TP n°3 - Modèle de régression Ridge et Lasso & Agrégation de modèles

Salim Nadir et Guillaume Ostrom

Application I : Régression Ridge et Lasso

Analyse préliminaire

Chargement du fichiers **usa_indicators.txt**:

```
setwd("D:\\Centrale\\OMA\\SDMA\\TP3\\")
usaIndicator = read.table('usa_indicators.txt', sep = ';', header = T)
nrow(usaIndicator)
```

```
## [1] 14
```

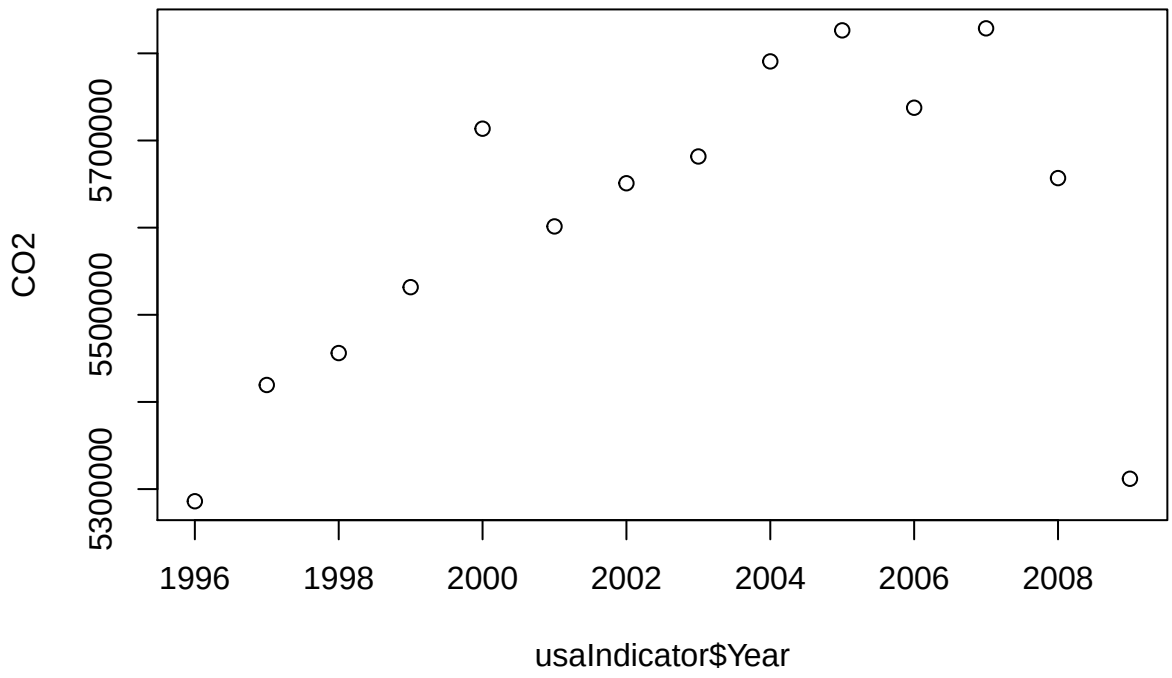
```
ncol(usaIndicator)
```

```
## [1] 110
```

On observe 14 observations pour 110 variables. Ce n'est pas un problème de régression évident, car la dimension est supérieure largement au nombre d'observations. On note un risque important d'overfitting. C'est le fléau de la dimension. Il faut donc regarder les méthodes de Ridge et Lasso, car ces méthodes vont introduire des critères de pénalisation pour les coefficients.

La variable contenant la quantité de CO2 émise par an est **EN.ATM.CO2E.KT**

```
CO2 = as.matrix(usaIndicator["EN.ATM.CO2E.KT"])
plot(usaIndicator$Year, CO2)
```



Les données étant de nature très diverse, les ordres de grandeur peuvent varier d'un indicateur à l'autre. Le coefficient de régression associéé risque à l'inverse d'être d'ordre de grandeur très faible, ce qui pourrait impliquer à tort qu'une variable importante ne l'est pas. La régression peut donc être impactée, nous allons devoir normaliser les données.

```
usaIndicatorScale = as.data.frame(scale(usaIndicator, center=FALSE))
```

Régression Ridge

1. Définition régression de type Ridge

La méthode de régression de type Ridge revient à trouver :

$$\widehat{\beta}_{RR} = \text{argmin}_{\beta}(\|Y - X\beta\|_2^2 + \lambda\|\beta\|_2^2)$$

avec $\lambda > 0$, un paramètre fixé. On va chercher une valeur optimale de λ par cross validation.

```
library(MASS)
```

2. Régression de Ridge pour $\lambda = 0$ et $\lambda = 100$

On récupéres les 5 coefficients les plus influents pour des régressions de Ridge pour $\lambda = 0$ et $\lambda = 100$.

```r
# Retrait de la colonne Year
usaIndicatorScale = subset(usaIndicatorScale, select = -Year)
# Ridge Lambda 0
resridge0 = lm.ridge(EN.ATM.CO2E.KT~., lambda = 0, data = usaIndicatorScale)
coef0 = coef(resridge0)
coef0_2 = resridge0$coef
# Ridge Lambda 100
resridge100 = lm.ridge(EN.ATM.CO2E.KT~., lambda = 100, data = usaIndicatorScale)
coef100 =coef(resridge100)
coef100_2 = resridge100$coef
# Ridge
sort0 = as.data.frame(sort(abs(coef0), decreasing = T)[1:6])
sort100 = as.data.frame(sort(abs(coef100), decreasing = T)[1:6])
print(sort0)
```

```
##                   sort(abs(coef0), decreasing = T)[1:6]
## AG.LND.TOTL.K2                                2.9003002
##                                               2.2198184
## EG.USE.COMM.FO.ZS                             0.5070829
## AG.LND.AGRI.K2                                0.2703058
## SP.RUR.TOTL                                   0.2456370
## AG.SRF.TOTL.K2                                0.1668424
```

```r
print(sort100)
```

```
##                   sort(abs(coef100), decreasing = T)[1:6]
## AG.LND.TOTL.K2                                 1.61377948
##                                                0.46300987
## SP.RUR.TOTL                                    0.13532244
## EG.USE.COMM.FO.ZS                              0.11966818
## AG.SRF.TOTL.K2                                 0.09561493
## SP.POP.65UP.TO.ZS                              0.07710010
```

Les 5 indicateurs qui sont les plus influents pour un paramètre $\lambda = 0$:
- Superficie de terrain (AG.LND.TOTL.K2)
- Consommation d'énergie fossile (EG.USE.COMM.FO.ZS)
- Superficie de cultures agricoles (AG.LND.AGRI.K2)
- Population rurale (SP.RUR.TOTL)
- Superficie totale (AG.SRF.TOTL.K2)

