# TD4 : Sélection de variables par pénalisation

## Exercice 1 : Découverte

### Importation des données

```
ozone <- read.table("Ozone.txt")
dim(ozone)
```

```
## [1] 112  13
```

### 1.

```
ozone2 <- ozone[,1:11]
model <- lm(maxO3v ~ ., data = ozone2)
summary(model)
```

```
##
## Call:
## lm(formula = maxO3v ~ ., data = ozone2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.808 -12.230  -0.793  10.892  77.775
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.82037   18.68964  -0.311  0.75612
## maxO3        0.67257    0.12017   5.597 1.88e-07 ***
## T9           4.18715    1.49849   2.794  0.00623 **
## T12         -2.57902    1.98773  -1.297  0.19742
## T15          0.32674    1.58379   0.206  0.83697
## Ne9          3.54716    1.28381   2.763  0.00681 **
## Ne12        -1.39694    1.88634  -0.741  0.46068
## Ne15        -0.64334    1.38494  -0.465  0.64327
## Vx9          1.84820    1.25438   1.473  0.14375
## Vx12        -1.04537    1.45496  -0.718  0.47412
## Vx15        -0.05731    1.26706  -0.045  0.96401
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.85 on 101 degrees of freedom
## Multiple R-squared:  0.5516, Adjusted R-squared:  0.5072
## F-statistic: 12.42 on 10 and 101 DF,  p-value: 8.32e-14
```

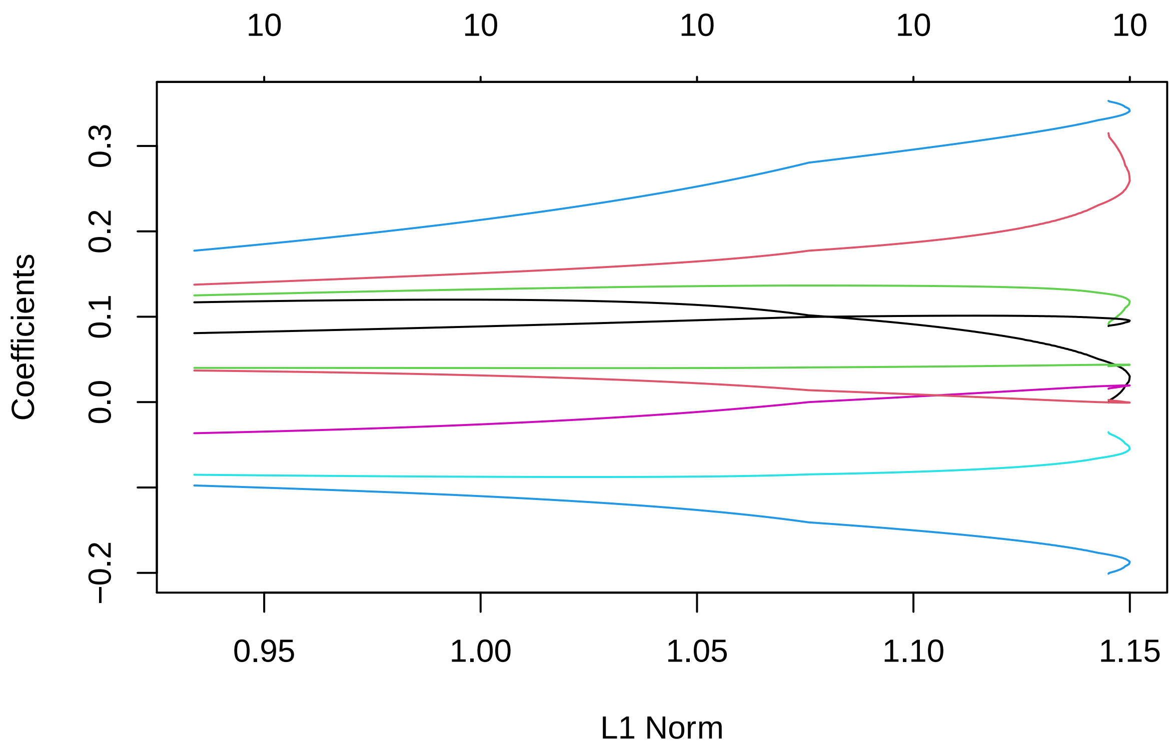
### 2.

```r
library(glmnet)
```

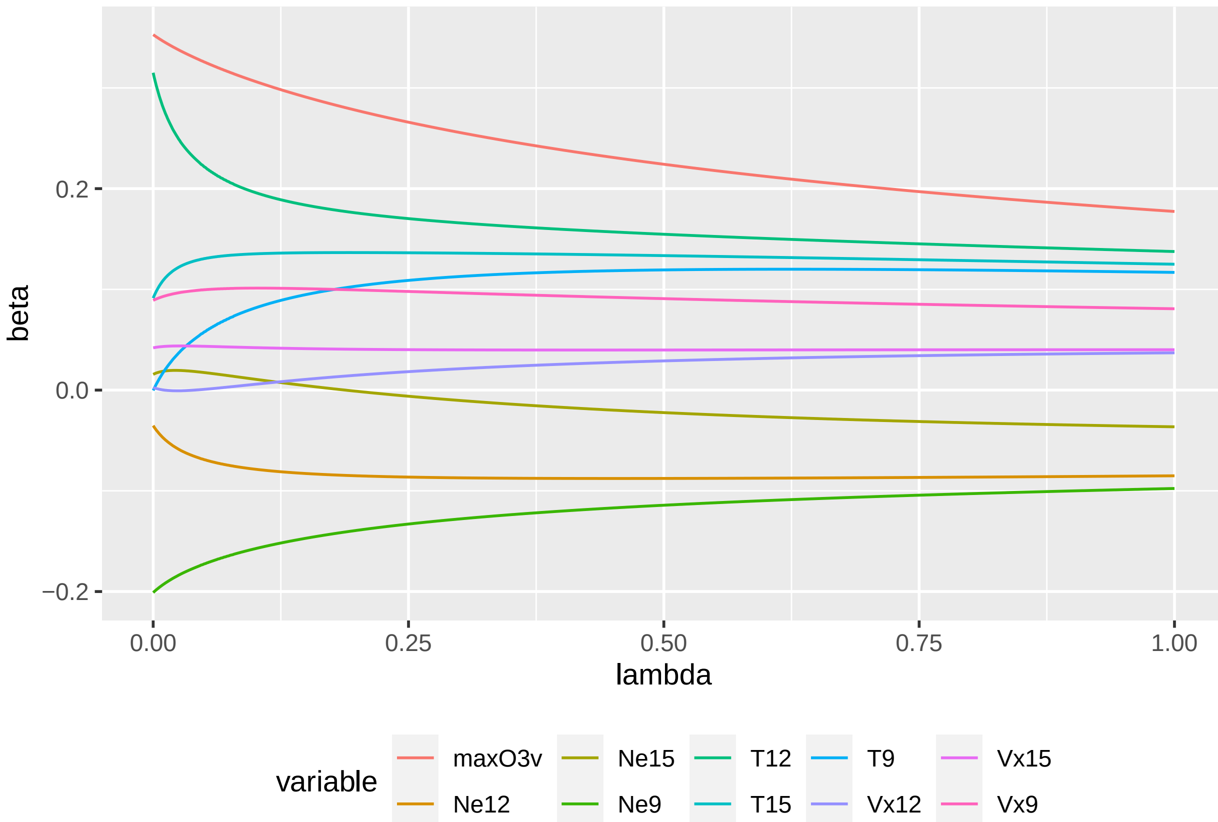
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-3
```

