DATA SCIENCETECH INSTITUTE

EXAM

# Advanced Statistical Analysis and Machine Learning

*Done by:* Mohamed Abid

# Exercise C

Let consider the files data1 and data2.
For data1, we want to explain the development while in data2, we want to explain the Grade.
By considering the nature of the variables, create several models and give the best one.

## C.1. DATA 1

### C.1.1. Loading of libraries

```
library(ggplot2)
library(ggpubr)
library(Matrix)
library(glmnet)
```

### C.1.2. Importation of the data

```
set.seed(3333)
D1 = read.csv('data1.csv', header=TRUE, sep=';')
dim(D1)
```

```
## [1] 12  4
```

```
head(D1)
```

```
##   Children     Treatment Psychologist Development
## 1        1       placebo            2        36.8
## 2        2 produit actif            1        93.7
## 3        3       placebo            1        56.6
## 4        4       placebo            1        37.7
## 5        5 produit actif            2        70.3
## 6        6       placebo            1        24.2
```

```
str(D1)
```

```
## 'data.frame':    12 obs. of  4 variables:
##  $ Children    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Treatment   : chr  "placebo" "produit actif" "placebo" "placebo" ...
##  $ Psychologist: int  2 1 1 1 2 1 2 2 1 2 ...
##  $ Development : num  36.8 93.7 56.6 37.7 70.3 24.2 42.5 50.9 71.8 72.5 ...
```

We have at our disposal a small dataset containing two qualitative explanatory variables associated with a quantitative response variable.

As a first step, we convert the type of the qualitative variables (“Treatment” and “Psychologist”) into factors.

```
D1$Psychologist = as.factor(D1$Psychologist)
D1$Psychologist
```

```
##  [1] 2 1 1 1 2 1 2 2 1 2 1 2
## Levels: 1 2
```

```
D1$Treatment = as.factor(D1$Treatment)
D1$Treatment
```

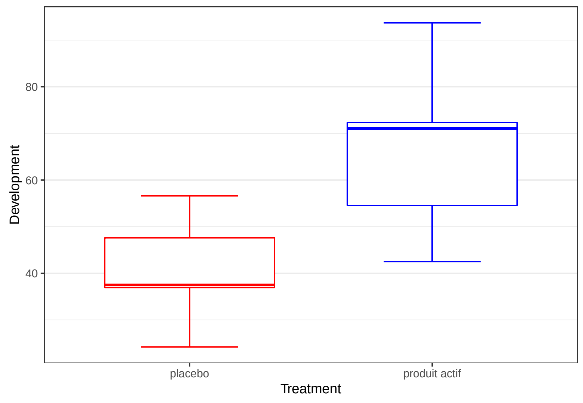
```
##  [1] placebo       produit actif placebo       placebo       produit actif
##  [6] placebo       produit actif placebo       produit actif produit actif
## [11] produit actif placebo
## Levels: placebo produit actif
```

Elimination of the column “Children”.

```
D1$Children = NULL
```

### C.1.4. Visualization of the data

```
box.plot.Treatment = ggboxplot(D1, x="Treatment", y="Development"
                               , bxp.errorbar = TRUE
                               , palette = c('red', 'blue')
                               , color = "Treatment")
box.plot.Treatment + theme_bw() + theme(legend.position='None') + rremove("x.grid")
```

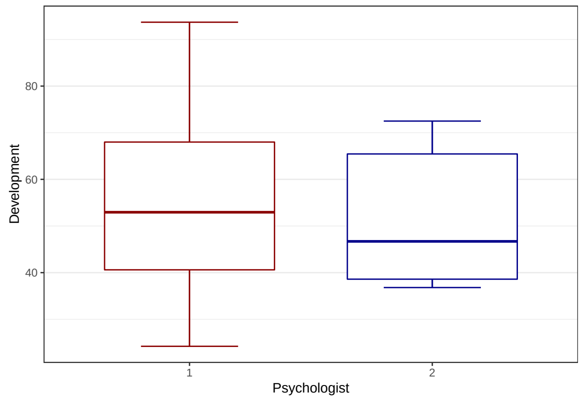


The above figure represents the boxplots of the explanatory variable “Treatment” with respect to the response variable “Development”.

We see that our boxplots for the modality “placebo” and “produit actif” are different: the median of the modality “produit actif” is bigger than the median of the modality “placebo”, and the observations of the modality “produit actif” are distributed at a higher level than the observations of the modality “placebo”.

⇒ We can anticipate that there will be some influence of the variable “Treatment” onto the response variable “Development”.

```
box.plot.Psychologist = ggboxplot(D1, x='Psychologist', y='Development'
                                  , bxp.errorbar = TRUE
                                  , palette = c('darkred', 'darkblue')
                                  , color = 'Psychologist')
box.plot.Psychologist + theme_bw() + theme(legend.position='None') + rremove("x.grid")
```



The above figure represents the boxplots of the explanatory variable “Psychologist” with respect to the response variable “Development”.

We can note that the two boxplots are quiet at the same level and present the same dispersion.

⇒ We have the impression that there will be no influence of the factor “Psychologist” onto the response variable “Development”.

### C.1.5. Assumption

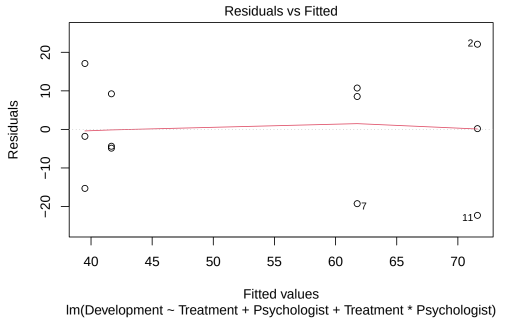
The goal in this step is to check the gaussianity of the noise.

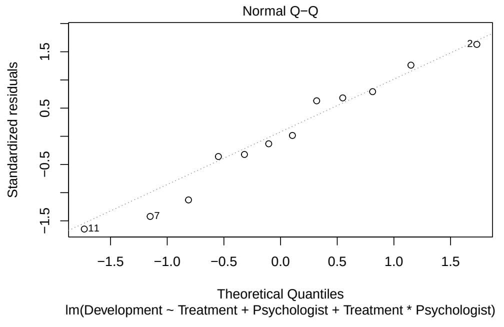
```
L1.c = lm(Development ~ Treatment + Psychologist + Treatment * Psychologist
          , data=D1)

summary(L1.c)
```

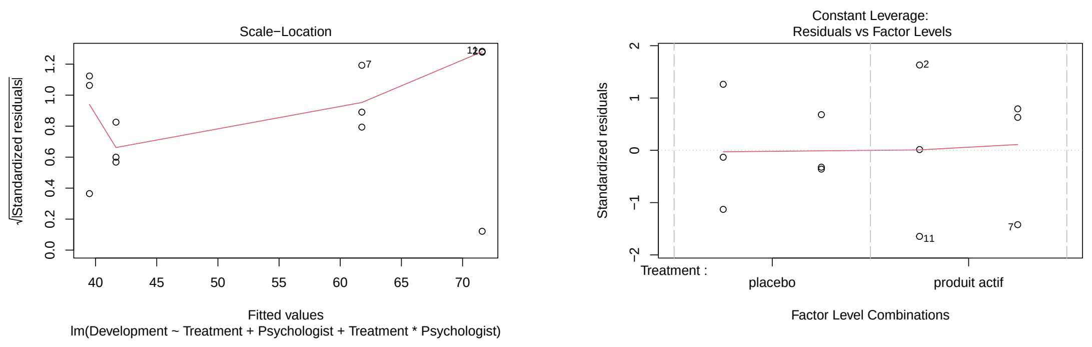
```
##
## Call:
## lm(formula = Development ~ Treatment + Psychologist + Treatment *
##     Psychologist, data = D1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.300  -7.475  -0.800   9.608  22.100
##
## Coefficients:
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                             39.500      9.580   4.123  0.00333 **
## Treatmentproduit actif                  32.100     13.548   2.369  0.04530 *
## Psychologist2                            2.167     13.548   0.160  0.87691
## Treatmentproduit actif:Psychologist2   -12.000     19.160  -0.626  0.54857
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.59 on 8 degrees of freedom
## Multiple R-squared:  0.4992, Adjusted R-squared:  0.3114
## F-statistic: 2.658 on 3 and 8 DF,  p-value: 0.1196
```

- From the “summary(L1.c)” we note that there is some symmetry of the “Residuals” with respect to the median and the median is close to 0.

```
plot(L1.c)
```







- In the "Residuals vs Fitted" plot presents some symmetry. Thus, we can consider that the noise is centered. The homoscedasticity can be retained.