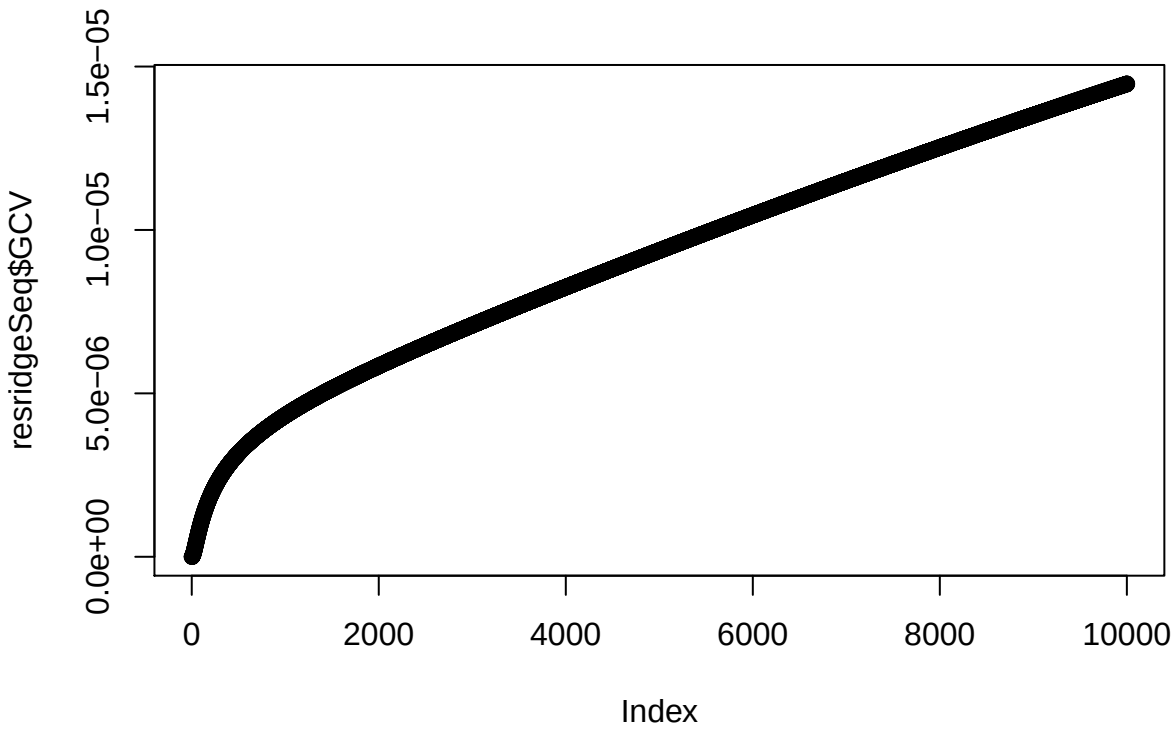
Les 5 indicateurs qui sont les plus influents pour un paramètre $\lambda = 100$:
- Superficie de terrain (AG.LND.TOTL.K2)
- Population rurale (SP.RUR.TOTL)
- Consommation d'énergie fossile (EG.USE.COMM.FO.ZS)
- Superficie totale (AG.SRF.TOTL.K2)
- Population +65 ans (% of total) (SP.POP.65UP.TO.ZS)

Nous observons que les variables pour $\lambda = 0$ et $\lambda = 100$ sont senssiblement les mêmes. Mais il n'y a pas de corrélation évidente avec le CO2 émis. On note 3 indicateurs de superficie pour $\lambda = 0$ ce qui laisse penser à une corrélation entre ces variables.

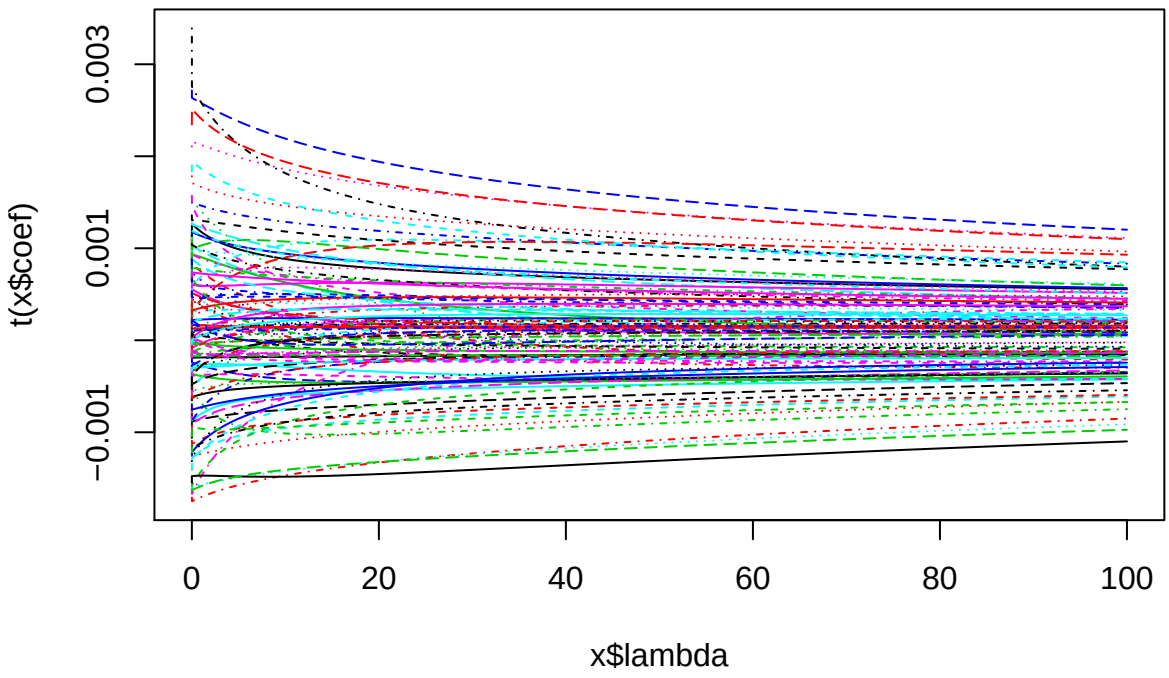
3. Régression sur l'ensemble des valeurs de pénalisation

Courbes des performances de la régression de Ridge.

```r
lambdaSeq = seq(0, 100, by = 0.01)
resridgeSeq = lm.ridge(EN.ATM.CO2E.KT~., lambda = lambdaSeq, data = usaIndicatorScale)
plot(resridgeSeq$GCV) # Performance de Lambda par cross-validation.
```




```r
plot(resridgeSeq) # Valeurs des coefficients selon lambda.
```