```r
ozone2 <- scale(ozone2)
lambda_seq <- seq(0, 1, by = 0.001)
model_ridge <- glmnet(ozone2[,2:11], ozone2[,1], alpha = 0, lambda = lambda_seq, intercept = F)
plot(model_ridge)
```



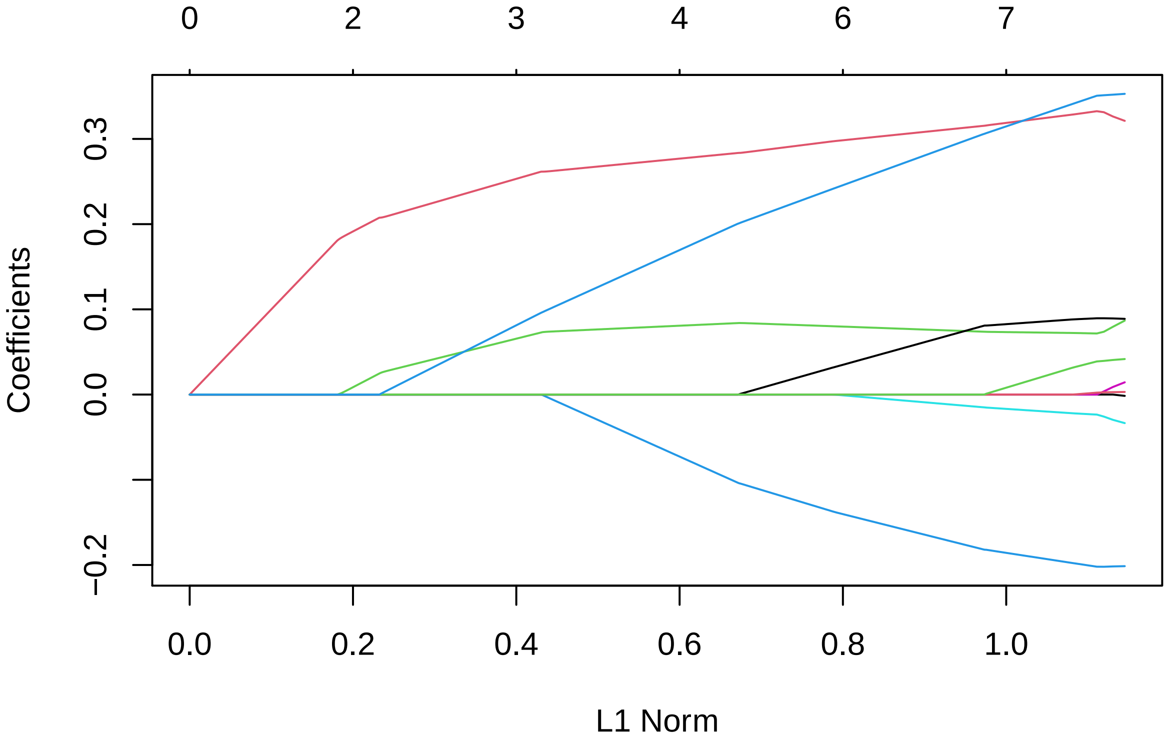

```r
library(ggplot2)

df = data.frame(lambda = rep(model_ridge$lambda, ncol(ozone2[,2:11])),
                beta = as.vector(t(model_ridge$beta)),
                variable = rep(colnames(ozone2[,2:11]), each = length(model_ridge$lambda)))
g1 = ggplot(df, aes(x = lambda, y = beta, col = variable)) + geom_line() +
  theme(legend.position = "bottom")
g1
```



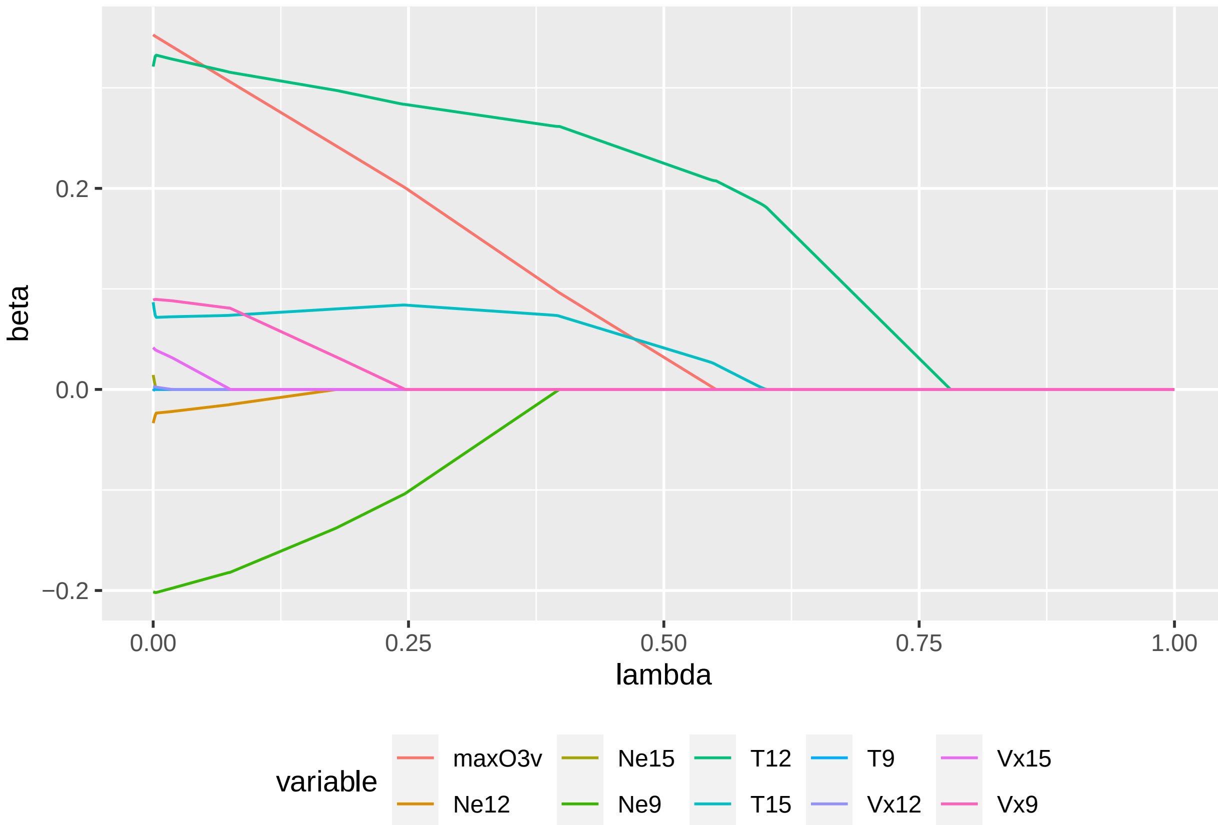

**3.**

```
model_lasso <- glmnet(ozone2[,2:11], ozone2[,1], alpha = 1, lambda = lambda_seq, intercept = F)
plot(model_lasso)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```

