Rapport TP2

Léos Coutrot - Colin Coërchon

06-11-2023

Contents

IV. Cookies Study

On nous donne dans cette partie un fichier `Cookies.csv` :

```
data_cookies<-read.table(file = "cookies.csv", header=TRUE,sep=",")
Y<-data_cookies$fat

head(data_cookies[, 1:8])
```

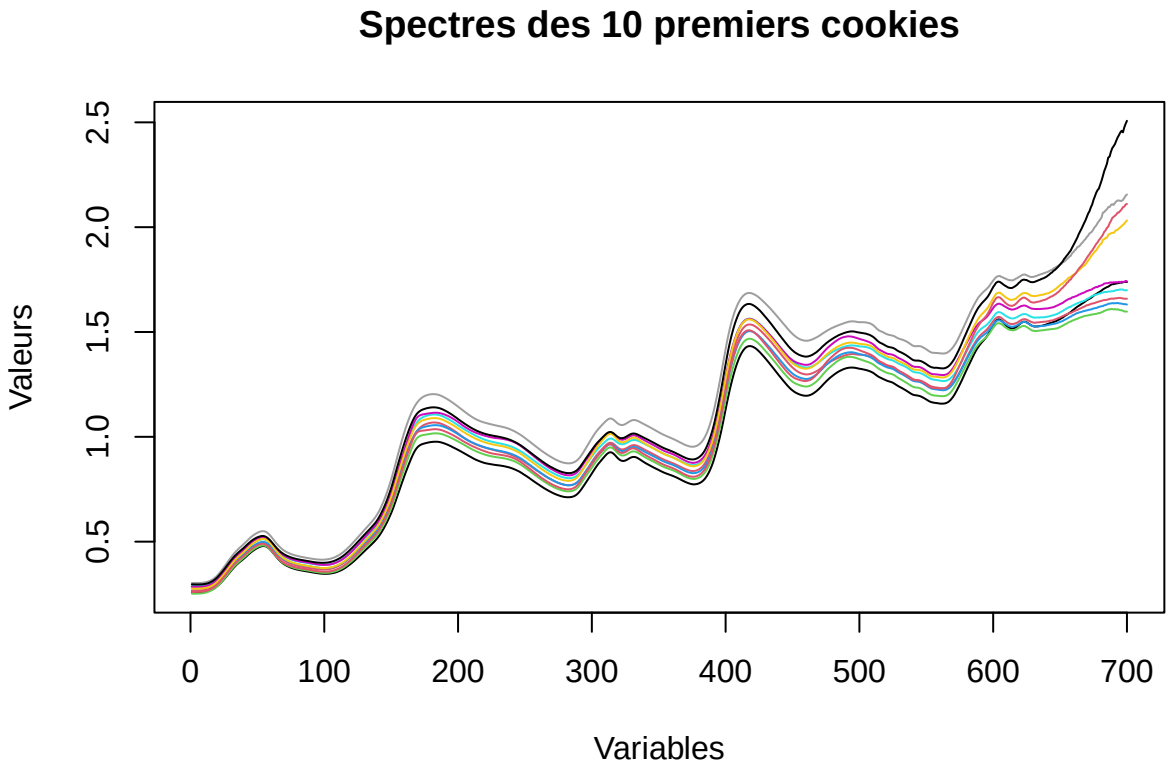
```
##     fat    X1100    X1102    X1104    X1106    X1108    X1110    X1112
## 1 12.57 0.259748 0.259482 0.259535 0.259270 0.259376 0.259270 0.259642
## 2 15.13 0.267625 0.267320 0.267117 0.266864 0.266965 0.267016 0.267422
## 3 12.63 0.251753 0.251654 0.251851 0.251900 0.252047 0.252342 0.252735
## 4 13.85 0.278077 0.277877 0.277827 0.278077 0.277777 0.278027 0.278177
## 5 15.25 0.288900 0.288484 0.288328 0.288328 0.288328 0.288796 0.289316
## 6 13.66 0.285423 0.284891 0.284625 0.284625 0.284678 0.284731 0.284785
```

Le fichier `Cookies.csv` comprend exactement 700 variables allant de “X1100” à “X2498”. Et nous avons seulement 32 observations (cookies) dans ce jeu de données. L’objectif est d’ici d’effectuer une régression linéaire de la variable cible “Y” (`fat`) de notre jeu de données.