- In the "Normal Q-Q" plot we can see that the cloud of points is distributed along the line. We can accept the gaussianity of the noise.

For further checking, we perform Kolmogorov test:

```
sres1 = rstandard(L1.c)
ks.test(sres1, 'pnorm')
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sres1
## D = 0.15227, p-value = 0.9048
## alternative hypothesis: two-sided
```

The p-value of the Kolmogorov-Smirnov test is equal to $0.9048 > 0.05 \Rightarrow$ we accept the gaussianity of the noise.

With the gaussianity of the noise being checked, we can now investigate all the elements provided by the "lm" function.

From the "summary(L1.c)" we can see that the p-value of the model is equal to $0.1196 > 0.05$.

$\Rightarrow$ We accept $H_0$ in this test (all the coefficients are null).

$\Rightarrow$ We cannot consider a linear model between all the explanatory varaibles and the response variable.

By looking at the column "Pr(> | t |)", some p-values are bigger than 0.05 while others are smaller than 0.05.

$\Rightarrow$ we have to perform variable selection to improve our model.

### C.1.6. ANOVA model

We begin by performing ANOVA for the complete model: additive model with cross effect

```
anova(L1.c)
```

```
## Analysis of Variance Table
##
## Response: Development
##                         Df  Sum Sq Mean Sq F value  Pr(>F)
## Treatment                1 2043.63 2043.63  7.4222 0.02607 *
## Psychologist             1   44.08   44.08  0.1601 0.69953
## Treatment:Psychologist  1  108.00  108.00  0.3922 0.54857
## Residuals               8 2202.71  275.34
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The application of the ANOVA procedure shows a p-value for the cross effect (Treatment:Psychologist) equal to $0.54857 > 0.05 \Rightarrow$ the cross effect has no influence on the response variable.

In this case, we construct the additive model:

```
L1.a = lm(Development ~ Treatment + Psychologist, data=D1)
anova(L1.a)
```

```
## Analysis of Variance Table
##
## Response: Development
##              Df  Sum Sq Mean Sq F value Pr(>F)
## Treatment     1 2043.63 2043.63  7.9597 0.0200 *
## Psychologist  1   44.08   44.08  0.1717 0.6883
## Residuals     9 2310.71  256.75
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- For the row "Treatment" we have p-value $= 0.0200 < 0.05$

$\Rightarrow$ "Treatment" has an influence on the response variable "Development".

- For the row "Psychologist" we have p-value $= 0.6883 > 0.05$

$\Rightarrow$ "Psychologist" has no influence on the response variable "Development".

We have to consider a model with only the factor "Treatment" to take a conclusion.

```
L1.Treatment = lm(Development ~ Treatment, data=D1)
anova(L1.Treatment)
```

```
## Analysis of Variance Table
##
## Response: Development
##           Df Sum Sq Mean Sq F value  Pr(>F)
## Treatment  1 2043.6 2043.63  8.6786 0.01464 *
## Residuals 10 2354.8  235.48
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(L1.Treatment)
```

```
##
## Call:
## lm(formula = Development ~ Treatment, data = D1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24.1833  -6.9333   0.3667   6.9417  27.0167
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 40.583      6.265   6.478 7.09e-05 ***
## Treatmentproduit actif   26.100      8.860   2.946   0.0146 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.35 on 10 degrees of freedom
## Multiple R-squared:  0.4646, Adjusted R-squared:  0.4111
## F-statistic: 8.679 on 1 and 10 DF,  p-value: 0.01464
```

We can conclude that the best model with ANOVA method is: Development $\sim$ Treatment.

### C.1.7. Step by step strategies: backward strategy

- We start by the complete model and suppress the variable which has the biggest p-value.

- We repeat the first step and stop once the biggest p-value becomes small (p-value< $\alpha$).

```
# The linear model with all the explanatory variables and the cross effect
L1.c
```

```
##
## Call:
## lm(formula = Development ~ Treatment + Psychologist + Treatment *
##     Psychologist, data = D1)
##
## Coefficients:
##                          (Intercept)                Treatmentproduit actif
##                               39.500                                32.100
##                        Psychologist2  Treatmentproduit actif:Psychologist2
##                                2.167                               -12.000
```

- We apply the backward strategy with AIC criterion

```
L1.B.AIC = step(L1.c, direction='backward')
```

```
## Start:  AIC=70.55
## Development ~ Treatment + Psychologist + Treatment * Psychologist
##
##                          Df Sum of Sq    RSS    AIC
## - Treatment:Psychologist  1       108 2310.7 69.125
## <none>                                2202.7 70.550
##
## Step:  AIC=69.12
## Development ~ Treatment + Psychologist
##
##                Df Sum of Sq    RSS    AIC
## - Psychologist  1     44.08 2354.8 67.352
## <none>                      2310.7 69.125
## - Treatment     1   2043.63 4354.3 74.728
##
## Step:  AIC=67.35
## Development ~ Treatment
##
##             Df Sum of Sq    RSS    AIC
## <none>                   2354.8 67.352
## - Treatment  1    2043.6 4398.4 72.849
```

```
L1.B.AIC
```

```
##
## Call:
## lm(formula = Development ~ Treatment, data = D1)
##
## Coefficients:
##           (Intercept)  Treatmentproduit actif
##                 40.58                   26.10
```

- We apply the backward strategy with Fisher criterion

```
L1.B.Fisher = step(L1.c, direction='backward', test='F')
```

```
## Start:  AIC=70.55
## Development ~ Treatment + Psychologist + Treatment * Psychologist
##
##                          Df Sum of Sq    RSS    AIC F value Pr(>F)
## - Treatment:Psychologist  1       108 2310.7 69.125  0.3922 0.5486
## <none>                                2202.7 70.550
##
## Step:  AIC=69.12
## Development ~ Treatment + Psychologist
##
##                Df Sum of Sq    RSS    AIC F value Pr(>F)
## - Psychologist  1     44.08 2354.8 67.352  0.1717 0.6883
## <none>                      2310.7 69.125
## - Treatment     1   2043.63 4354.3 74.728  7.9597 0.0200 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=67.35
## Development ~ Treatment
##
##             Df Sum of Sq    RSS    AIC F value  Pr(>F)
## <none>                   2354.8 67.352
## - Treatment  1    2043.6 4398.4 72.849  8.6786 0.01464 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
L1.B.Fisher
```

```
##
## Call:
## lm(formula = Development ~ Treatment, data = D1)
##
## Coefficients:
##           (Intercept)  Treatmentproduit actif
##                 40.58                   26.10
```

We can conclude that the model resulting from the backward strategy is: Development ∼ Treatment.

### C.1.8. Step by step strategies: forward strategy

We create a linear model with just the intercept.

```
L1.1 = lm(D1$Development ~ 1, data=D1)
```

- We start by comparing all the models of dimension 1 and we keep the best variable associated to the smallest p-value.

- We repeat the first step and stop when the smallest p-value becomes too big (p-value> $\alpha$). We stop at the step before.