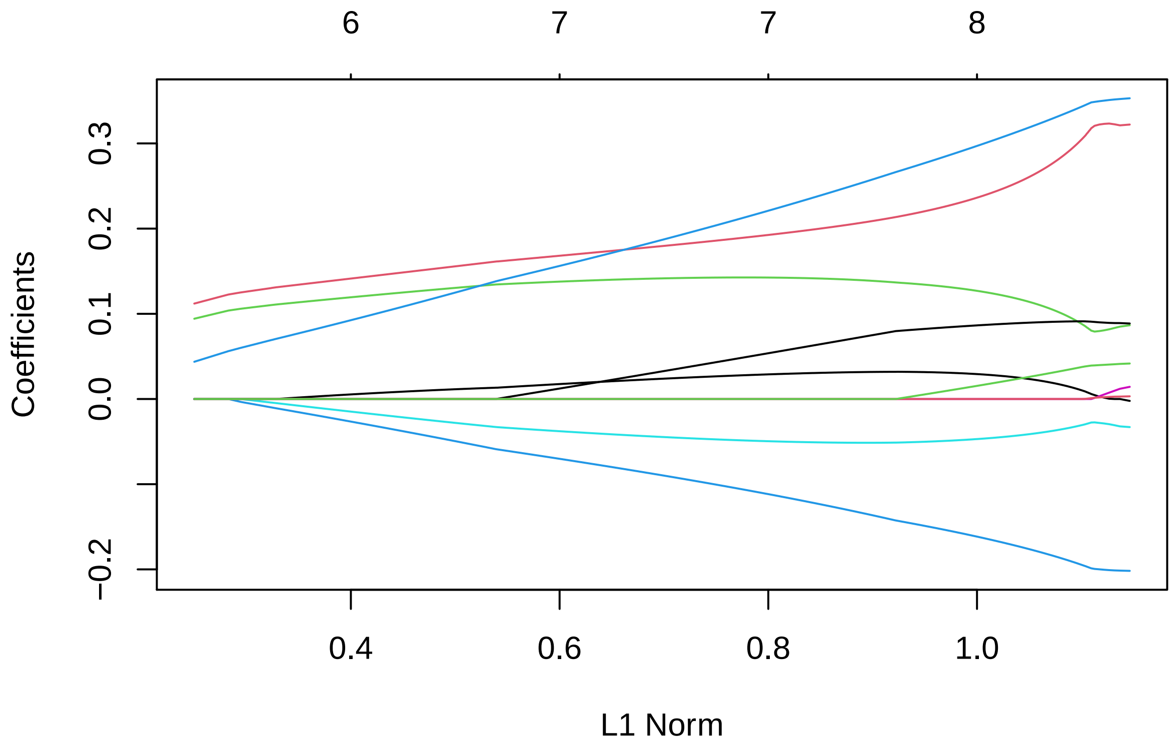



```
df = data.frame(lambda = rep(model_lasso$lambda, ncol(ozone2[,2:11])),
                beta = as.vector(t(model_lasso$beta)),
                variable = rep(colnames(ozone2[,2:11]), each = length(model_lasso$lambda)))
g2 = ggplot(df, aes(x = lambda, y = beta, col = variable)) + geom_line() +
  theme(legend.position = "bottom")
g2
```

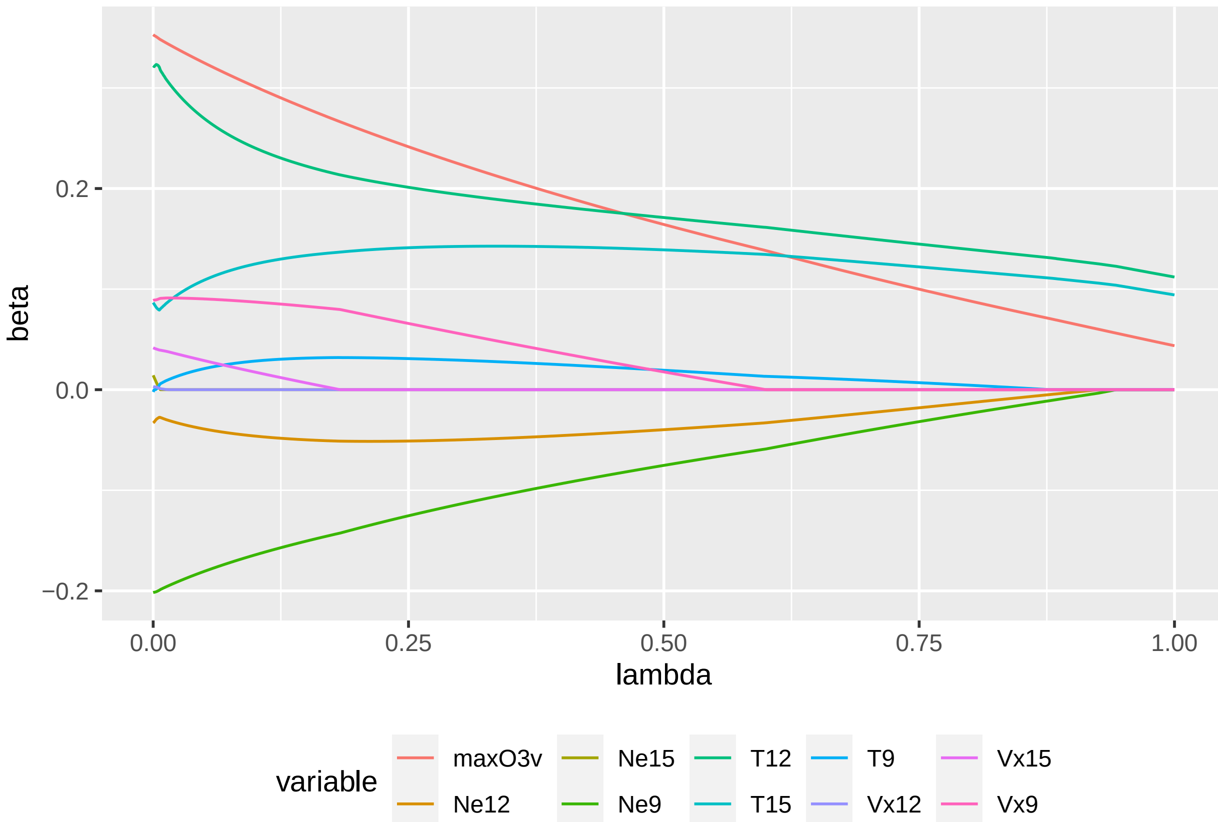



**4.**

```
model_EN <- glmnet(ozone2[,2:11], ozone2[,1], alpha = 0.5, lambda = lambda_seq, intercept = F)
plot(model_EN)
```