Et voilà en qualité d’exemple le spectre des valeurs pour les 5 premiers cookies pour les 700 variables de notre problème.

```
# On sélectionne les 10 premiers cookies
spectres <- as.matrix(data_cookies[1:10, -1])

# Le premier spectre
plot(spectres[1,], type='l', ylim=range(spectres), main="Spectres des 10 premiers cookies",
```

```r
     xlab="Variables", ylab="Valeurs", col=1)

# Le tracé des 9 autres spectres
for (i in 2:10) {
    lines(spectres[i,], col=i)
}
```



Cette régression linéaire pourrait prendre en compte les 700 variables du problèmes et prendrait alors la forme :

$$Y = \beta_1 + \sum_{j=2}^{p+1} \beta_j X_j + \varepsilon$$

Avec :

- Y la valeur cible.
- $X = (X_1, \ldots, X^{p+1})$ les valeurs des différentes variables (avec $p = 700$ ici)
- Le "Data Set" : $D_n = \{(x_i, y_i) \,|\, i \in [\![1, n]\!],\, y_i \in \mathbb{R}, x_i \in \mathbb{R}^p\}$ (avec $n = 32$ ici).

Dans notre étude, nous avons clairement $p \gg n$. Quand le nombre de variables prédictives p est beaucoup plus grand que le nombre d'observations n, cela peut entraîner un **surajustement du modèle**, où le modèle peut parfaitement s'adapter aux données d'entraînement mais se comporte mal sur de nouvelles données. Il y a donc différentes méthodes connues pour pallier à se problème dont nous avons abordé quelques unes dans le cours n°2.

Les premières idées

Premièrement, la première idée qui nous vient naturellement serait de mettre de côté les variables non-significatives pour notre régression linéaire. Cette idée est très intéressantes, et nous avons ainsi vu dans le cours des méthodes de **sélections de variables**.

Pour cela, on définit :

$$Y = X_{\mathcal{M}}\beta + \varepsilon$$

Où $\mathcal{M} \subset [\![1, p]\!]$ et $X_{\mathcal{M}} = [X_{i,j_k}]_{i\in[\![1,n]\!];\, j_k\in\mathcal{M}}$.

On définit ainsi notre nouvelle "erreur quadratique" $\mathrm{RSS}(\mathcal{M})$ s'écrivant :

$$\begin{aligned} \mathrm{RSS}(\mathcal{M}) &\triangleq \left\|Y - X_{\mathcal{M}}(X_{\mathcal{M}}^T X_{\mathcal{M}})^{-1} X_{\mathcal{M}}^T Y\right\|^2 \\ \text{Avec :} \quad \widehat{\mathcal{M}} &\triangleq \underset{\mathcal{M}\subset[\![1,p]\!]}{\operatorname{argmin}}\ \mathrm{RSS}(\mathcal{M}) + \text{"penalty"} \end{aligned}$$

Pour tenter de nous approcher de ce sous-ensemble souhaité ($\widehat{\mathcal{M}}$), nous avons en effet découvert les algorithmes gloutons suivant en cours :

- "**Forward selection**" : On commence avec aucune variable et ajoute la variable la plus significative à chaque étape dans notre nouveau "Data Set" $\mathcal{M}$.
- "**Backward elimination**" : On commence avec toutes les variables et on élimine la moins significative à chaque étape.
- "**Stepwise regression**" : Une sorte d'entre d'eux des deux méthodes précédentes.

Ces méthodes sont dites incrémentales car elles ajoutent ou retirent une variable à la fois et réévaluent le modèle à chaque étape. Cependant, elles peuvent être assez gourmandes en temps de calcul et ne sont pas toujours pratiques lorsque p est très grand (ce qui est clairement le cas ici !).

Il est donc nécessaire de trouver une autre solution.

Nous avons ensuite découvert deux nouvelles méthodes dans le cours qui tendent elle aussi à s'approcher du meilleur $\widehat{\mathcal{M}}$ possible : la **régression Ridge** et la **régression Lasso**.

La régression Ridge (l_2) et la régression Lasso (l_1) sont **des méthodes de régression pénalisée** qui visent à résoudre ces problèmes d'une manière différente. Au lieu d'ajouter ou de retirer des variables une par une, elles pénalisent la taille des coefficients de régression.

En quelque sorte, elles construisent un modèle en optimisant tous les coefficients simultanément avec la contrainte de la pénalité k (ou λ selon les préférences).