- We apply the forward strategy with AIC criterion

```
L1.F.AIC = step(L1.1, scope=list(L1.1, L1.c), direction='forward')
```

```
## Start:  AIC=72.85
## D1$Development ~ 1
```

```
L1.F.AIC
```

```
##
## Call:
## lm(formula = D1$Development ~ 1, data = D1)
##
## Coefficients:
## (Intercept)
##       53.63
```

- We apply the forward strategy with Fisher criterion

```
L1.F.Fisher = step(L1.1, scope=list(L1.1, L1.c), direction='forward', test='F')
```

```
## Start:  AIC=72.85
## D1$Development ~ 1
```

```
L1.F.Fisher
```

```
##
## Call:
## lm(formula = D1$Development ~ 1, data = D1)
##
## Coefficients:
## (Intercept)
##       53.63
```

The model provided by the forward strategy is Development $\sim$ (Intercept) and thus will not be retained.

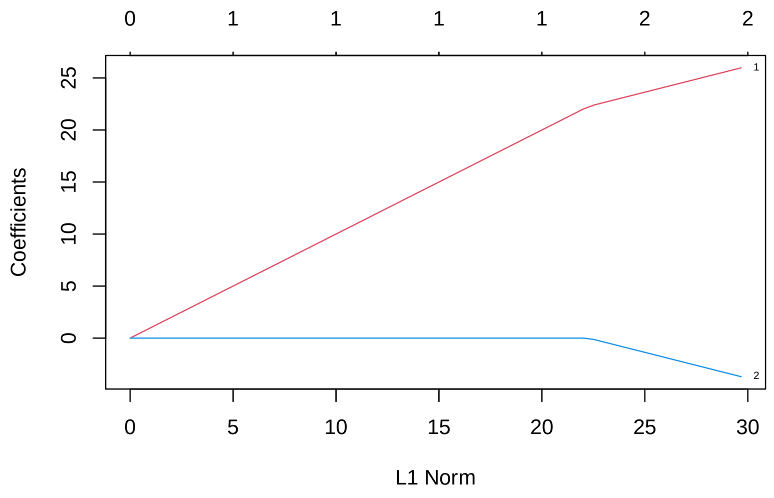
### C.1.9. Lasso method

We are going to perform the Lasso method, the solution of which solves a penalized least squared criterion $\Rightarrow$ we can write: $\widehat{\beta}_{Lasso,\lambda} = \min_{\beta} ||Y - X\beta||^2 + \lambda \sum_i |\beta_i|$

The penalty cost in the "glmnet" function is $\frac{1-\alpha}{2}||\beta||_2^2 + \alpha||\beta||_1$ with $\alpha \in [0,1]$ where $||\beta||_2^2$ is the penalty term for Ridge and $||\beta||_1$ is the penalty term for Lasso.

$\Rightarrow$ We take $\alpha = 1$ to perform Lasso.

```
X1 = data.matrix(D1[, -3])
Y1 = D1$Development
D1.Lasso = glmnet(X1, Y1, alpha=1)
plot(D1.Lasso, col=c(2, 4), label=TRUE)
```



This graph gives the trajectories of the $\beta_i$ obtained for different values of $\lambda$ with respect to the $\mathcal{L}_1$ norm of the $\beta$ vector.
When we decrease $\lambda$ (increase the $\mathcal{L}_1$ norm), some coefficients become different from 0 but not all at the same time.
$\Rightarrow$ We obtain many models, one for each value of $\lambda$ and we select the best value of $\lambda$.

To perform this, we apply 'cv.glmnet" function to compute the cross validation error for each model. Then we keep the best model presenting the lowest error.
Since we have a small dataset, we will take the number of the splitting in K-folds "nfolds = 3" (by default nfolds = 10).

```
D1.Lasso.cv = cv.glmnet(X1, Y1, alpha=1, nfolds=3)
D1.Lasso.cv
```

```
##
## Call:  cv.glmnet(x = X1, y = Y1, nfolds = 3, alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min  3.233    16   334.2  74.77       1
## 1se 13.050     1   382.2 174.03       0
```

In order to delete the instability due to the search of a minimum, we look for the "1se" row and not the "min".

```
D1.best.Lambd = D1.Lasso.cv$lambda.1se
D1.best.Lambd
```

```
## [1] 13.05
```

We plug the $\lambda$ of the "1se" row into the "glmnet" function to obtain the best model for the Lasso.

```
D1.best.Lasso = glmnet(X1, Y1, alpha=1, lambda=D1.best.Lambd)
coefficients(D1.best.Lasso)
```

```
## 3 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  5.363333e+01
## Treatment    1.275321e-14
## Psychologist .
```

We can conclude that the best model with Lasso method is: Development ∼ Treatment.

Since the Lasso is a biased model we will only consider the information about the selected variables and not the estimated coefficients it provides.

To obtain an unbiased model we apply classical ordinary squared to the selected variables by the Lasso.

### C.1.10. Conclusion

All the considered approaches: ANOVA method, Step by Step method and Lasso method suggest the same model that is Development ∼ Treatment.

Applying the “lm” function, we obtain the below parameters for this model:

```
L1.best = lm(Development ~ Treatment, data=D1)
L1.best
```

```
##
## Call:
## lm(formula = Development ~ Treatment, data = D1)
##
## Coefficients:
##           (Intercept)  Treatmentproduit actif
##                 40.58                   26.10
```

```
summary(L1.best)
```

```
##
## Call:
## lm(formula = Development ~ Treatment, data = D1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24.1833  -6.9333   0.3667   6.9417  27.0167
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)              40.583      6.265   6.478 7.09e-05 ***
## Treatmentproduit actif   26.100      8.860   2.946   0.0146 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.35 on 10 degrees of freedom
## Multiple R-squared:  0.4646, Adjusted R-squared:  0.4111
## F-statistic: 8.679 on 1 and 10 DF,  p-value: 0.01464
```

## C.2. DATA 2

### C.2.1. Loading of libraries

```
library(ggplot2)
library(ggpubr)
library(MASS)
library(leaps)
library(Matrix)
library(glmnet)
library(dplyr)
library(VSURF)
```

### C.2.2. Importation of the data

```
set.seed(3333)
D2 = read.csv('data2.csv', header=TRUE, sep=';')
dim(D2)
```

```
## [1] 16  6
```

```
head(D2)
```

```
##   Product Sugar Acid Bitter Pulpy Grade
## 1       1  6.21 7.08   2.00  2.54  4.97
## 2       2  7.75 3.29   1.54  2.26  6.98
## 3       3  7.21 4.38   1.79  2.58  4.58
## 4       4  8.33 2.79   1.63  2.71  6.45
## 5       5  4.87 7.71   1.96  1.70  4.33
## 6       6  5.09 7.50   2.13  2.42  4.26
```

```
str(D2)
```

```
## 'data.frame':    16 obs. of  6 variables:
##  $ Product: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Sugar  : num  6.21 7.75 7.21 8.33 4.87 5.09 6.04 6.09 6.08 6.17 ...
##  $ Acid   : num  7.08 3.29 4.38 2.79 7.71 7.5 6.58 5.13 5.5 5.58 ...
##  $ Bitter : num  2 1.54 1.79 1.63 1.96 2.13 2.04 2 2.09 2.13 ...
##  $ Pulpy  : num  2.54 2.26 2.58 2.71 1.7 2.42 2.04 2.42 2.46 2.48 ...
##  $ Grade  : num  4.97 6.98 4.58 6.45 4.33 4.26 6.16 6.26 5.83 5.74 ...
```

We have at our disposal a small dataset containing four quantitative explanatory variables associated with a quantitative response variable.

As we have a small dataset, a false correlation phenomenon may occur.