On prends le meilleur $\lambda = 0.1$

```
bestLambda = lambdaSeq[which.min(resridgeSeq$GCV)]
print(bestLambda)
```

```
## [1] 0.01
```

```
bestResridge = lm.ridge(EN.ATM.CO2E.KT~., lambda = bestLambda, data = usaIndicatorScale)
bestcoef = coef(bestResridge)
print(bestcoef)
```

```
##                     AG.AGR.TRAC.NO    AG.LND.AGRI.K2    AG.LND.CREL.HA
##      -2.068541e+00      8.249449e-02      2.027198e-01      2.670341e-02
##    AG.LND.TOTL.K2    AG.PRD.CREL.MT    AG.PRD.FOOD.XD    AG.PRD.LVSK.XD
##       2.903792e+00      1.088765e-02      9.143461e-03      1.688029e-03
##    AG.SRF.TOTL.K2 EG.EGY.PROD.KT.OE    EG.ELC.COAL.KH    EG.ELC.HYRO.KH
##      -1.798206e-01     -4.520155e-02      6.956609e-02     -6.909666e-03
##    EG.ELC.LOSS.KH    EG.ELC.NGAS.KH    EG.ELC.NUCL.KH    EG.ELC.PETR.KH
##       1.388324e-02      2.285951e-03      2.600143e-03      3.735170e-03
##    EG.ELC.PROD.KH    EG.ELC.RNEW.KH    EG.IMP.CONS.ZS EG.USE.COMM.FO.ZS
##       2.180345e-02     -1.213700e-02      1.955830e-02      4.123885e-01
## EG.USE.COMM.KT.OE    EN.CO2.BLDG.MT       EN.POP.DNST       EN.URB.LCTY
##       8.382940e-02      2.111437e-02      2.661500e-03      2.173089e-02
## FD.AST.PRVT.GD.ZS    FI.RES.TOTL.CD    FM.AST.DOMS.CN    FM.LBL.MONY.CN
##       1.115862e-02     -2.537902e-03      6.303422e-04     -1.517319e-02
##       FP.CPI.TOTL    FP.CPI.TOTL.ZG       FR.INR.LEND       FR.INR.RINR
```

```
##      4.986157e-04      1.608857e-03     -3.154921e-04     -5.094251e-04
##    IP.JRN.ARTC.SC       IP.PAT.NRES       IP.PAT.RESD       IP.TMK.TOTL
##     -8.041035e-03     -1.246753e-03      2.877671e-03      1.099077e-02
##       IS.AIR.PSGR    IS.ROD.ENGY.KT    IS.ROD.SGAS.KT       IT.MLT.MAIN
##      7.198471e-03      2.375793e-02      4.468715e-02      1.169993e-02
##    MS.MIL.MPRT.KD    MS.MIL.TOTL.P1 MS.MIL.TOTL.TF.ZS    MS.MIL.XPND.CN
##     -7.304120e-04     -8.338215e-03     -7.244471e-03     -1.013992e-03
##    MS.MIL.XPRT.KD    NE.CON.GOVT.CD    NE.CON.PETC.CD    NE.DAB.TOTL.CD
##      1.255069e-03     -2.009560e-04      1.132010e-03      1.652242e-03
##    NE.EXP.GNFS.CD    NE.TRD.GNFS.ZS NY.GDP.DEFL.KD.ZG NY.GDP.FRST.RT.ZS
##      1.941022e-03      1.191633e-02      1.028696e-03     -1.842392e-03
## NY.GDP.TOTL.RT.ZS    NY.GDS.TOTL.CD    NY.GSR.NFCY.CD    NY.TRF.NCTR.CD
##      1.111320e-03      8.785549e-03     -3.017990e-03     -1.527741e-03
## SE.ENR.TERT.FM.ZS       SE.PRE.ENRR    SE.PRE.ENRR.FE       SE.PRM.ENRL
##      1.080761e-02      1.035755e-02      1.086389e-02     -3.703018e-02
## SE.PRM.ENRL.FE.ZS SE.PRM.ENRL.TC.ZS       SE.PRM.ENRR       SE.PRM.NENR
##     -1.468046e-01     -1.149812e-02     -8.722225e-02     -4.806035e-02
##    SE.PRM.PRIV.ZS       SE.PRM.UNER       SE.SEC.ENRL       SE.TER.ENRR
##     -1.283104e-02      4.241249e-03      5.002632e-04      1.943310e-03
## SE.XPD.TOTL.GB.ZS       SH.DTH.IMRT       SH.DTH.MORT       SH.DYN.MORT
##      5.521609e-03     -3.118160e-03     -6.971499e-03     -5.517670e-03
##    SH.MED.BEDS.ZS    SL.EMP.SELF.ZS SL.UEM.TOTL.NE.ZS       SP.ADO.TFRT
##      2.308934e-04     -2.595260e-02     -7.889269e-03     -1.620422e-03
##    SP.DYN.AMRT.FE    SP.DYN.AMRT.MA    SP.DYN.CBRT.IN    SP.DYN.CDRT.IN
##     -4.480697e-02     -3.607397e-02      3.877559e-02     -4.441653e-02
##    SP.DYN.LE00.IN SP.POP.0014.TO.ZS SP.POP.1564.TO.ZS SP.POP.65UP.TO.ZS
##      6.724651e-02     -7.714770e-04      5.419512e-02     -1.343028e-01
##       SP.POP.GROW SP.POP.SCIE.RD.P6       SP.POP.TOTL       SP.RUR.TOTL
##      3.025814e-04     -5.646966e-03      3.253829e-03     -1.901862e-01
##       SP.URB.TOTL TG.VAL.TOTL.GD.ZS TM.VAL.AGRI.ZS.UN TM.VAL.FOOD.ZS.UN
##      3.035218e-03      1.335730e-02     -6.055878e-04     -1.927735e-02
## TM.VAL.FUEL.ZS.UN TM.VAL.MANF.ZS.UN TM.VAL.MMTL.ZS.UN TM.VAL.MRCH.AL.ZS
##     -2.818138e-04      4.945700e-03      1.302916e-03      6.014206e-04
## TM.VAL.MRCH.CD.WT TM.VAL.MRCH.HI.ZS TM.VAL.MRCH.OR.ZS TM.VAL.MRCH.R1.ZS
##      2.680081e-03      7.149058e-04     -4.730982e-04     -8.891375e-04
## TM.VAL.MRCH.R3.ZS TM.VAL.MRCH.R4.ZS TM.VAL.MRCH.R5.ZS TM.VAL.MRCH.R6.ZS
##      5.864368e-03      1.700212e-05     -1.293134e-03      6.764683e-05
## TX.VAL.AGRI.ZS.UN TX.VAL.FOOD.ZS.UN TX.VAL.FUEL.ZS.UN TX.VAL.MANF.ZS.UN
##     -7.012103e-03     -1.417639e-02     -1.676849e-03      3.862605e-02
## TX.VAL.MMTL.ZS.UN
##     -5.709292e-04
```

On pénalise les grandes valeurs de $\|\beta\|_2^2$ ce qui impliquent que les coefficients se rapprochent relativement de 0 plus λ croit. Notre cross-validation, donne un $\lambda_o ptimal = 0.01$ qui minimise l'erreur du modèle.