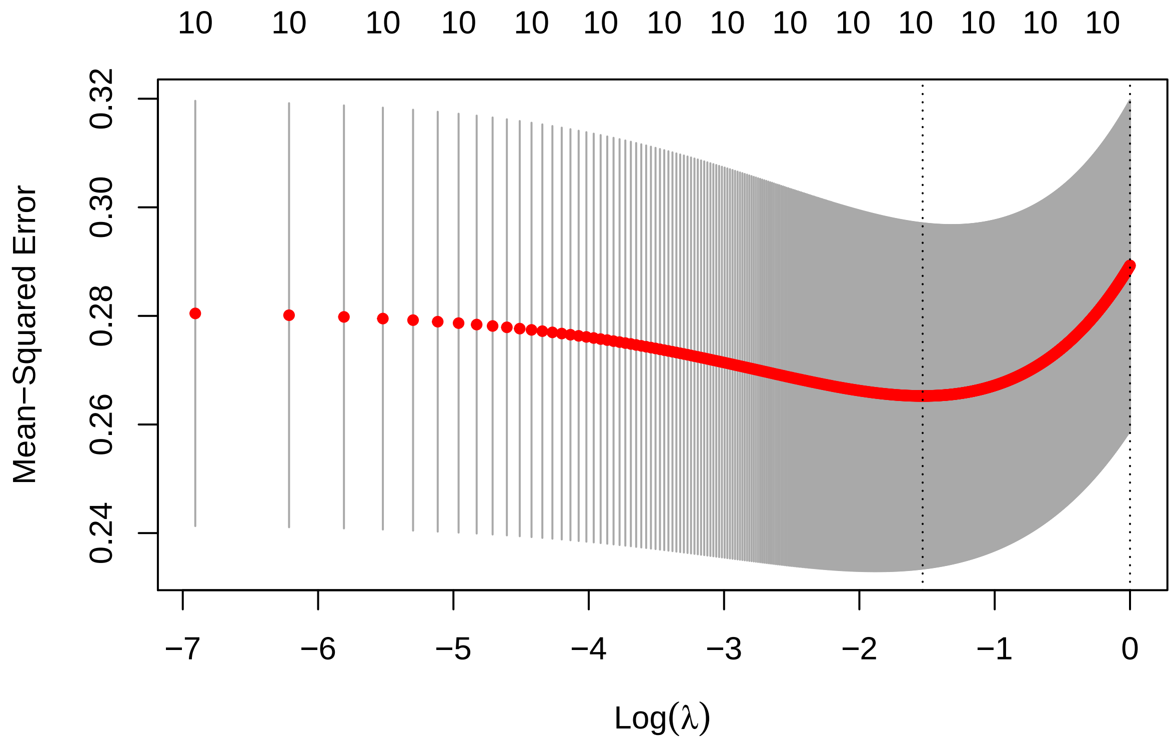

```
df = data.frame(lambda = rep(model_EN$lambda, ncol(ozone2[,2:11])),
                beta = as.vector(t(model_EN$beta)),
                variable = rep(colnames(ozone2[,2:11]), each = length(model_EN$lambda)))
g3 = ggplot(df, aes(x = lambda, y = beta, col = variable)) + geom_line() +
  theme(legend.position = "bottom")
g3
```




**5.**

ridge

```
ridge_cv <- cv.glmnet(ozone2[,2:11], ozone2[,1], alpha = 0, lambda = lambda_seq,
                      nfolds = 10, intercept = F)
best_lambda <- ridge_cv$lambda.min
plot(ridge_cv)
```

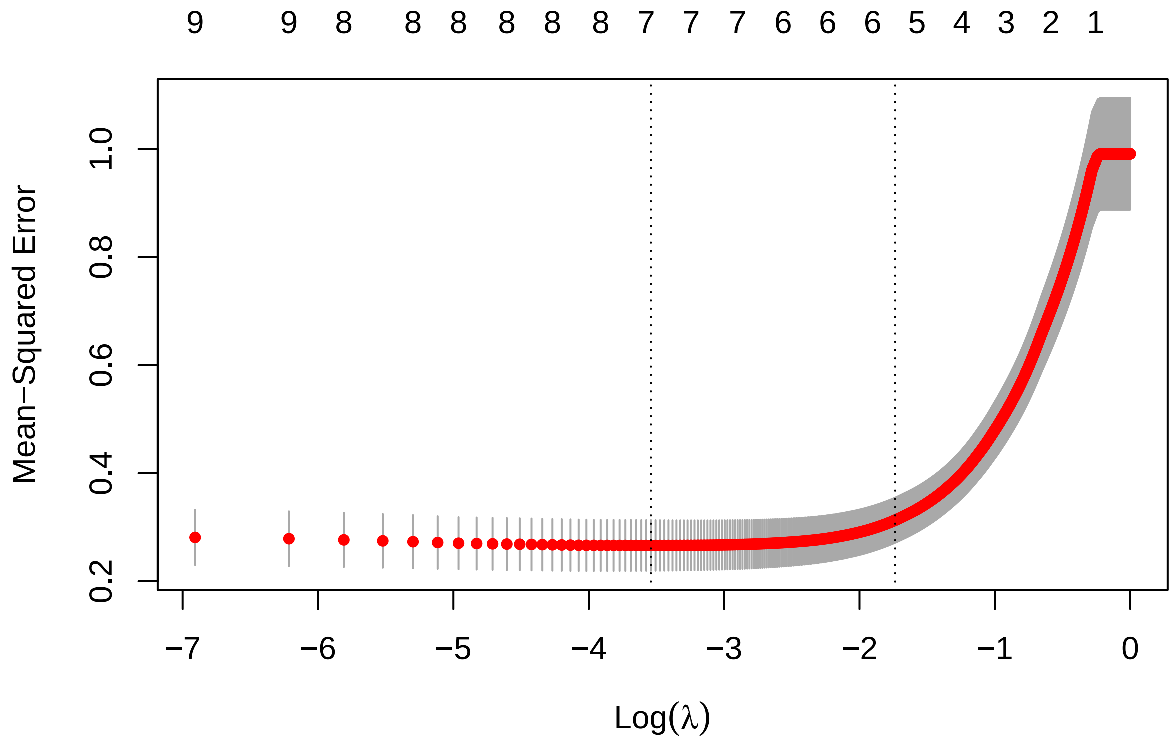



```r
ridge <- glmnet(ozone2[,2:11], ozone2[,1], alpha = 0, lambda = best_lambda, intercept = F)
ridge$beta
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                  s0
## T9       0.105451486
## T12      0.173426622
## T15      0.136551834
## Ne9     -0.137049940
## Ne12    -0.085721008
## Ne15    -0.002860282
## Vx9      0.098979917
## Vx12     0.016007823
## Vx15     0.040355949
## maxO3v   0.273700413
```

lasso

```r
lasso_cv <- cv.glmnet(ozone2[,2:11], ozone2[,1], alpha = 1, lambda = lambda_seq, nfolds = 10,
                      intercept = F)
best_lambda <- lasso_cv$lambda.min
plot(lasso_cv)
```

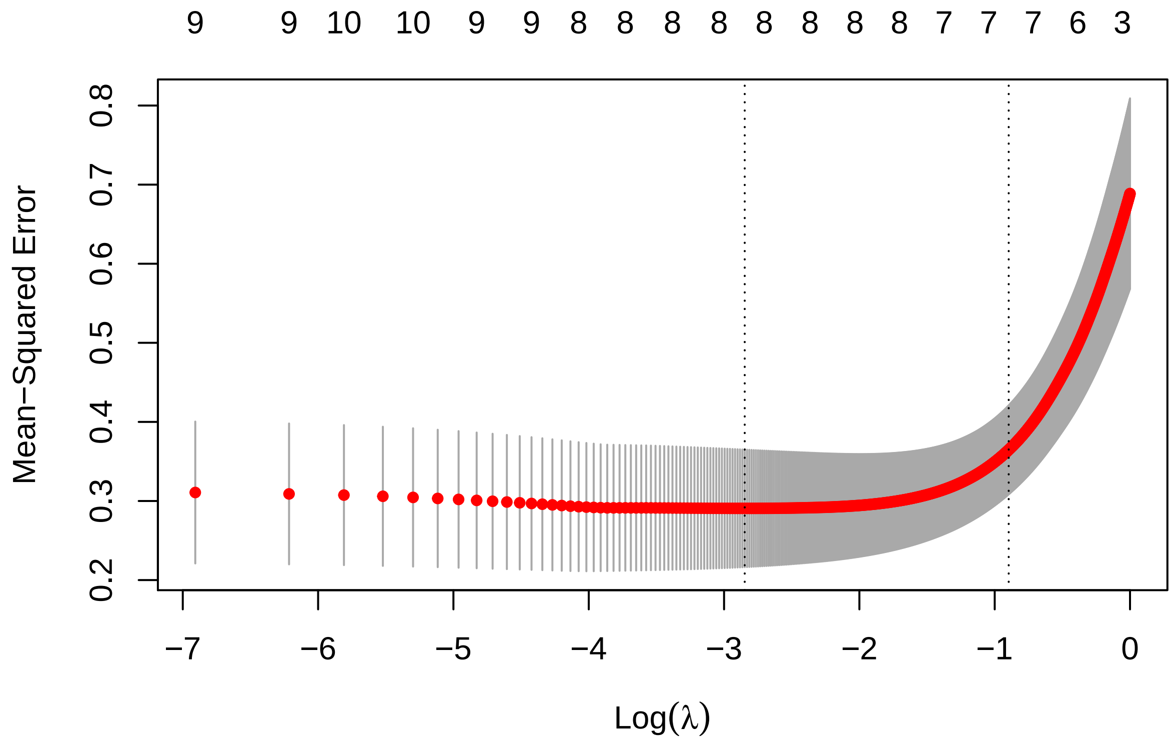



```
lasso <- glmnet(ozone2[,2:11], ozone2[,1], alpha = 1, lambda = best_lambda, intercept = F)
lasso$beta
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                  s0
## T9       .
## T12      0.32773184
## T15      0.07118473
## Ne9     -0.19473644
## Ne12    -0.02059813
## Ne15     .
## Vx9      0.08675663
## Vx12     .
## Vx15     0.02585961
## max03v   0.33449615
```