### C.2.3. Preparation of the data

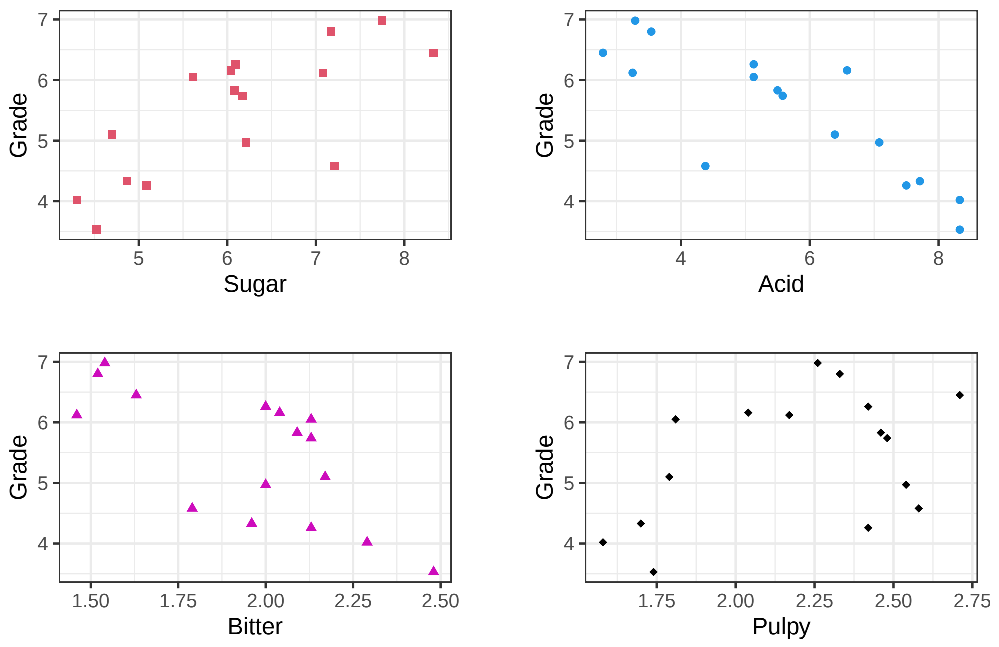
- Elimination of the column "Product"

```
D2$Product = NULL
```

### C.2.4. Visualization of the data

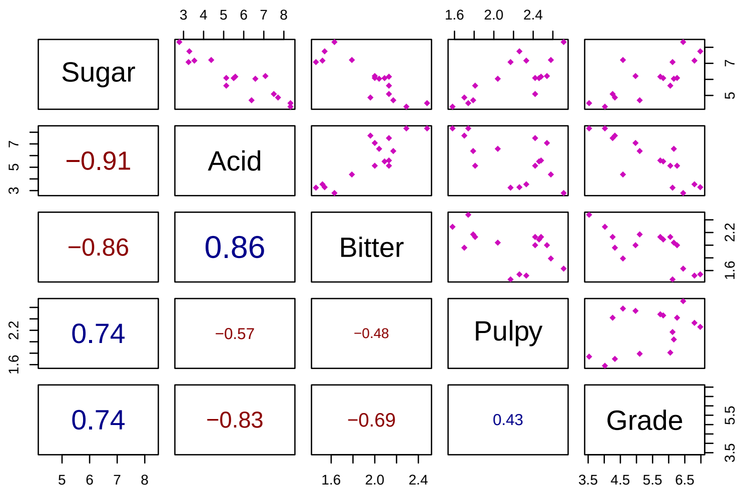
- Scatter plot:

```
scatter.color = c(2,4,6,1)
for (i in 1:4) {
  print(ggscatter(D2, x=names(D2)[i] , y='Grade'
                  , col=scatter.color[i], shape=14+i, size=1.5) + theme_bw())
}
```



- pairs plot:

```
# Correlation panel
panel.cor <- function (x, y) {
  usr = par('usr'); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r = round(cor(x, y), digits=2)
  txt = paste0('', r)
  cex.cor = 0.6 / strwidth(txt)
  text(0.5, 0.5, txt, cex=abs(cex.cor*r), col=c('darkred', 'darkblue')[floor(r)+2])
}
# Customize upper panel
panel.plot <- function (x, y) {
  points(x, y, pch=18, col=6)
}
# Create the plots
pairs(D2, lower.panel=panel.cor, upper.panel=panel.plot)
```



We can see that there is high correlation between some explanatory variables.
⇒ We need probably to perform variable selection.

### C.2.5. Assumption

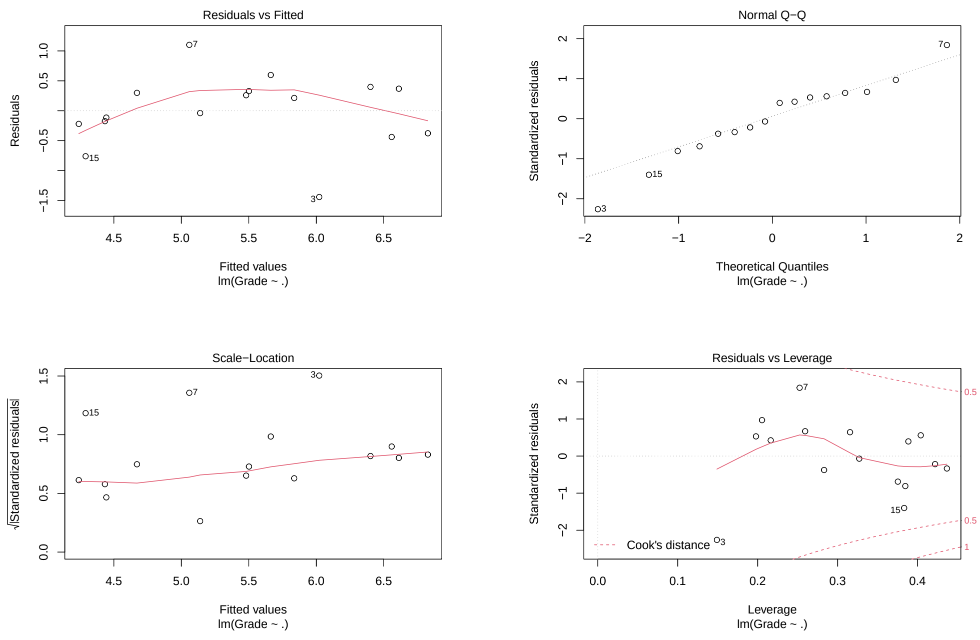
The goal in this step is to check the gaussianity of the noise.

```
L2.c = lm(Grade ~ ., data=D2)
summary(L2.c)
```

```
##
## Call:
## lm(formula = Grade ~ ., data = D2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.44199 -0.25923  0.08703  0.33864  1.10096
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.5091     4.8970   1.533   0.1534
## Sugar         0.1006     0.5589   0.180   0.8605
## Acid         -0.4973     0.2624  -1.895   0.0846 .
## Bitter        0.3869     1.3644   0.284   0.7820
## Pulpy        -0.2816     0.8537  -0.330   0.7477
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6914 on 11 degrees of freedom
## Multiple R-squared:  0.6883, Adjusted R-squared:  0.575
## F-statistic: 6.073 on 4 and 11 DF,  p-value: 0.00786
```

- From the "summary(L2.c)" we note that there is some symmetry of the "Residuals" with respect to the median and the median is close to 0.

```
plot(L2.c)
```



- In the “Normal Q-Q” plot, we note that the main part of the points are quiet close to the line, which suggests the gaussianity of the noise.

- However, in the “Residuals vs Fitted” plot, the distribution of the residuals does not conform perfectly with a gaussian behaviour.

For further checking, we perform Kolmogorov test:

```
# Kolmogorov-Smirnov test
sres = rstandard(L2.c)
ks.test(sres, 'pnorm')
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sres
## D = 0.15354, p-value = 0.7919
## alternative hypothesis: two-sided
```

The Kolmogorov-Smirnov test returns a p-value of $0.7919$. We will try to find a transformation of the response variable to improve the gaussianity test.