Méthode de régression Ridge

On souhaite ici minimiser la fonction Φ donnée par :

$$\begin{aligned} & \Phi(\beta) = \|Y - X\beta\|_2^2 + k\,\|\beta\|_2^2 && (\text{avec } k \in \mathbb{R}_+^*) \\ \Longrightarrow \quad & \Phi(\beta) = (Y - X\beta)^T(Y - X\beta) + k\sum_{j=1}^{p}\beta_j^2 && (\text{avec } k \in \mathbb{R}_+^*) \end{aligned}$$

```r
data_cookies <- read.table(file = "cookies.csv", header = TRUE, sep = ",")

Y <- data_cookies[, 1]
X <- as.matrix(data_cookies[, -1])

# La régression Ridge nécessite que les données soient centrées et réduites
X_scaled <- scale(X)
ridge_model <- lm.ridge(Y ~ X_scaled, lambda = seq(0, 20, by = 0.1))

# On sélectionne le meilleur k via la validation croisée ou d'autres critères
best_k <- select(ridge_model)
```

```
## modified HKB estimator is -2.818936e-28
## modified L-W estimator is -1.068692e-27
## smallest value of GCV  at 0.1
```

```r
best_k
```

```
## NULL
```

Il est important de noter que la régression Ridge nécessite que les données soient centrées et réduites pour s'assurer que la pénalité est appliquée uniformément à tous les prédicteurs, ce qui est réalisé ici avec la fonction scale().

En utilisant la librairie "MASS", il semble y avoir un problème dans la méthode de régression pénalisée "Ridge", il n'arrive pas à trouver un k qui semble convenir dans notre régression. Nous avons donc décidé d'utiliser la librairie "glmnet".

```r
Y_matrix <- matrix(Y, length(Y), 1)

cv_ridge <- cv.glmnet(X_scaled, Y_matrix, alpha = 0)  # alpha = 0 pour ridge

# Affiche le meilleur lambda (k)  trouvé
print(cv_ridge$lambda.min)
```

```
## [1] 11.42432
```

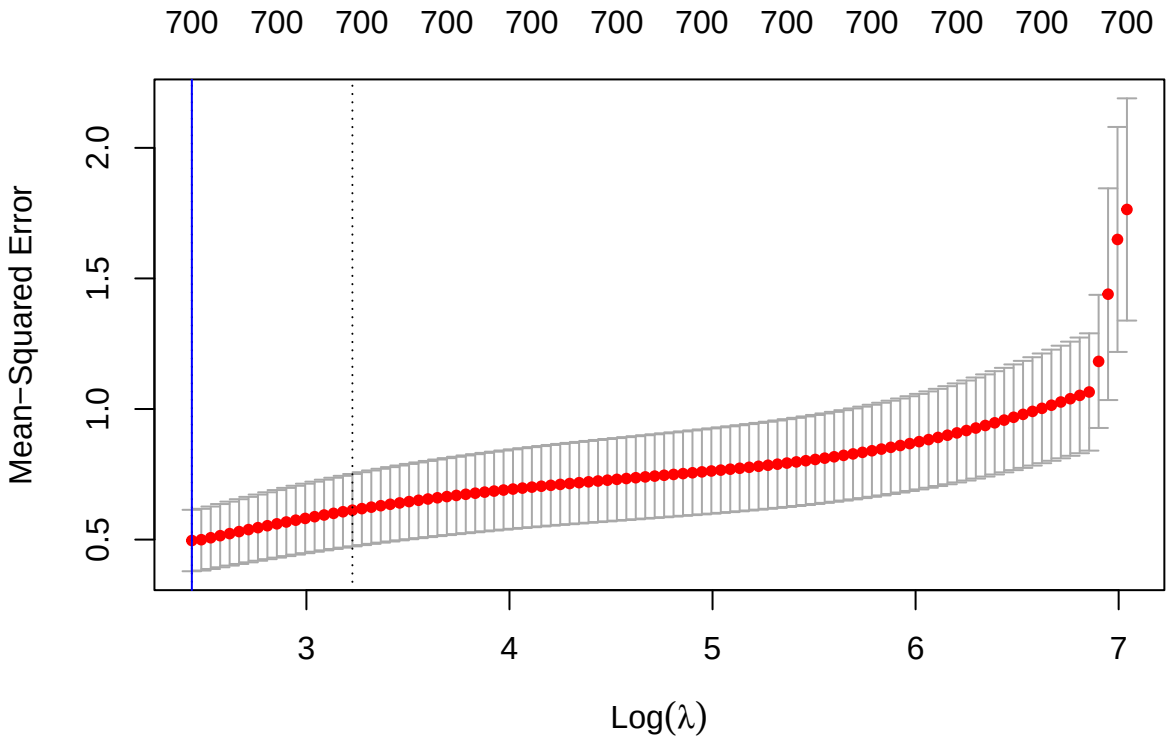
```r
# Ajuste le modèle final avec le meilleur lambda
final_ridge_model <- glmnet(X_scaled, Y_matrix, alpha = 0, lambda = cv_ridge$lambda.min)
```