elastic net

```
en_cv <- cv.glmnet(ozone2[,2:11], ozone2[,1], alpha = 0.5, lambda = lambda_seq, nfolds = 10,
                   intercept = F)
best_lambda <- en_cv$lambda.min
plot(en_cv)
```

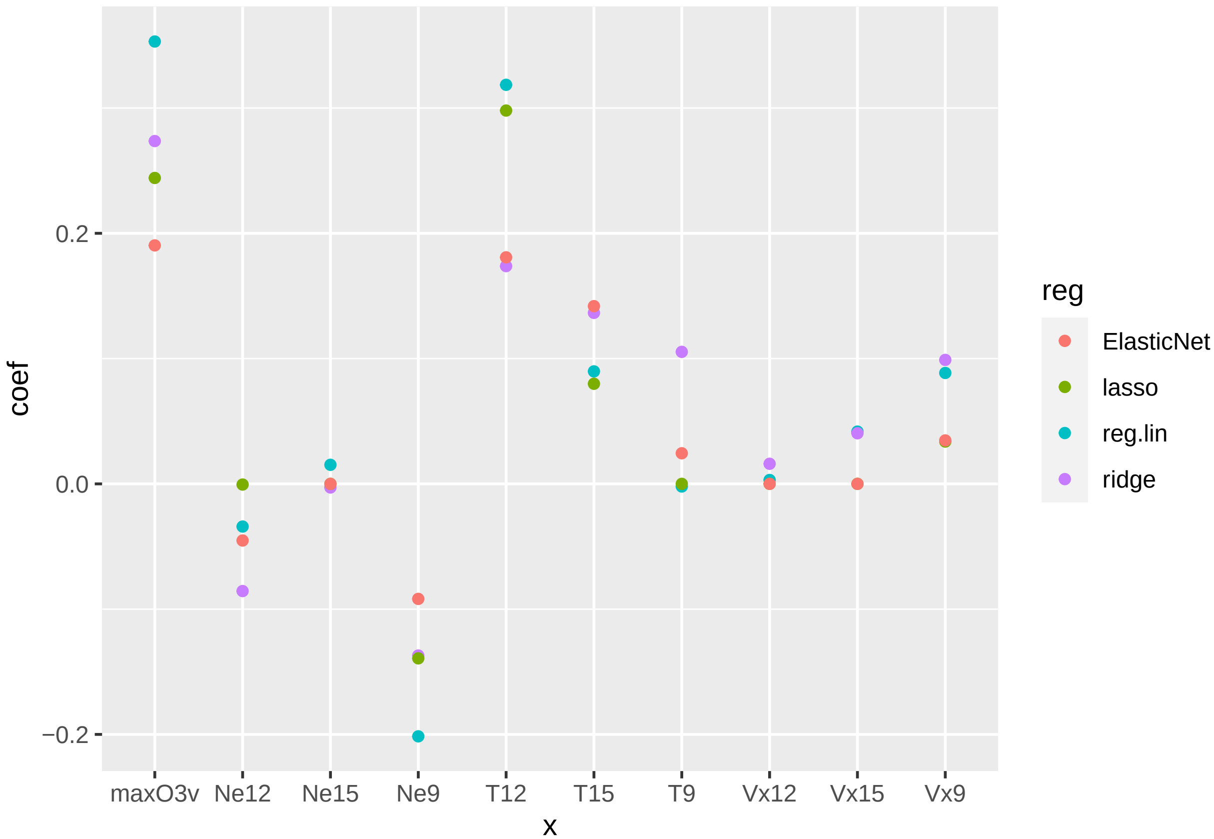



```
en <- glmnet(ozone2[,2:11], ozone2[,1], alpha = 0.5, lambda = best_lambda, intercept = F)
en$beta
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                 s0
## T9      0.02226732
## T12     0.26548393
## T15     0.11112647
## Ne9    -0.17770222
## Ne12   -0.04037481
## Ne15    .
## Vx9     0.08980734
## Vx12    .
## Vx15    0.02705269
## maxO3v  0.32087457
```

graphique qui résume

```
regusuel <- lm(ozone2[,1]~ozone2[,2:11]-1) # modèle linéaire sans intercept
df4 <- data.frame(x=rep(colnames(ozone2[,2:11]),4),
                  coef=c(as.vector(regusuel$coefficients),
                         as.vector(coef(ridge_cv,s=ridge_cv$lambda.min)[-1]),
                         as.vector(coef(lasso_cv)[-1]),as.vector(coef(en_cv)[-1])),
                  reg=c(rep("reg.lin",ncol(ozone2[,2:11])),
                        rep("ridge",ncol(ozone2[,2:11])),
                        rep("lasso",ncol(ozone2[,2:11])),
                        rep("ElasticNet",ncol(ozone2[,2:11]))))
g4 <- ggplot(df4)+ geom_point(aes(x=x,y=coef,col=reg))
g4
```




## Exercice 2 : Comparaison des méthodes

### Jeu de données 1 : petit signal et beaucoup de bruit

```
p <- 5000
n <- 1000
real_p <- 15
x <- matrix(rnorm(n*p), nrow=n, ncol=p)
y <- apply(x[,1:real_p], 1, sum) + rnorm(n)

train_rows <- sample(1:n, .66*n)
x.train1 <- x[train_rows,]
x.test1 <- x[-train_rows,]
y.train1 <- y[train_rows]
y.test1 <- y[-train_rows]
```

### Jeu de données 2 : gros signal et beaucoup de bruit

```
p <- 5000
n <- 1000
real_p <- 1000
x <- matrix(rnorm(n*p), nrow=n, ncol=p)
y <- apply(x[,1:real_p], 1, sum) + rnorm(n)

train_rows <- sample(1:n, .66*n)
```

```r
x.train2 <- x[train_rows,]
x.test2 <- x[-train_rows,]
y.train2 <- y[train_rows]
y.test2 <- y[-train_rows]
```

### Jeu de données 3 : signal varié et variables corrélées

```r
library(MASS)
p <- 50
n <- 100
CovMatrix <- outer(1:p, 1:p, function(x,y) {.7^abs(x-y)})
x <- mvrnorm(n, rep(0,p), CovMatrix)
y <- 10 * apply(x[, 1:2], 1, sum) + 5 * apply(x[, 3:4], 1, sum) +
  apply(x[, 4:14], 1, sum) + rnorm(n)

train_rows <- sample(1:n, .66*n)
x.train3 <- x[train_rows,]
x.test3 <- x[-train_rows,]
y.train3 <- y[train_rows]
y.test3 <- y[-train_rows]
```