Possible transformations include applying the square root, the reciprocal or the logarithmic functions to the data.

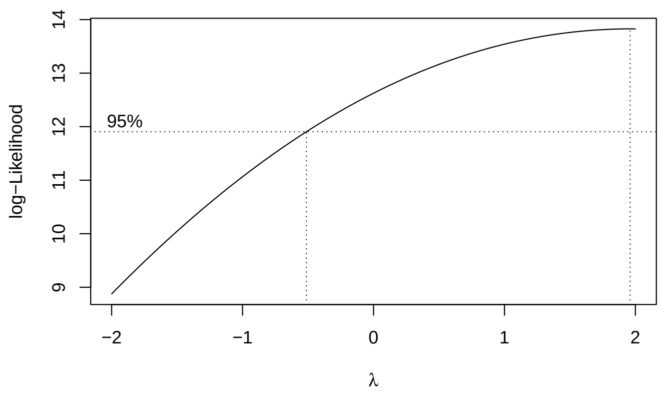
To find the best transformation we are going to use the ”Box-Cox” transformation which is defined by:

$$y(\lambda) = \begin{cases} \dfrac{y^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(y) & \text{if } \lambda = 0 \end{cases}$$

where $y$ is a list of $n$ strictly positive numbers and $\lambda$ is the transformation constant.

We are going to perform a ”Box-Cox” transformation in R by using the ”boxcox” function from the ”MASS” package. By using the ”boxcox” function we can compute the best $\lambda$ associated with the best transformation of the response variable.

```
# Application of the function "boxcox":
bc = boxcox(Grade ~ ., data=D2)
```



```
# Saving of the best lambda:
Lambda = bc$x[which.max(bc$y)]
Lambda
```

```
## [1] 1.959596
```

$\Rightarrow$ We find $\lambda = 1.959 \neq 0 \Rightarrow$ we apply the transformation $y = \dfrac{y^\lambda - 1}{\lambda}$

```
# Transformation of the response variable:
D2.t = D2[,1:4]
D2.t$Grade.t = (D2$Grad^Lambda - 1) / Lambda
# Creation of the model
new.L2.c = lm(Grade.t ~ ., data=D2.t)
summary(new.L2.c)
```

```
##
## Call:
## lm(formula = Grade.t ~ ., data = D2.t)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.3523 -1.2286  0.1371  1.7154  5.0220
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  22.1352    24.0297   0.921   0.3767
## Sugar         0.9939     2.7428   0.362   0.7239
## Acid         -2.3154     1.2877  -1.798   0.0996 .
## Bitter        2.1414     6.6952   0.320   0.7551
## Pulpy        -2.3584     4.1893  -0.563   0.5848
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.393 on 11 degrees of freedom
## Multiple R-squared:  0.695,  Adjusted R-squared:  0.5841
## F-statistic: 6.267 on 4 and 11 DF,  p-value: 0.007028
```

We perform the Kolmogorov-Smirnov test:

```
# Kolmogorov-Smirnov test
sres2 = rstandard(new.L2.c)
ks.test(sres2, 'pnorm')
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sres2
## D = 0.11213, p-value = 0.9742
## alternative hypothesis: two-sided
```

The p-value of the Kolmogorov-Smirnov test is equal to $0.9742 > 0.05 \Rightarrow$ we accept the gaussianity of the noise.

With the gaussianity of the noise being checked, we can now investigate all the elements provided by the “lm” function.

From the “summary(new.L2.c)” we can see that the p-value of the model is equal to $0.007028 < 0.05$.

$\Rightarrow$ We accept $H_1$ in this test (at least one of the coefficients is not null).

⇒ We can consider a linear model between all the explanatory variables and the response variable.

By looking at the column "Pr(> | t |)", we can see that all the p-values are big but as we have a high correlation between some explanatory variables, we need to perform variable selection to improve our model.

### C.2.6. Adjusted R-squared

```
L2.AR2 = leaps(D2.t[,-5], D2.t$Grade.t, method='adjr2', nbest=1)
L2.AR2
```

```
## $which
##       1    2     3     4
## 1 FALSE TRUE FALSE FALSE
## 2 FALSE TRUE FALSE  TRUE
## 3  TRUE TRUE FALSE  TRUE
## 4  TRUE TRUE  TRUE  TRUE
##
## $label
## [1] "(Intercept)" "1"           "2"           "3"           "4"
##
## $size
## [1] 2 3 4 5
##
## $adjr2
## [1] 0.6630176 0.6430580 0.6152232 0.5841111
```

We can see that the biggest value of the "adjr2" is obtained by the first model.

```
id.max.adjr2 = which(L2.AR2$adjr2==max(L2.AR2$adjr2))
L2.AR2.Coeff = L2.AR2$which[id.max.adjr2,]
L2.AR2.Coeff
```

```
##     1     2     3     4
## FALSE  TRUE FALSE FALSE
```

```
names(D2.t[,-5])[which(L2.AR2.Coeff==TRUE)]
```

```
## [1] "Acid"
```

We can conclude that the best model according to the "adjusted R-squared" procedure is: Grade.t ∼ Acid.

### C.2.7. backward-AIC

```
L2.B.AIC = step(new.L2.c, direction='backward')
```

```
## Start:  AIC=43.1
## Grade.t ~ Sugar + Acid + Bitter + Pulpy
##
##           Df Sum of Sq    RSS    AIC
## - Bitter   1     1.178 127.81 41.248
## - Sugar    1     1.512 128.15 41.289
## - Pulpy    1     3.648 130.28 41.554
## <none>                 126.64 43.099
## - Acid     1    37.223 163.86 45.223
##
## Step:  AIC=41.25
## Grade.t ~ Sugar + Acid + Pulpy
```

```
##
##          Df Sum of Sq    RSS    AIC
## - Sugar  1     0.635 128.45 39.327
## - Pulpy  1     2.646 130.46 39.575
## <none>               127.81 41.248
## - Acid   1    36.104 163.92 43.228
##
## Step:  AIC=39.33
## Grade.t ~ Acid + Pulpy
##
##          Df Sum of Sq    RSS    AIC
## - Pulpy  1     2.145 130.59 37.592
## <none>               128.45 39.327
## - Acid   1   216.507 344.95 53.133
##
## Step:  AIC=37.59
## Grade.t ~ Acid
##
##         Df Sum of Sq    RSS    AIC
## <none>              130.59 37.592
## - Acid  1    284.62 415.22 54.099
```

```
L2.B.AIC
```

```
##
## Call:
## lm(formula = Grade.t ~ Acid, data = D2.t)
##
## Coefficients:
## (Intercept)         Acid
##      27.392       -2.348
```

We can conclude that the best model with the “backward-AIC” procedure is: Grade.t ∼ Acid.