4. Erreur quadritique moyenne entre les données cibles.

```
Ones= c(1)
X = merge(Ones, usaIndicatorScale)
X$Year <- NULL
X$EN.ATM.CO2E.KT <- NULL
```

```
Yridge =(Yridge=as.matrix(X)%*%as.vector(bestcoef))
error = sum((usaIndicatorScale$EN.ATM.CO2E.KT - Yridge)^2)/length(Yridge)
print(error)
```

```
## [1] 3.590837e-11
```

L'erreur quadratique est de $3,6.10^{-11}$, soit une erreur très faible. La grandeur de la dimension econduit à l'overfitting.

Régression Lasso

5. Import de la bibliothéque

Dans une régression de type Lasso on cherche à trouver

$$\widehat{\beta}_{RL} = \text{argmin}_{\beta}(\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1)$$

avec $\lambda > 0$, un paramètre fixé. On va chercher une valeur optimale de λ par cross validation.

Import de la bibliothéque lars:

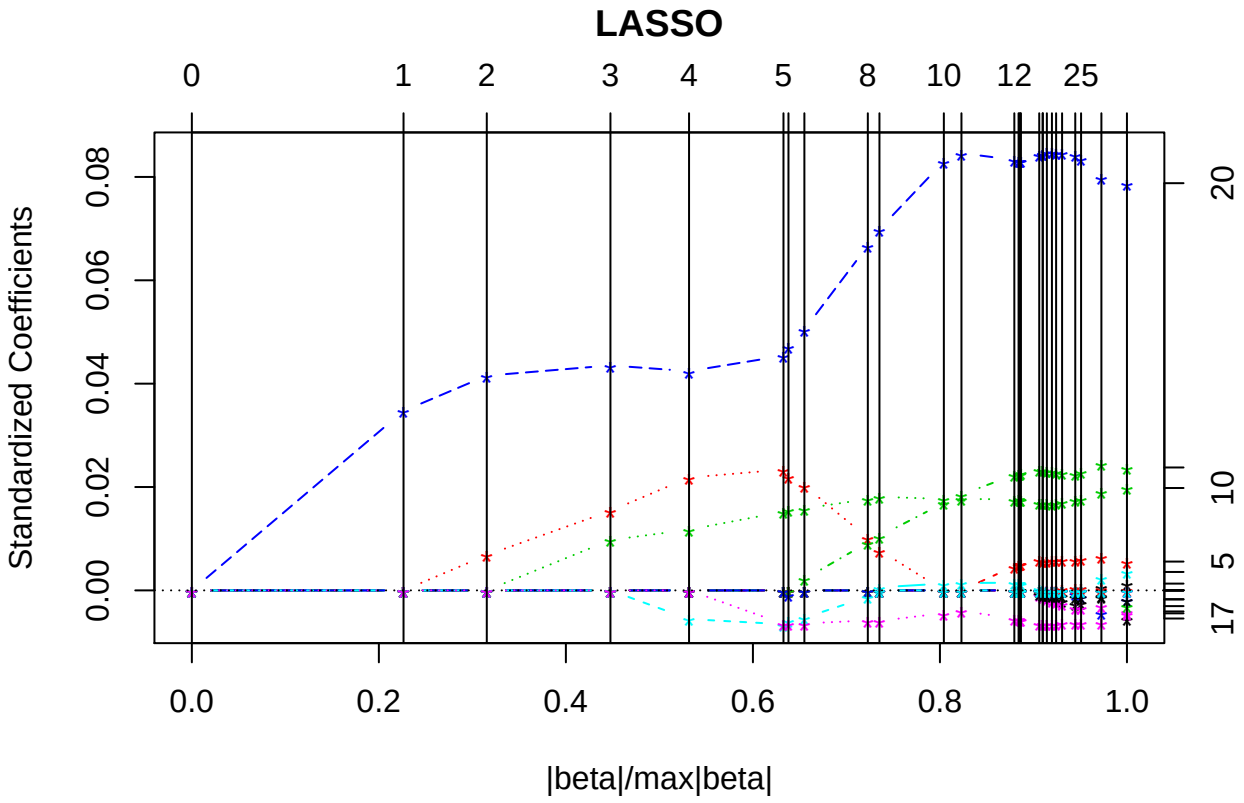
```
#install.packages("lars")
library("lars")
```

```
## Warning: package 'lars' was built under R version 3.3.2
```

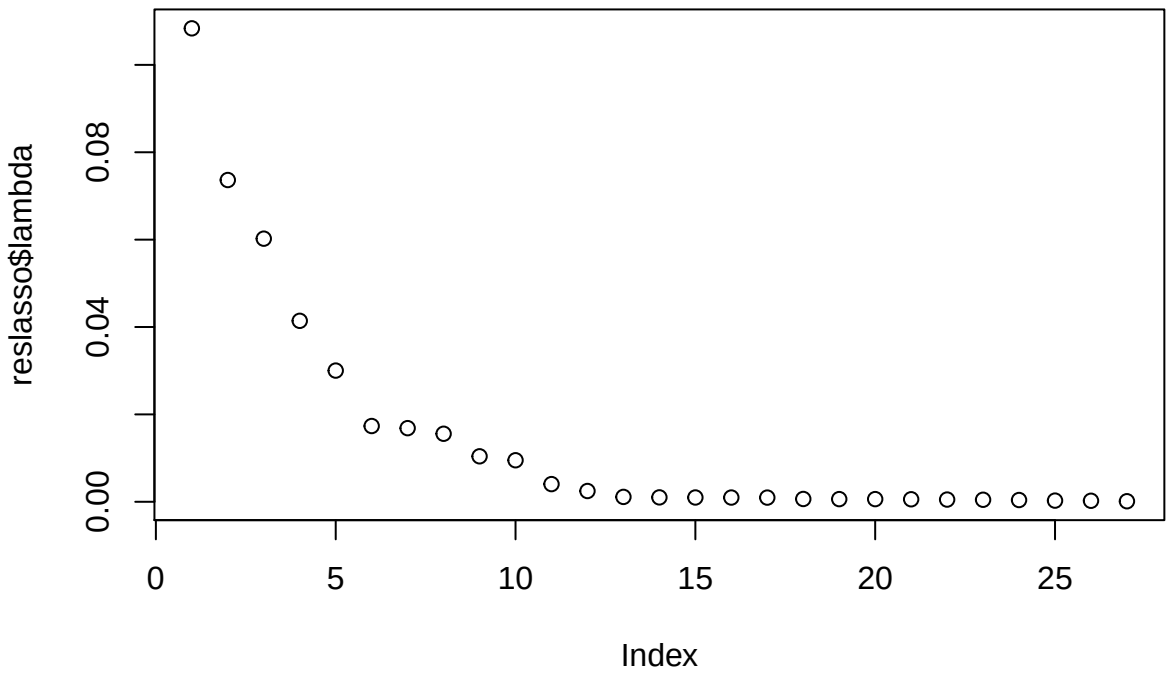
```
## Loaded lars 1.2
```

6. Régression de Lasso

```
X = usaIndicatorScale
X$Year <-  NULL
X$EN.ATM.CO2E.KT <- NULL
X = as.matrix(X)
reslasso = lars(X, usaIndicatorScale$EN.ATM.CO2E.KT, type='lasso')
plot(reslasso) # Régression Lasso
```




```
plot(reslasso$lambda)
```




```
#print(reslasso$beta)
```

La régression Lasso montre selon les 27 valeurs de λ les valeurs des coefficients. Les λ croient de la droit vers la gauche de l'axe des abscisses. Un $\lambda = 0$ est équivalent à une régression linéaire simple.