#### 1.

Jeu 1

```r
fit.lasso <- glmnet(x.train1, y.train1, family="gaussian", alpha=1)
fit.ridge <- glmnet(x.train1, y.train1, family="gaussian", alpha=0)
fit.elnet <- glmnet(x.train1, y.train1, family="gaussian", alpha=.5)
```

Jeu 2

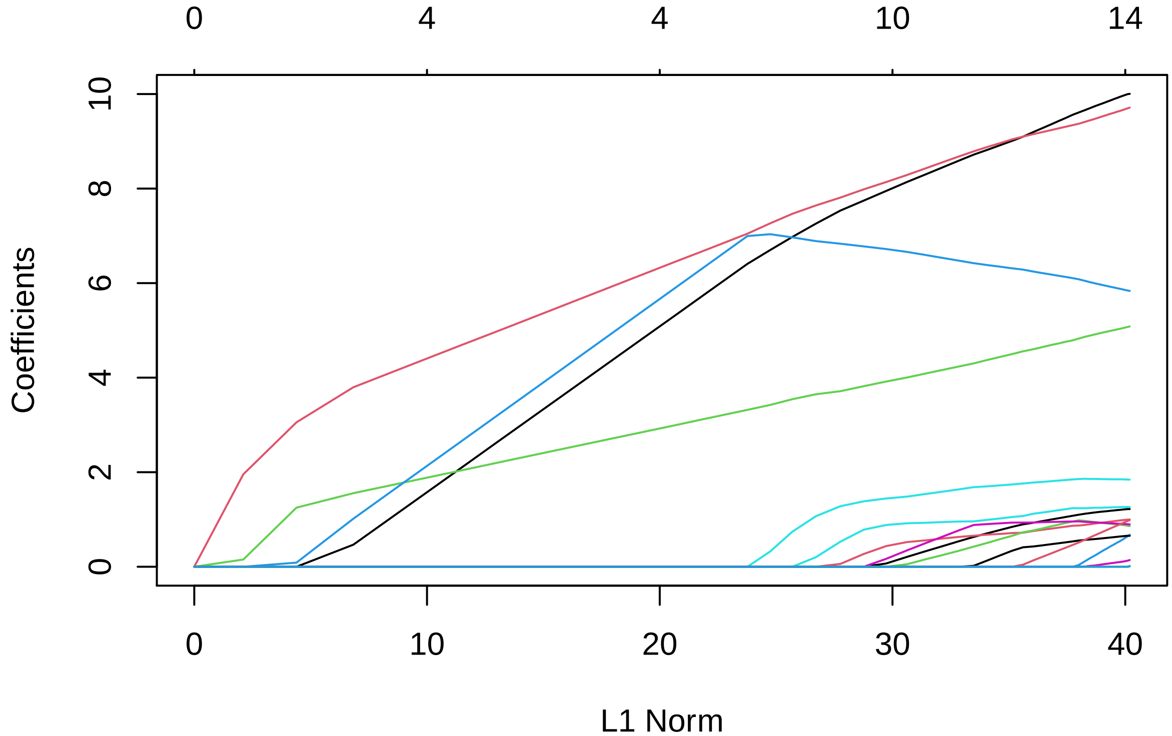
```r
fit.lasso <- glmnet(x.train2, y.train2, family="gaussian", alpha=1)
fit.ridge <- glmnet(x.train2, y.train2, family="gaussian", alpha=0)
fit.elnet <- glmnet(x.train2, y.train2, family="gaussian", alpha=.5)
```

Jeu 3

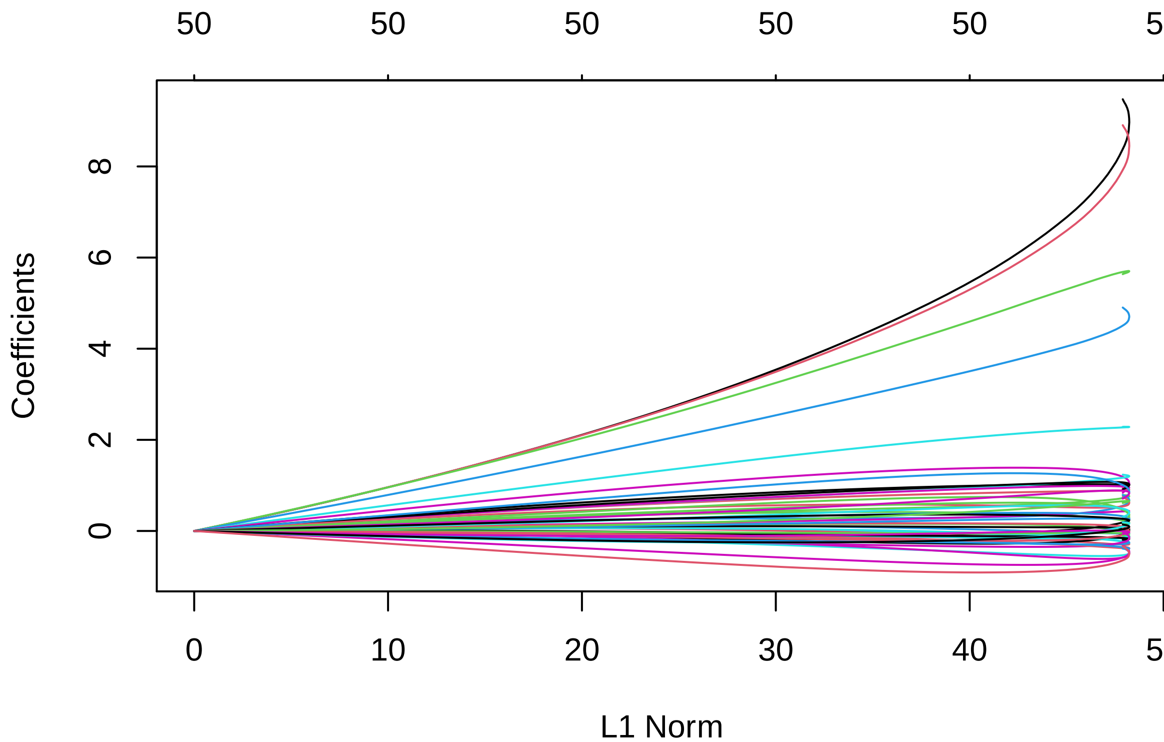
```r
fit.lasso <- glmnet(x.train3, y.train3, family="gaussian", alpha=1)
fit.ridge <- glmnet(x.train3, y.train3, family="gaussian", alpha=0)
fit.elnet <- glmnet(x.train3, y.train3, family="gaussian", alpha=.5)
```