### C.2.8. backward-Fisher

```
L2.B.Fisher = step(new.L2.c, direction='backward', test='F')
```

```
## Start:  AIC=43.1
## Grade.t ~ Sugar + Acid + Bitter + Pulpy
##
##           Df Sum of Sq    RSS    AIC F value  Pr(>F)
## - Bitter   1     1.178 127.81 41.248  0.1023 0.75508
## - Sugar    1     1.512 128.15 41.289  0.1313 0.72394
## - Pulpy    1     3.648 130.28 41.554  0.3169 0.58476
## <none>               126.64 43.099
## - Acid     1    37.223 163.86 45.223  3.2334 0.09962 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=41.25
## Grade.t ~ Sugar + Acid + Pulpy
##
##          Df Sum of Sq    RSS    AIC F value  Pr(>F)
## - Sugar  1     0.635 128.45 39.327  0.0596 0.81129
## - Pulpy  1     2.646 130.46 39.575  0.2484 0.62720
## <none>               127.81 41.248
## - Acid   1    36.104 163.92 43.228  3.3898 0.09045 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=39.33
## Grade.t ~ Acid + Pulpy
##
##         Df Sum of Sq    RSS    AIC F value    Pr(>F)
## - Pulpy  1     2.145 130.59 37.592  0.2171 0.6489372
## <none>                128.45 39.327
## - Acid   1   216.507 344.95 53.133 21.9125 0.0004297 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Step:  AIC=37.59
## Grade.t ~ Acid
##
##        Df Sum of Sq    RSS    AIC F value    Pr(>F)
## <none>              130.59 37.592
## - Acid  1    284.62 415.22 54.099  30.513 7.497e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
L2.B.Fisher
```

```
##
## Call:
## lm(formula = Grade.t ~ Acid, data = D2.t)
##
## Coefficients:
## (Intercept)         Acid
##      27.392       -2.348
```

We can conclude that the best model with the “backward-Fisher” procedure is: Grade.t $\sim$ Acid.

### C.2.9. forward-AIC

We create a linear model with just the intercept:

```
new.L2.1 = lm(D2.t$Grade ~ 1, data=D2.t)
L2.F.AIC = step(new.L2.1, scope=list(new.L2.1, new.L2.c), direction='forward')
```

```
## Start:  AIC=54.1
## D2.t$Grade ~ 1
```

The model provided by the “forward-AIC” strategy is Grade.t $\sim$ (Intercept) and thus will not be retained.

### C.2.10. forward-Fisher

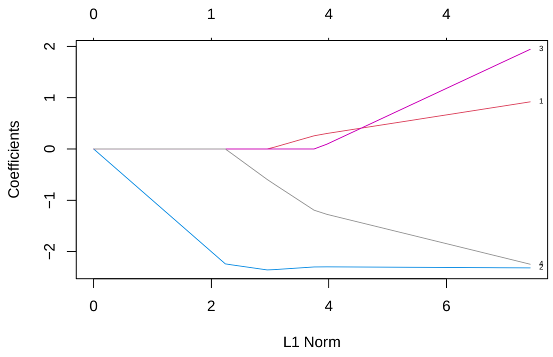
```
L2.F.Fisher = step(new.L2.1, scope=list(new.L2.1, new.L2.c)
                   , direction='forward', test='F')
```

```
## Start:  AIC=54.1
## D2.t$Grade ~ 1
```

The model provided by the “forward-Fisher” strategy is Grade.t $\sim$ (Intercept) and thus will not be retained.

### C.2.11. LASSO

```
X2 = data.matrix(D2.t[,-5])
Y2 = D2.t$Grade
D2.Lasso = glmnet(X2, Y2, alpha=1)
plot(D2.Lasso, col=c(2,4,6,8), label=TRUE)
```



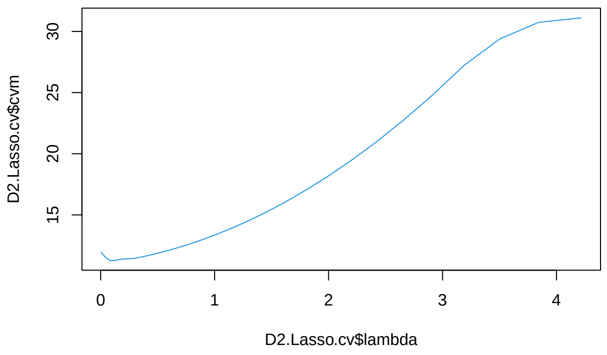
This graph gives the trajectories of the $\beta_i$ obtained for different values of $\lambda$ with respect to the $\mathcal{L}_1$ norm of the $\beta$ vector.
When we decrease $\lambda$ (increase the $\mathcal{L}_1$ norm), some coefficients become different from 0 but not all at the same time.
$\Rightarrow$ We obtain many models, one for each value of $\lambda$ and we select the best value of $\lambda$.

To perform this, we apply 'cv.glmnet" function to compute the cross validation error for each model. Then we keep the best model presenting the lowest error.
Since we have a small dataset, we will take the number of the splitting in K-folds "nfolds = 3" (by default nfolds = 10).

```
# Cross validation error
D2.Lasso.cv = cv.glmnet(X2, Y2, alpha=1, nfolds=3)
D2.Lasso.cv
```

```
##
## Call:  cv.glmnet(x = X2, y = Y2, nfolds = 3, alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  0.093    42   11.27 4.929       3
## 1se  1.516    12   15.53 0.875       1
```

```
plot(D2.Lasso.cv$lambda, D2.Lasso.cv$cvm, type='l', col=4)
```



In order to delete the instability due to the search of a minimum, we look for the “1se” row and not the “min”.

```
# Saving the best lambda from "1se" row:
D2.best.Lambd = D2.Lasso.cv$lambda.1se
# Selection of the variables associated with the best lambda:
D2.best.Lasso = glmnet(X2, Y2, alpha=1, lambda=D2.best.Lambd)
coefficients(D2.best.Lasso)
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                    s0
## (Intercept) 22.618734
## Sugar        .
## Acid        -1.504126
## Bitter       .
## Pulpy        .
```

We can conclude that the best model with Lasso method is: Grade.t ∼ Acid.

### C.2.12. Initial results

All the previous procedures return the same model: Grade.t ∼ Acid. However since we have a small dataset we need to apply Bagging so that the dataset can appear more populated than it really is.

This is the objective of the next section in which we combine Bagging with the computation of the testing error.

### C.2.13. Computation of the testing error with Bagging

- **Computation of the testing error**

To compute the testing error using Bagging, we repeat the following steps (100 times): - We split te initial dataset at random into learning sample and testing sample.

- We use the learning sample to create the linear model using the set of the explanatory variables determined by each of the considered procedures (Adjusted R squared, step by step backward, step by step forward, Lasso, VSURF)

- We use the testing sample to compute the error for all the created models.