7. Réduction du nombre de variables explicatives

Les coefficients pour $\lambda = 0$ sont:

```
coef=predict.lars(reslasso,X,type="coefficients",mode="lambda",s=0)
print(coef)
```

```
## $s
## [1] 0
##
## $fraction
## [1] 1
##
## $mode
## [1] "lambda"
##
## $coefficients
##    AG.AGR.TRAC.NO    AG.LND.AGRI.K2    AG.LND.CREL.HA    AG.LND.TOTL.K2
##      0.000000e+00      0.000000e+00      0.000000e+00      6.797761e-01
```

```
##    AG.PRD.CREL.MT    AG.PRD.FOOD.XD    AG.PRD.LVSK.XD    AG.SRF.TOTL.K2
##      1.583399e-02      0.000000e+00      0.000000e+00      0.000000e+00
## EG.EGY.PROD.KT.OE    EG.ELC.COAL.KH    EG.ELC.HYRO.KH    EG.ELC.LOSS.KH
##      0.000000e+00      1.398132e-01     -3.557505e-03      8.990883e-03
##    EG.ELC.NGAS.KH    EG.ELC.NUCL.KH    EG.ELC.PETR.KH    EG.ELC.PROD.KH
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##    EG.ELC.RNEW.KH    EG.IMP.CONS.ZS EG.USE.COMM.FO.ZS EG.USE.COMM.KT.OE
##     -1.375454e-02      0.000000e+00      9.507131e-01      7.017888e-01
##    EN.CO2.BLDG.MT       EN.POP.DNST       EN.URB.LCTY FD.AST.PRVT.GD.ZS
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##    FI.RES.TOTL.CD    FM.AST.DOMS.CN    FM.LBL.MONY.CN       FP.CPI.TOTL
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##    FP.CPI.TOTL.ZG       FR.INR.LEND       FR.INR.RINR    IP.JRN.ARTC.SC
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##       IP.PAT.NRES       IP.PAT.RESD       IP.TMK.TOTL       IS.AIR.PSGR
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##    IS.ROD.ENGY.KT    IS.ROD.SGAS.KT       IT.MLT.MAIN    MS.MIL.MPRT.KD
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##    MS.MIL.TOTL.P1 MS.MIL.TOTL.TF.ZS    MS.MIL.XPND.CN    MS.MIL.XPRT.KD
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##    NE.CON.GOVT.CD    NE.CON.PETC.CD    NE.DAB.TOTL.CD    NE.EXP.GNFS.CD
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##    NE.TRD.GNFS.ZS NY.GDP.DEFL.KD.ZG NY.GDP.FRST.RT.ZS NY.GDP.TOTL.RT.ZS
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##    NY.GDS.TOTL.CD    NY.GSR.NFCY.CD    NY.TRF.NCTR.CD SE.ENR.TERT.FM.ZS
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##       SE.PRE.ENRR    SE.PRE.ENRR.FE       SE.PRM.ENRL SE.PRM.ENRL.FE.ZS
##      0.000000e+00      0.000000e+00     -7.546348e-02     -7.605736e-02
## SE.PRM.ENRL.TC.ZS       SE.PRM.ENRR       SE.PRM.NENR    SE.PRM.PRIV.ZS
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##       SE.PRM.UNER       SE.SEC.ENRL       SE.TER.ENRR SE.XPD.TOTL.GB.ZS
##      6.571287e-05      0.000000e+00      0.000000e+00      0.000000e+00
##       SH.DTH.IMRT       SH.DTH.MORT       SH.DYN.MORT    SH.MED.BEDS.ZS
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##    SL.EMP.SELF.ZS SL.UEM.TOTL.NE.ZS       SP.ADO.TFRT    SP.DYN.AMRT.FE
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
##    SP.DYN.AMRT.MA    SP.DYN.CBRT.IN    SP.DYN.CDRT.IN    SP.DYN.LE00.IN
##     -2.262141e-02      0.000000e+00      0.000000e+00      0.000000e+00
## SP.POP.0014.TO.ZS SP.POP.1564.TO.ZS SP.POP.65UP.TO.ZS       SP.POP.GROW
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
## SP.POP.SCIE.RD.P6       SP.POP.TOTL       SP.RUR.TOTL       SP.URB.TOTL
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
## TG.VAL.TOTL.GD.ZS TM.VAL.AGRI.ZS.UN TM.VAL.FOOD.ZS.UN TM.VAL.FUEL.ZS.UN
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
## TM.VAL.MANF.ZS.UN TM.VAL.MMTL.ZS.UN TM.VAL.MRCH.AL.ZS TM.VAL.MRCH.CD.WT
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
## TM.VAL.MRCH.HI.ZS TM.VAL.MRCH.OR.ZS TM.VAL.MRCH.R1.ZS TM.VAL.MRCH.R3.ZS
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
## TM.VAL.MRCH.R4.ZS TM.VAL.MRCH.R5.ZS TM.VAL.MRCH.R6.ZS TX.VAL.AGRI.ZS.UN
##      0.000000e+00      0.000000e+00      0.000000e+00      0.000000e+00
## TX.VAL.FOOD.ZS.UN TX.VAL.FUEL.ZS.UN TX.VAL.MANF.ZS.UN TX.VAL.MMTL.ZS.UN
##     -1.032529e-02      0.000000e+00      0.000000e+00      0.000000e+00
```

On note qu'une majorité de coefficients sont nuls. Il reste 13 variables explicatives.

8. Variation de λ

```
coef2 = predict.lars(reslasso,X,type="coefficients",mode="lambda",s=0.02)
sort(abs(coef2$coefficients), decreasing = T)[1:6]
```

```
## EG.USE.COMM.KT.OE    EG.ELC.COAL.KH    EG.IMP.CONS.ZS TM.VAL.FOOD.ZS.UN
##        0.39875294        0.10229683        0.05581962        0.02213216
## TX.VAL.FOOD.ZS.UN    AG.AGR.TRAC.NO
##        0.01189325        0.00000000
```

```
coef4 = predict.lars(reslasso,X,type="coefficients",mode="lambda",s=0.04)
sort(abs(coef4$coefficients), decreasing = T)[1:5]
```

```
## EG.USE.COMM.KT.OE    EG.ELC.COAL.KH    EG.IMP.CONS.ZS TM.VAL.FOOD.ZS.UN
##        0.38650117        0.07052334        0.03916298        0.00241451
##    AG.AGR.TRAC.NO
##        0.00000000
```

```
coef6 = predict.lars(reslasso,X,type="coefficients",mode="lambda",s=0.06)
sort(abs(coef6$coefficients), decreasing = T)[1:4]
```

```
## EG.USE.COMM.KT.OE    EG.IMP.CONS.ZS    EG.ELC.COAL.KH    AG.AGR.TRAC.NO
##      0.3702266322      0.0167739099      0.0008096456      0.0000000000
```

On observe pour $\lambda = 0.02$, $\lambda = 0.04$ et $\lambda = 0.06$ qu'il reste 5, 4 et 3 coefficients non nuls. Pour $\lambda = 0.06$:
- Quantité d'énergie primaire utilisée (EG.USE.COMM.KT.OE)
- Proportion d'énergie importée (EG.IMP.CONS.ZS)
- Production d'énergie par charbon (EG.ELC.COAL.KH)

Ces variables sont liées à l'énergie. On peut en déduire une corrélation entre elles. Et dans l'absolu il apparait plus logique que dans le modèle de Ridge que les émissions de CO2 soient liées à la production et l'utilisation de l'énergie.