#### 2.

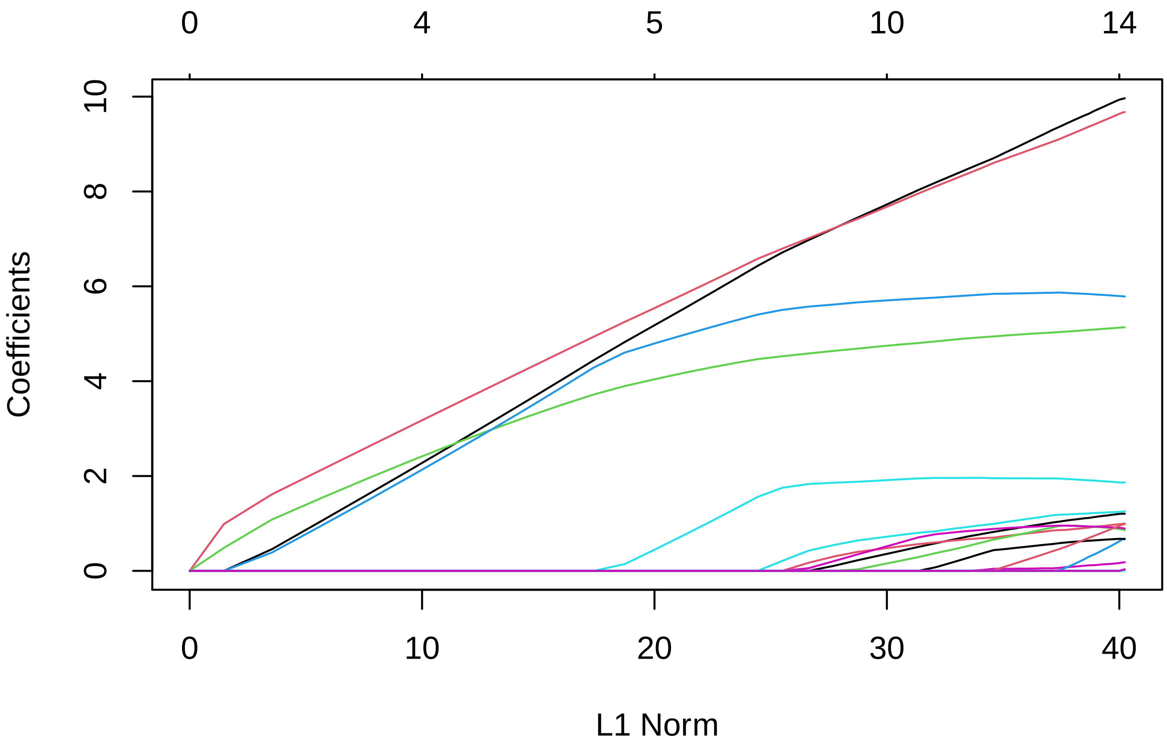
```r
plot(fit.lasso)
```




```
plot(fit.ridge)
```




```
plot(fit.elnet)
```




**3.**

Jeu 1 :

```
lasso <- cv.glmnet(x.train1, y.train1, type.measure="mse",alpha=1,family="gaussian")
EN <- cv.glmnet(x.train1, y.train1, type.measure="mse",alpha=0.5,family="gaussian")
ridge <- cv.glmnet(x.train1, y.train1, type.measure="mse",alpha=0,family="gaussian")

lasso.model <- glmnet(x.train1, y.train1, alpha=1, family="gaussian", lambda=lasso$lambda.min)
EN.model <- glmnet(x.train1, y.train1, alpha=0.5, family="gaussian", lambda=EN$lambda.min)
ridge.model <- glmnet(x.train1, y.train1, alpha=0, family="gaussian", lambda=ridge$lambda.min)

yhat_lasso <- predict(lasso.model, s=lasso.model$lambda.min, newx=x.test1)
yhat_EN <- predict(EN.model, s=lasso.model$lambda.min, newx=x.test1)
yhat_ridge <- predict(ridge.model, s=lasso.model$lambda.min, newx=x.test1)

mse_lasso <- mean((y.test1 - yhat_lasso)^2)
mse_EN <- mean((y.test1 - yhat_EN)^2)
mse_ridge <- mean((y.test1 - yhat_ridge)^2)

c(mse_lasso,mse_EN,mse_ridge)
```

```
## [1]  1.179451  1.244971 13.114988
```

Lasso est le meilleur (normal, il est fait pour ça)

**4.**

Jeu 2: ridge est le meilleur (bon en prédiction mais pas vraiment interprétable) Jeu 3: EN est le meilleur

# Exercice 3 : Comparaison pls et Lasso

## Importation des données

```
library(plsgenomics)
```

```
## For any news related to the 'plsgenomics' package (update, corrected bugs), please check http://thotl
```

```
## C++ based sparse PLS routines will soon be available on the CRAN in the new 'fastPLS' package.
```

```
data(Colon)
length(Colon)
```

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

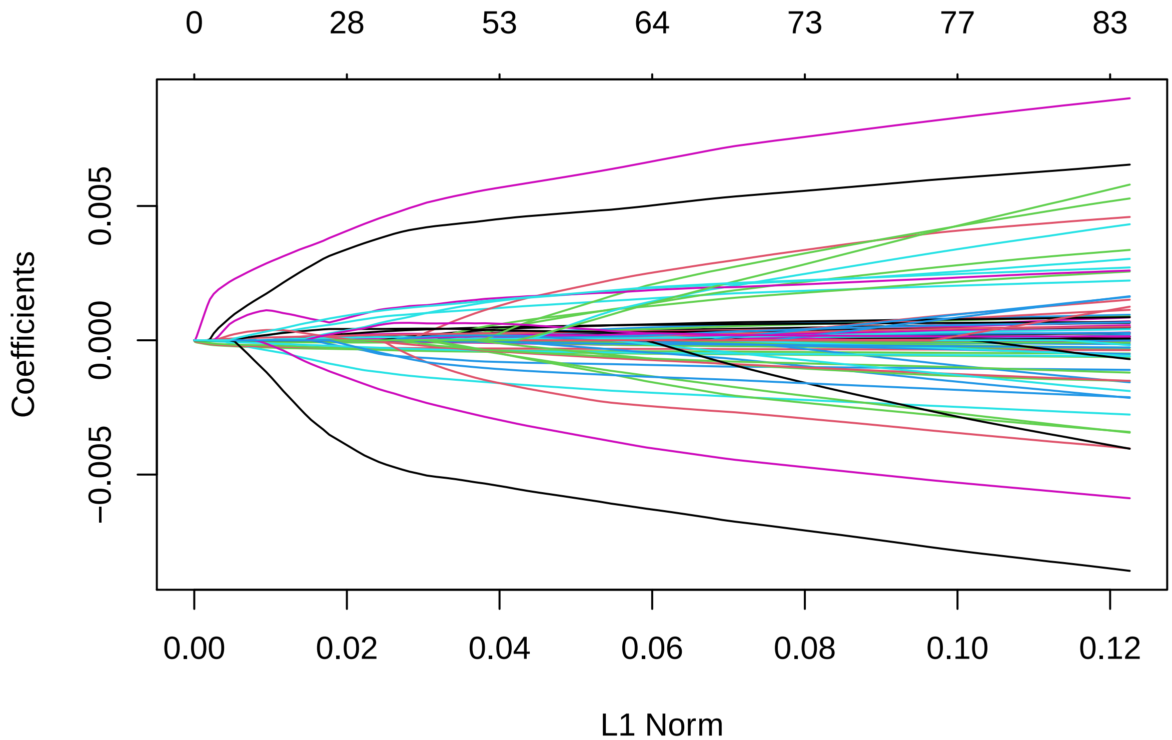
```
X <- Colon$X
Y <- Colon$Y
Y <- Y-1
gene <- Colon$gene.names
Colon <- data.frame(X=I(X),Y=Y)
```