The below code implements all these steps.

```
# Initialization
n.row = dim(D2.t)[1]
n.col = dim(D2.t)[2]
n.learn = floor(2 * n.row / 3)
u = 1 : n.row
k = 100

df.best.L2b = data.frame()
df.error = data.frame(ind=integer(), Procedure=character(), Error=double())

for (i in 1:k){
  #---- Splitting the dataset into learning and testing datasets ---------------
  l = sample(u, n.learn, replace=FALSE)
  D.learn = D2.t[l,]
  D.test = D2.t[-l,]
  X.learn = D.learn[, -5]
  X.learn = data.matrix(X.learn)
```

```r
  df.X.learn = as.data.frame(X.learn)
  Y.learn = D.learn$Grade.t
  X.test = D.test[, -5]
  Y.test = D.test$Grade.t

  #----- Creation of the model with all the explanatory variables --------------
  L2b.split.c = lm(Grade.t ~ ., data=D.learn)

  #----- Creation of the model with only the intercept -------------------------
  L2b.split.1 = lm(Grade.t ~ 1, data=D.learn)

  #============================================================================#
  #                      ----- Adjusted R squared -----                        #
  #============================================================================#
 cat(' \n ##### Adjusted R squared ############################## ', i, ' # \n')

  #----- Variables selection ---------------------------------------------------
  D2b.best.AdjR2 = leaps(df.X.learn, Y.learn, method='adjr2', nbest=1)
  # Selection of the index of the biggest adjusted R squared:
  id.max.adjr2 = which(D2b.best.AdjR2$adjr2==max(D2b.best.AdjR2$adjr2))
  # Selection of the variables:
  Coef.D2b.best.AdjR2 = D2b.best.AdjR2$which[id.max.adjr2,]
  # Storing the names of the selected variables:
  nm.Coef.best.AdjR2 = names(df.X.learn)[which(Coef.D2b.best.AdjR2==TRUE)]

  #----- Creation of the model with the selected variables ---------------------
  # Concatenating the names of the variables in a string:
  formula.Coef.best.AdjR2 = paste(nm.Coef.best.AdjR2, collapse = '+')
  # Storing the formula of the model in a string:
  str.L2b.AdjR2 = sprintf('Grade.t ~  %s', formula.Coef.best.AdjR2)
  # Computing the model with the selected variables:
  best.L2b.AdjR2 = do.call('lm', list(str.L2b.AdjR2, quote(D.learn)))

  #----- Prediction ------------------------------------------------------------
  Y.prdc.AdjR2 = predict(best.L2b.AdjR2, newdata=D.test)

  #----- Computation of the error ----------------------------------------------
  err.AdjR2 = (1 / nrow(X.test)) * sum((Y.test - Y.prdc.AdjR2)^2)
  # Storing the error in a data frame:
  df.err.AdjR2 = data.frame(ind=i, Procedure='AdjR2', Error=err.AdjR2)
  df.error = bind_rows(df.error, df.err.AdjR2)

  #----- Saving the coefficients of the best model -----------------------------
  Coef.best.L2b.AdjR2 = best.L2b.AdjR2$coefficients
  # Storing the coefficients in a data frame:
  df.Coef.AdjR2 = data.frame(as.list(Coef.best.L2b.AdjR2))
  df.Coef.AdjR2 = data.frame(id=i, Procedure='AdjR2', df.Coef.AdjR2)
  df.best.L2b = bind_rows(df.best.L2b, df.Coef.AdjR2)

  #----- Deletion of the intermediate variables --------------------------------
  rm(D2b.best.AdjR2, id.max.adjr2, Coef.D2b.best.AdjR2, nm.Coef.best.AdjR2
     , formula.Coef.best.AdjR2, str.L2b.AdjR2, best.L2b.AdjR2, Y.prdc.AdjR2
     , err.AdjR2, df.err.AdjR2, Coef.best.L2b.AdjR2, df.Coef.AdjR2)
```

```
#==============================================================================#
#          ----- Step by step "Backward" method with AIC test -----           #
#==============================================================================#
cat(' \n ##### Backward - AIC ################################### ', i, ' # \n')

#----- Variables selection ----------------------------------------------------
L2b.Back.AIC = step(L2b.split.c, direction='backward')
# formula of the model with the selected variables:
L2b.Back.AIC.formula = L2b.Back.AIC$call$formula

#----- Creation of the model with the selected variables ----------------------
best.L2b.Back.AIC = lm(L2b.Back.AIC.formula, data=D.learn)

#----- Prediction -------------------------------------------------------------
Y.prdc.Back.AIC = predict(best.L2b.Back.AIC, newdata=X.test)

#----- Computation of the error -----------------------------------------------
err.Back.AIC = (1 / nrow(X.test)) * sum((Y.test - Y.prdc.Back.AIC)^2)
# Storing the error in a data frame:
df.err.Back.AIC = data.frame(ind=i, Procedure='Backword_AIC', Error=err.Back.AIC)
df.error = bind_rows(df.error, df.err.Back.AIC)

#----- Saving the coefficients of the best model ------------------------------
Coef.best.L2b.Back.AIC = best.L2b.Back.AIC$coefficients
# Storing the coefficients in a data frame:
df.Coef.Back.AIC = data.frame(as.list(Coef.best.L2b.Back.AIC))
df.Coef.Back.AIC = data.frame(id=i, Procedure='Backword_AIC', df.Coef.Back.AIC)
df.best.L2b = bind_rows(df.best.L2b, df.Coef.Back.AIC)

#----- Deletion of the intermediate variables ---------------------------------
rm(L2b.Back.AIC, L2b.Back.AIC.formula, best.L2b.Back.AIC, Y.prdc.Back.AIC
   , err.Back.AIC, df.err.Back.AIC, Coef.best.L2b.Back.AIC, df.Coef.Back.AIC)

#==============================================================================#
#          ----- Step by step "Forward" method with AIC test -----            #
#==============================================================================#
cat(' \n ##### Forward  - AIC ################################### ', i, ' # \n')

#----- Variables selection ----------------------------------------------------
L2b.Forw.AIC = step(L2b.split.c, scope=list(L2b.split.1, L2b.split.c)
                    , direction='forward')
L2b.Forw.AIC.formula = L2b.Forw.AIC$call$formula

#----- Creation of the model with the selected variables ----------------------
best.L2b.Forw.AIC = lm(L2b.Forw.AIC.formula, data=D.learn)

#----- Prediction -------------------------------------------------------------
Y.prdc.Forw.AIC = predict(best.L2b.Forw.AIC, newdata=D.test)

#----- Computation of the error -----------------------------------------------
err.Forw.AIC = (1 / nrow(X.test)) * sum((Y.test - Y.prdc.Forw.AIC)^2)
# Storing the error in a data frame:
df.err.Forw.AIC = data.frame(ind=i, Procedure='Forward_AIC', Error=err.Forw.AIC)
df.error = bind_rows(df.error, df.err.Forw.AIC)
```

```
#----- Saving the coefficients of the best model ------------------------------
Coef.best.L2b.Forw.AIC = best.L2b.Forw.AIC$coefficients
# Storing the coefficients in a data frame:
df.Coef.Forw.AIC = data.frame(as.list(Coef.best.L2b.Forw.AIC))
df.Coef.Forw.AIC = data.frame(id=i, Procedure='Forward_AIC', df.Coef.Forw.AIC)
df.best.L2b = bind_rows(df.best.L2b, df.Coef.Forw.AIC)

#----- Deletion of the intermediate variables ---------------------------------
rm(L2b.Forw.AIC, L2b.Forw.AIC.formula, best.L2b.Forw.AIC, Y.prdc.Forw.AIC
   , err.Forw.AIC, df.err.Forw.AIC, Coef.best.L2b.Forw.AIC, df.Coef.Forw.AIC)

#=============================================================================#
#                              ----- LASSO -----                              #
#=============================================================================#
cat(' \n ##### LASSO ########################################### ', i, ' # \n')

#----- Variables selection ----------------------------------------------------
# Computing of the cross validation error
D2b.Lasso.cv = cv.glmnet(X.learn, Y.learn, alpha=1, nfolds=3)
# Storing the best lambda from "1se" row:
D2b.best.Lambd = D2b.Lasso.cv$lambda.1se
# Selection of variables:
D2b.best.Lasso = glmnet(X.learn, Y.learn, alpha=1, lambda=D2b.best.Lambd)
Coef.D2b.best.Lasso = coefficients(D2b.best.Lasso)
# Selecting the names of variables whose coefficients are not null:
nm.Coef.best.Lasso = names(which(Coef.D2b.best.Lasso[,1]!=0))
# Rename the "(Intercept)" as "1":
nm.Coef.best.Lasso[1] = '1'

#----- Creation of the model with the selected variables ---------------------
# Concatenating the names of the variables in a string:
formula.Coef.best.Lasso = paste(nm.Coef.best.Lasso, collapse = '+')
# Storing the formula of the model in a string:
str.L2b.Lasso = sprintf('Grade.t ~  %s', formula.Coef.best.Lasso)
# Computing the model with the selected variables:
best.L2b.Lasso = do.call('lm', list(str.L2b.Lasso, quote(D.learn)))

#----- Prediction -------------------------------------------------------------
Y.prdc.Lasso = predict(best.L2b.Lasso, newdata=D.test)

#----- Computation of the error -----------------------------------------------
err.Lasso = (1 / nrow(X.test)) * sum((Y.test - Y.prdc.Lasso)^2)
# Storing the error in a data frame:
df.err.Lasso = data.frame(ind=i, Procedure='LASSO', Error=err.Lasso)
df.error = bind_rows(df.error, df.err.Lasso)

#----- Saving the coefficients of the best model ------------------------------
Coef.best.L2b.Lasso = best.L2b.Lasso$coefficients
# Storing the coefficients in a data frame:
df.Coef.Lasso = data.frame(as.list(Coef.best.L2b.Lasso))
df.Coef.Lasso = data.frame(id=i, Procedure='LASSO', df.Coef.Lasso)
df.best.L2b = bind_rows(df.best.L2b, df.Coef.Lasso)

#----- Deletion of the intermediate variables ---------------------------------
rm(D2b.Lasso.cv, D2b.best.Lambd, D2b.best.Lasso, Coef.D2b.best.Lasso
    , nm.Coef.best.Lasso, formula.Coef.best.Lasso, str.L2b.Lasso, best.L2b.Lasso
    , Y.prdc.Lasso, err.Lasso, df.err.Lasso, Coef.best.L2b.Lasso, df.Coef.Lasso)

  #============================================================================#
  #          ----- VSURF: Variable Selection Using Random Forests -----         #
  #============================================================================#
  cat(' \n ##### VSURF ########################################### ', i, ' # \n')

  #----- Variables selection -------------------------------------------------
  Var.Selec.VSURF = VSURF(df.X.learn, Y.learn)
  # Storing the names of the selected variables:
  nm.Coef.VSURF = colnames(df.X.learn[Var.Selec.VSURF$varselect.pred])

  #----- Creation of the model with the selected variables ---------------------
  # Add the "Intercept" to the names of the selected variables:
  nm.Coef.VSURF = c('1', nm.Coef.VSURF)
  # Concatenating the names of the variables in a string:
  formula.Coef.best.VSURF = paste(nm.Coef.VSURF, collapse = '+')
  # Storing the formula of the model in a string:
  str.L2b.VSURF = sprintf('Grade.t ~  %s', formula.Coef.best.VSURF)
  # Computing the model with the selected variables:
  best.L2b.VSURF = do.call('lm', list(str.L2b.VSURF, quote(D.learn)))

  #----- Prediction ------------------------------------------------------------
  Y.prdc.VSURF = predict(best.L2b.VSURF, newdata=D.test)

  #----- Computation of the error ----------------------------------------------
  err.VSURF = (1 / nrow(X.test)) * sum((Y.test - Y.prdc.VSURF)^2)
  # Storing the error in a data frame:
  df.err.VSURF = data.frame(ind=i, Procedure='VSURF', Error=err.VSURF)
  df.error = bind_rows(df.error, df.err.VSURF)

  #----- Saving the coefficients of the best model -----------------------------
  Coef.best.L2b.VSURF = best.L2b.VSURF$coefficients
  # Storing the coefficients in a data frame:
  df.Coef.VSURF = data.frame(as.list(Coef.best.L2b.VSURF))
  df.Coef.VSURF = data.frame(id=i, Procedure='VSURF', df.Coef.VSURF)
  df.best.L2b = bind_rows(df.best.L2b, df.Coef.VSURF)

  #----- Deletion of the intermediate variables --------------------------------
  rm(Var.Selec.VSURF, nm.Coef.VSURF, formula.Coef.best.VSURF, str.L2b.VSURF
    , best.L2b.VSURF, Y.prdc.VSURF, err.VSURF, df.err.VSURF, Coef.best.L2b.VSURF
    , df.Coef.VSURF)
}
```