9. Calculer l'erreur quadratique

```
pY = predict.lars(reslasso,X,type="fit",mode="lambda",s=0.06)
error = sum((usaIndicatorScale$EN.ATM.CO2E.KT - pY$fit)^2)/length(pY$fit)
print(error)
```

```
## [1] 0.000308241
```

L'erreur quadratique est de 3.10^-4. L'erreur du modèle de Lasso est donc plus important que la régression de Ridge.

En conclusion, le modèle de régression de Lasso met en avant plus d'informations sur les variables les plus significatives avec une méthode de régularisation des coefficients plus brutale. le modèle de régression de Ridge fait apparaitre toutes les variables dans son modèle.

Application II : agrégation de modèles

Analyse préliminaire

```
tabSpam=read.table('spam.txt',header=T,sep=';')
```

On observe **4601** observations de 58 variables dont une variable cible.
- 55 nombres réels
- 2 entiers naturels
- le label email/spam

Les variables de type nombre réels sont les fréquences des mots et caractères observés dans les emails. Il y a 2788 emails et 1813 spams (39.4%).

N.B.: Le fichier indtrain.txt n'est pas fourni. Nous prenons donc des échantillons aléatoires.

```
desc = table(tabSpam$spam)
proportionSpam = desc[2]/sum(desc)
print(proportionSpam)
```

```
##      spam
## 0.3940448
```

```
#On prend aléatoirement sans remise les données d'apprentissage (75%) et les données de test (25%).
dt = sort(sample(nrow(tabSpam), 0.75*nrow(tabSpam)))
trainData75<-tabSpam[dt,]
testData25<-tabSpam[-dt,]
```

Arbres de classification

1. Génération de l'arbre et visualisation

Import de la bibliothéque lars:

```
#install.packages("rpart")
library("rpart")
```

Génération de l'arbre.

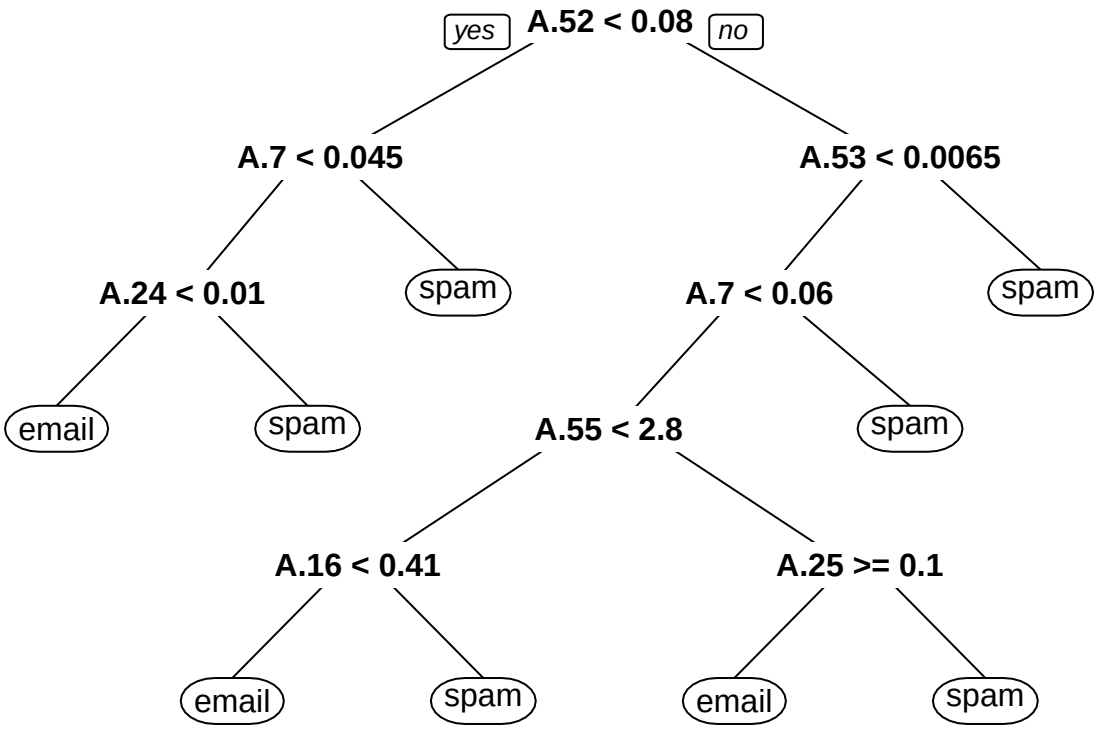
```
spamTree = rpart('spam~.', data=trainData75)
```

On utilise un package pour afficher les arbres de classification.

```
#install.packages("rpart.plot")
library("rpart.plot")
```

```
## Warning: package 'rpart.plot' was built under R version 3.3.2
```

```
prp(spamTree)
```



La variable la plus influente est **A.52** c'est la séparation qui a lieu à la racine de l'arbre. 7 variables interviennent dans notre arbre: A.7, A.16, A.17, A.25, A.52, A.53, A.55.

2. Calcul des erreurs, matrice de confusion

Avec les données d'apprentissage nous obtenons :

```
treePredTrain = predict(spamTree, trainData75, type="class")
table(treePredTrain, trainData75[,ncol(trainData75)])
```

```
##
## treePredTrain email spam
##         email  1938  197
##         spam    142 1173
```

Nous observons 5% de faux positif et 18% de faux négatif.

Avec les données de test nous obtenons :

```
treePredTest = predict(spamTree, testData25, type="class")
table(treePredTest, testData25[,ncol(testData25)])
```

```
##
## treePredTest email spam
##        email   658   60
##        spam     50  383
```

Nous observons 4% de faux positif et 21% de faux négatif.

3. Question manquante dans l'énoncé.

Question absente dans l'énoncé.

