beta[3], b=-beta[2]/beta[3], col=2, lwd=2)
box()
```


50−nearest neighbors
x2
x1