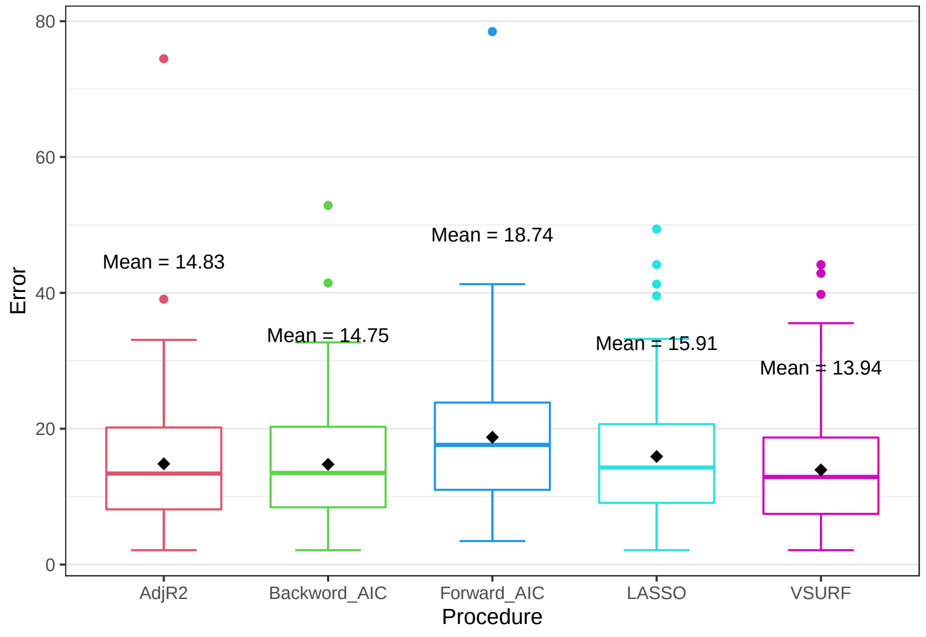
The above code returns the coefficients of all models for each learning sample in the variable "df.best.L2b" as well as their associated testing error stored in the variable "df.error".

To compare between all procedures we compute the average of their testing errors and select the one that presents the lowest average.

The obtained results are displayed in the below boxplot.

They show that the "VSURF" is the procedure that presents the lowest average testing error.

```
# Initialization of the labels of the means:
label.mean = function(x) {
  data.frame(y = (mean(x) + max(x)) / 2
             , label = paste('Mean =', round(mean(x), digit=2)))
}
# Plotting the box plots:
plot.Error = ggboxplot(df.error, x = 'Procedure', y = 'Error'
                       , bxp.errorbar = TRUE
                       , palette = c(2:6)
                       , color = 'Procedure')

plot.Error + theme_bw() + theme(legend.position = 'None') + rremove('x.grid') +
  stat_summary(fun.data=label.mean, geo='text', size=3.5) +
  stat_summary(fun=mean, shape=18, size=0.6)
```



```
# Extraction of the mean for each procedure:
list.mean.error = df.error %>%
  group_by(Procedure) %>%
  summarise(mean_error=mean(Error))
list.mean.error
```

```
## # A tibble: 5 x 2
##   Procedure    mean_error
##   <chr>             <dbl>
## 1 AdjR2              14.8
## 2 Backword_AIC       14.8
## 3 Forward_AIC        18.7
## 4 LASSO              15.9
## 5 VSURF              13.9
```

- **Creation of the model**

The best model is obtained by averaging all the coefficients associated with the models of the procedure “VSURF”.

The below code implements this step and provides the best model.

```
# Storing the index of the smallest mean:
id.min.error = which.min(as.data.frame(list.mean.error)[,2])
id.min.error
```

```
## [1] 5
```

```
# Replacement of the "NA" values by 0 in the data frame "df.best.L2b":
df.best.L2b[is.na(df.best.L2b)] = 0

# Computation of the mean of each coefficient:
list.mean.coeff = df.best.L2b %>%
  group_by(Procedure) %>%
  summarise(Intercept=mean(X.Intercept.)
            , Sugar=mean(Sugar)
            , Acid=mean(Acid)
            , Bitter=mean(Bitter)
            , Pulpy=mean(Pulpy))

# The best model
best.model = list.mean.coeff[id.min.error,]
best.model
```

```
## # A tibble: 1 x 6
##   Procedure     Intercept     Sugar     Acid      Bitter      Pulpy
##   <chr>             <dbl>     <dbl>    <dbl>       <dbl>      <dbl>
## 1 VSURF              23.7     0.388    -2.01      -0.102     -0.186
```