4 & 5 Bagging

Bagging génére **25** arbres de classification à l'aide déchantillons bootstrap des données d'apprentissage.

```
#install.packages("ipred")
library("ipred")
```

```
## Warning: package 'ipred' was built under R version 3.3.2
```

```
baggingPred = bagging(spam~., data=trainData75)
```

```
baggingPredTrain = predict(baggingPred, trainData75, type="class")
table(baggingPredTrain, trainData75[,ncol(trainData75)])
```

```
##
## baggingPredTrain email spam
##            email  2078    2
##            spam      2 1368
```

Nous obtenons 0.1% de faux positif et 0.3% de faux négatif.

```
baggingPredTest = predict(baggingPred, testData25, type="class")
table(baggingPredTest, testData25[, ncol(trainData75)])
```

```
##
## baggingPredTest email spam
##           email   685   37
##           spam     23  406
```

Nous obtenons 2.5% de faux positif et 11.5% de faux négatif.

Les résultats obtenus sont de très bons. Le bagging permet d'atteindre une très bonne précision.

6. & 7. & 8. Random Forest

Sur **500** arbres nous obtenons avec la méthode Random Forest:

```
#install.packages("randomForest")
library("randomForest")
```

```
## Warning: package 'randomForest' was built under R version 3.3.2
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
randomForestPredic = randomForest(spam~., data=trainData75)
```

```
randomForestPredicTrain = predict(randomForestPredic, trainData75, type="class")
table(randomForestPredicTrain, trainData75[,ncol(trainData75)])
```

```
##
## randomForestPredicTrain email spam
##                   email  2080   10
##                   spam      0 1360
```

Nous obtenons 0.05% de faux positif et 0.6% de faux négatif.

```
randomForestPredicTest = predict(randomForestPredic, testData25, type="class")
table(randomForestPredicTest, testData25[, ncol(trainData75)])
```

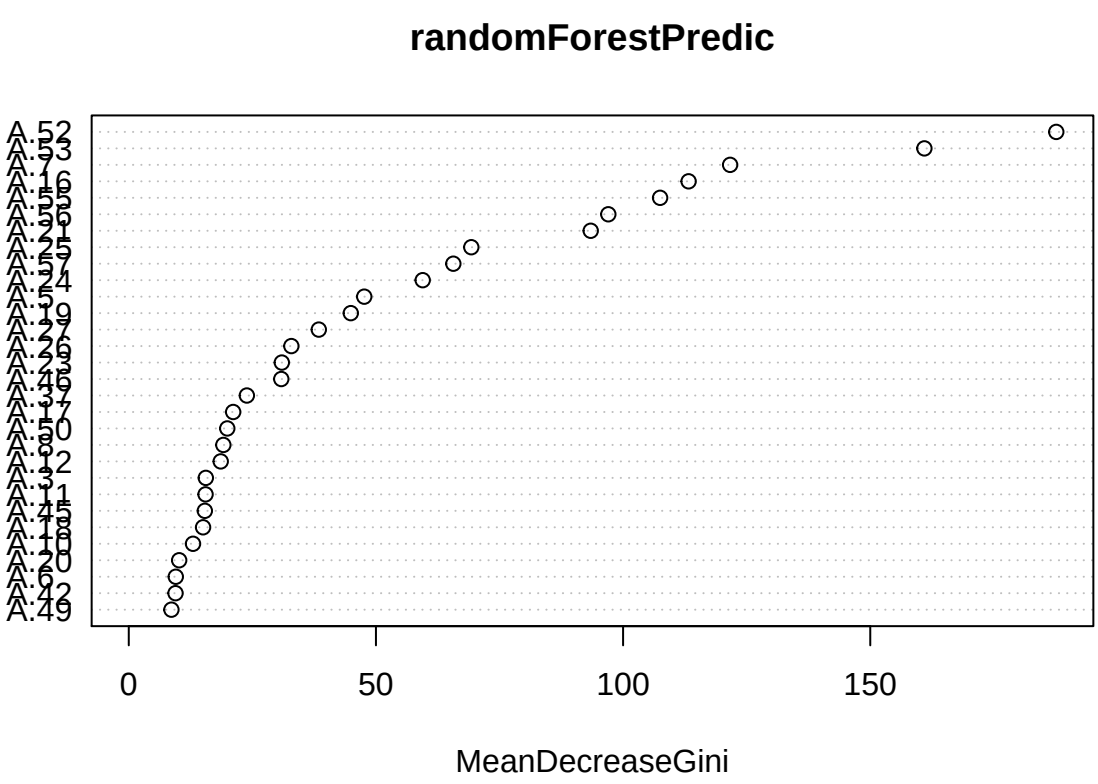
```
##
## randomForestPredicTest email spam
##                   email   691   29
##                   spam     17  414
```

Nous obtenons 2% de faux positif et 9% de faux négatif.

Il n'y a pas débat Random forest est clairement la méthode la plus efficace.

Observons l'importances des variables pour notre modèle Random Forest:

```
varImpPlot(randomForestPredic, sort=T)
```



Les variables les plus fréquentes sont A.52 et A.53.

9. & 10 & 11 & 12 Comparaisons des modèles étudiés : Scoring Classification

Comparons les modèles: -CART,
-SVM,
-Regréssion Logistique,
-Analyse disciminante linéaire,
-Bagging,
-Random Forest.

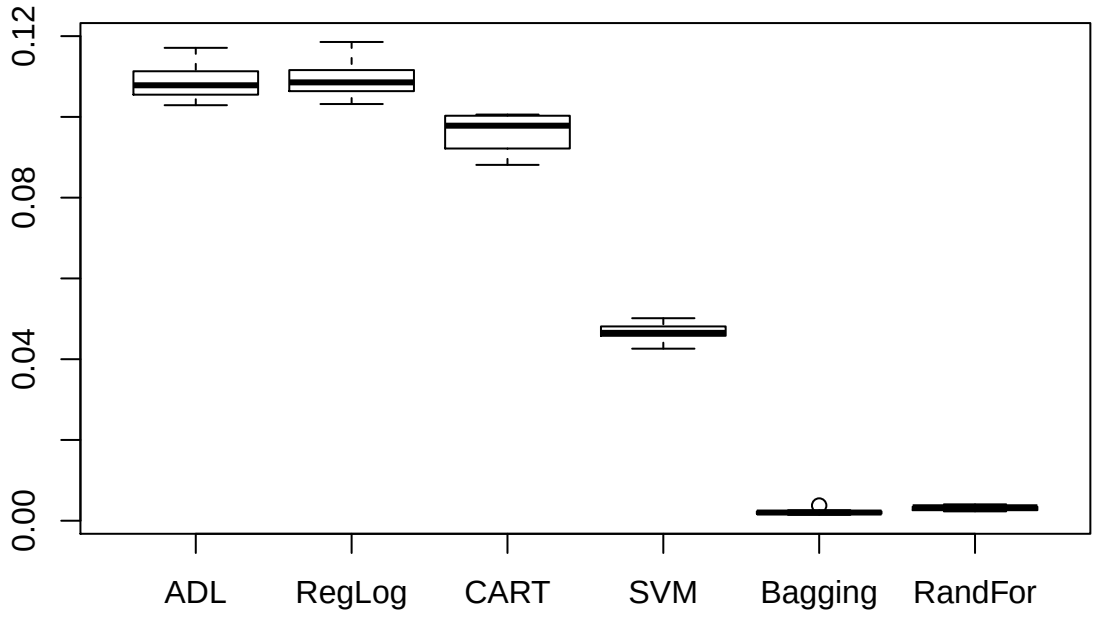
avec une cross-validation de 10 itérations pour chaque méthode.

```
library("kernlab")
library("MASS")
getError = function(modelname, data, type){
  matrix = table(predict(modelname, data, type=type), data[,ncol(data)])
  return ((matrix[1,2]+matrix[2,1])/sum(matrix))
}

K = 10

abscisse = c("ADL", "RegLog", "CART", "SVM", "Bagging", "RandFor")

crossValidationTrain = data.frame(matrix(0, nrow=K, length(abscisse)))
crossValidationTest = data.frame(matrix(0, nrow=K, length(abscisse)))
```