### 1.

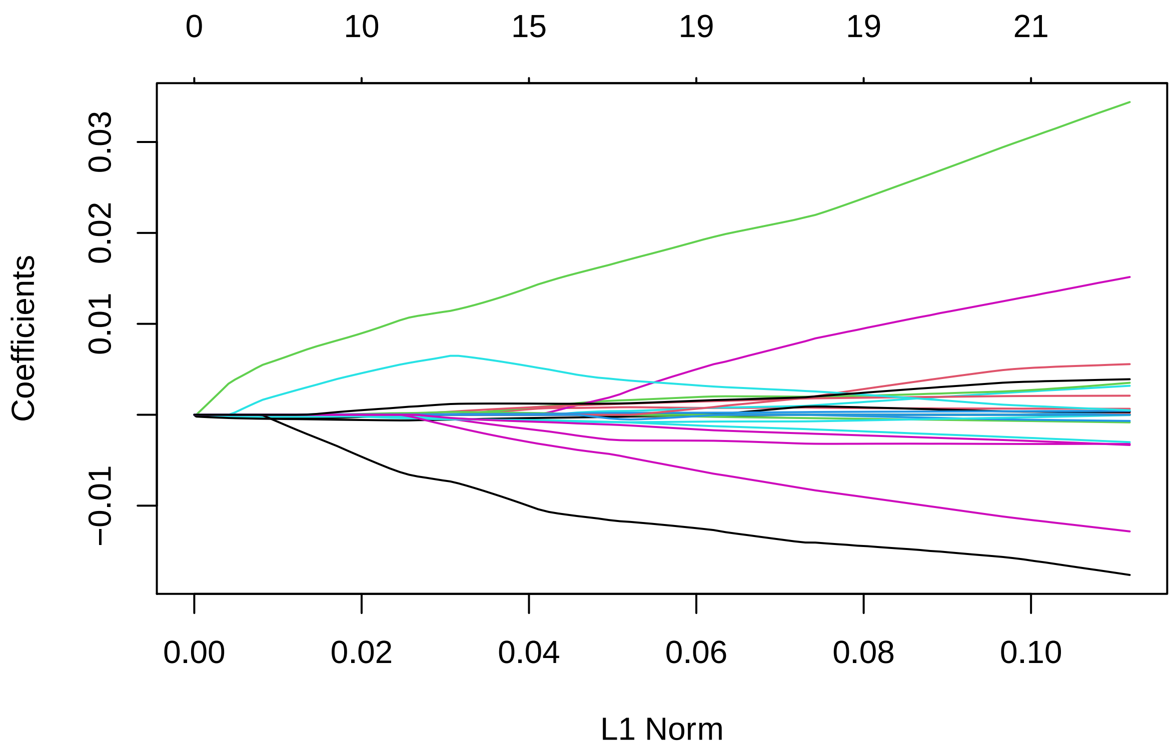
```
train <- rbinom(length(Colon$Y),1,2/3)
Colon.train <- c()
Colon.test <- c()
Colon.train$X <- Colon$X[train==1,]
Colon.train$Y <- Colon$Y[train==1]
Colon.test$X <- Colon$X[train==0,]
Colon.test$Y <- Colon$Y[train==0]
```

### 2.

```
colon.glm <- glmnet(Colon.train$X, Colon.train$Y, family="binomial", alpha = 0.5)
plot(colon.glm) # elastic net
```




```
colon.glm <- glmnet(Colon.train$X,Colon.train$Y,family="binomial",alpha=1)
plot(colon.glm) # ici, il s'agit du lasso
```




**3.**

```r
colon.cv <- cv.glmnet(Colon.train$X,Colon.train$Y,nfolds=5)
colon.glmnet.model <- glmnet(Colon.train$X,Colon.train$Y,family="binomial",nlambda=1,lambda=colon.cv$la
length(which(abs(colon.glmnet.model$beta)>0))
```

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

**4.**

```r
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
plscolon <- plsr(Colon.train$Y~Colon.train$X,ncomp=30,scale=TRUE,validation="CV",segments=5)
summary(plscolon)
```

```
## Data:    X dimension: 42 2000
##  Y dimension: 42 1
## Fit method: kernelpls
## Number of components considered: 30
##
## VALIDATION: RMSEP
## Cross-validated using 5 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.4975   0.4663   0.4460   0.4496   0.4850   0.5364   0.5642
## adjCV       0.4975   0.4739   0.4341   0.4344   0.4557   0.4855   0.5066
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.5725   0.5828   0.6013    0.6166    0.6324    0.6415    0.6452
## adjCV   0.5137   0.5230   0.5390    0.5521    0.5659    0.5737    0.5767
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV       0.6458    0.6473    0.6468    0.6464    0.6460    0.6459    0.6459
## adjCV    0.5771    0.5784    0.5779    0.5775    0.5772    0.5771    0.5771
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV       0.6459    0.6459    0.6459    0.6459    0.6459    0.6459    0.6459
## adjCV    0.5771    0.5771    0.5771    0.5771    0.5771    0.5771    0.5771
##        28 comps  29 comps  30 comps
## CV       0.6459    0.6459    0.6459
## adjCV    0.5771    0.5771    0.5771
##
## TRAINING: % variance explained
##               1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X               36.50    54.44    61.50    66.26    68.03    69.96    72.85
## Colon.train$Y   23.12    52.47    65.82    78.72    91.91    95.67    97.33
##               8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X               77.12    79.66     80.97     82.66     84.41     85.73
## Colon.train$Y   97.97    98.53     99.06     99.40     99.66     99.83
##               14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## X                86.77     87.98     88.72     89.51     90.18     90.82
## Colon.train$Y    99.93     99.96     99.99    100.00    100.00    100.00
##               20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
```

```
## X                  91.56     92.29     92.88     93.54     94.27     94.59
## Colon.train$Y     100.00    100.00    100.00    100.00    100.00    100.00
##                26 comps  27 comps  28 comps  29 comps  30 comps
## X                  95.1     95.56     95.94     96.31     96.64
## Colon.train$Y     100.0    100.00    100.00    100.00    100.00
```

```r
ncomponents=3 # 75% de la variance des Y expliquée
pls.reduction.matrix <- plscolon$loadings[,1:ncomponents]
Colon.train$pls.reducedX <- Colon.train$X %*% pls.reduction.matrix
Colon.test$pls.reducedX <- Colon.test$X %*% pls.reduction.matrix
pls.model <- glm(Y~pls.reducedX,data=Colon.train,family=binomial(link="logit"))
test.prediction <- predict(pls.model,newdata = Colon.test,type="response")
rbind(test.prediction,Colon.test$Y) # 4 erreurs uniquement
```

```
##                          1          2         3         4        5        6
## test.prediction 0.3440091 0.09131012 0.2816004 0.6894953 0.885135 0.754415
##                 1.0000000 0.00000000 0.0000000 0.0000000 1.000000 1.000000
##                         7        8         9        10        11        12
## test.prediction 0.9997971 0.996201 0.9888173 0.9680977 0.9681195 0.1395826
##                 1.0000000 1.000000 1.0000000 1.0000000 1.0000000 0.0000000
##                        13        14       15        16         17        18
## test.prediction 0.9994825 0.9992456 0.949932 0.2549138 0.06755999 0.9889428
##                 1.0000000 1.0000000 1.000000 0.0000000 1.00000000 1.0000000
##                         19        20
## test.prediction 0.09817902 0.7914988
##                 0.00000000 1.0000000
```

**5.**

```r
colonpredict.glmnet <- predict.glmnet(colon.glmnet.model,Colon.test$X,type="response")
sum((1/length(Colon.test$Y))*(colonpredict.glmnet-Colon.test$Y)^2)
```

```
## [1] 0.8182012
```

```r
sum((1/length(Colon.test$Y))*(test.prediction-Colon.test$Y)^2)
```

```
## [1] 0.1039347
```