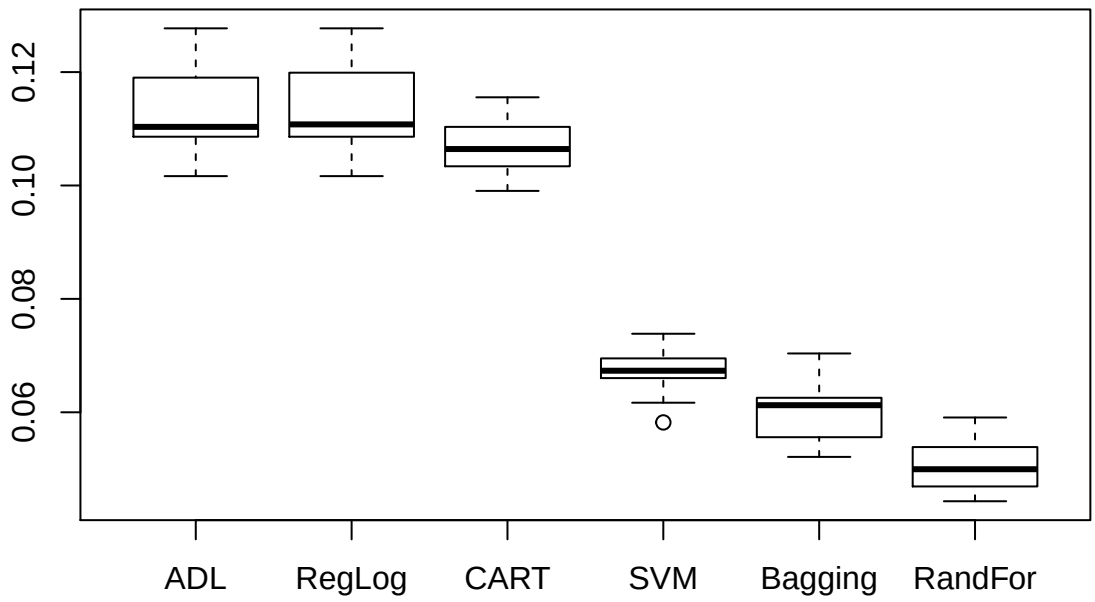
```r
names(crossValidationTrain) = abscisse
names(crossValidationTest) = abscisse

for (i in 1:K) {

#On prend aléatoirement sans remise les données d'apprentissage (75%) et les données de test (25%).
dt = sort(sample(nrow(tabSpam), 0.75*nrow(tabSpam)))
tabTrain<-tabSpam[dt,]
tabTest<-tabSpam[-dt,]

  # ADL
  adlPred = lda(spam~., data=tabTrain)
  confMatLDATrain = table(predict(adlPred, tabTrain)$class, tabTrain[, ncol(tabTrain)])
  confMatLDATest = table(predict(adlPred, tabTest)$class, tabTest[, ncol(tabTest)])

  crossValidationTrain$ADL[i] = (confMatLDATrain[1,2]+confMatLDATrain[2,1])/(confMatLDATrain[1,1]+confMa
  crossValidationTest$ADL[i] = (confMatLDATest[1,2]+confMatLDATest[2,1])/(confMatLDATest[1,1]+confMatLD

  # Regression Logistic
  tabTrainLR = tabTrain #parsing sinon erreur de glm
  tabTestLR = tabTest
  tabTrainLR[, ncol(tabTrainLR)] = as.numeric(tabTrain[, ncol(tabTrain)])
  tabTestLR[, ncol(tabTestLR)] = as.numeric(tabTest[, ncol(tabTest)] )
  lr = glm(spam~., data=tabTrainLR)

  rgPredTrain = ifelse(predict(lr, tabTrainLR)<1.5, 1, 2)
  confMatLogRegTrain = table(rgPredTrain, tabTrainLR[, ncol(tabTrainLR)])
  crossValidationTrain$RegLog[i] = (confMatLogRegTrain[1,2]+confMatLogRegTrain[2,1])/sum(confMatLogRegTr
  rgPredTest = ifelse(predict(lr, tabTestLR)<1.5, 1, 2)
  confMatLogRegTest = table(rgPredTest, tabTestLR[, ncol(tabTestLR)])
  crossValidationTest$RegLog[i] = (confMatLogRegTest[1,2]+confMatLogRegTest[2,1])/sum(confMatLogRegTest

  # CART
  tree = rpart('spam~.', data=tabTrain)
  crossValidationTrain$CART[i] = getError(tree, tabTrain, "class")
  crossValidationTest$CART[i] = getError(tree, tabTest, "class")

  # SVM
  svm = ksvm(spam~., data=tabTrain)
  crossValidationTrain$SVM[i] = getError(svm, tabTrain, "response")
  crossValidationTest$SVM[i] = getError(svm, tabTest, "response")

  # Bagging
  baggingPred = bagging(spam~., data=tabTrain)
  crossValidationTrain$Bagging[i] = getError(baggingPred, tabTrain, "class")
  crossValidationTest$Bagging[i] = getError(baggingPred, tabTest, "class")

  # Random Forest
  randomForestPred = randomForest(spam~., data=tabTrain)
  crossValidationTrain$RandFor[i] = getError(randomForestPred, tabTrain, "class")
  crossValidationTest$RandFor[i] = getError(randomForestPred, tabTest, "class")

}
```

Voici le résultat avec les boites à moustaches des erreurs de chaque méthode pour les données d'apprentissage et de test:

```
boxplot(crossValidationTrain)
```




```
boxplot(crossValidationTest)
```



En conclusion, les méthodes de Bagging et de Random Forest sont les méthodes les plus performantes. On notera que ces deux méthodes sont presques parfaite sur les données d'apprentissage et qu'elle se trompent faiblement sur les données de test. Ajoutons que la méthode de régression logistique a des performances relativement comparable que l'analyse discriminante linéaire sur les données d'apprentissage sans augmentation d'erreur sur les données de test. On peut ajouter que les données mail/spam sont adaptés aux modèles de frontières de décision, ainsi Bagging et Random Forest offrent de meilleurs résultats qu'un arbre de décision classique. Random Forest est la meilleure méthode de prédiction pour ce type de données.