La fonction `cv.glmnet()` trouve empiriquement la valeur qui minimise l'erreur de validation croisée (c'est-à-dire, une minimisation de notre fonction Φ).

Le modèle final est ajusté avec ce k optimal, qui vaut ici environ 11,4.

La fonction `cv.glmnet()` a donc utilisé une méthode de validation croisée, et cette librairie permet l'affichage très élégant de cette méthode là.

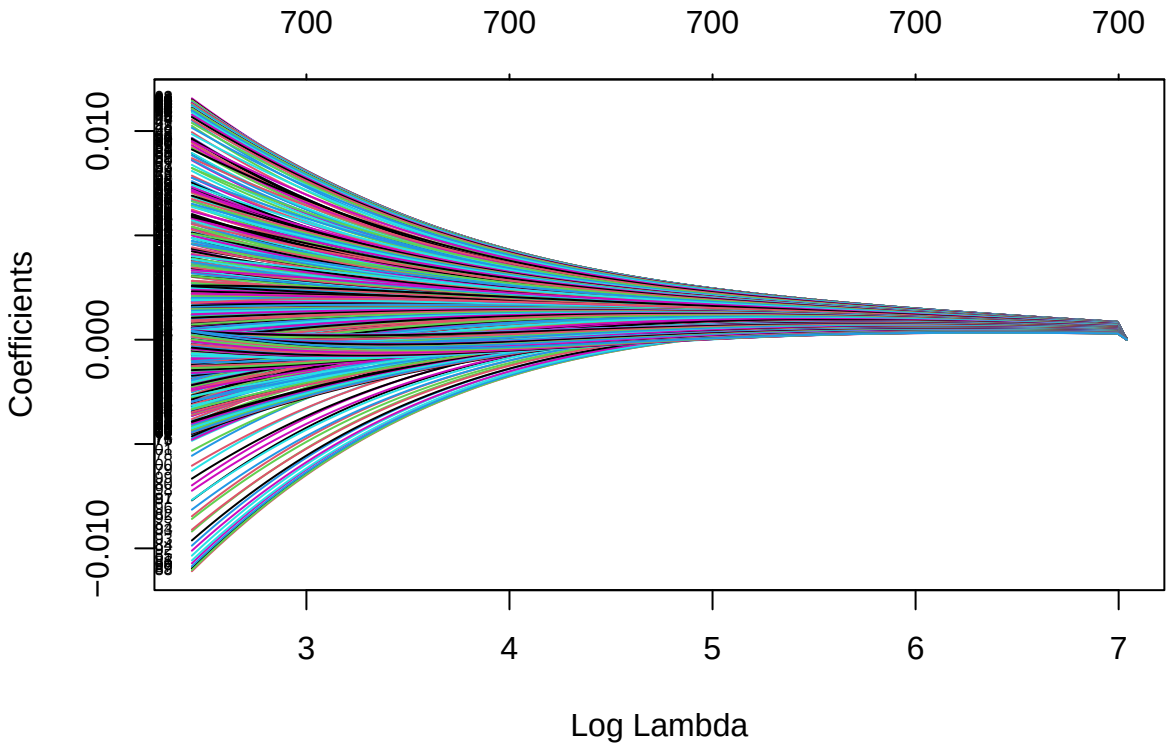
```r
# Validation croisée pour Ridge
cv_ridge <- cv.glmnet(X_scaled, Y, alpha = 0)

# Plot de la validation croisée pour Ridge
plot(cv_ridge)
abline(v = log(cv_ridge$lambda.min), col = "blue")
```



Ce graphique de la validation croisée permet de visualiser l'influence du paramètre de régularisation k ($= \lambda$) sur la performance du modèle. La ligne verticale bleue marque le **lambda optimal** ($\approx 11.4 \approx \ln(2.43)$).

On peut alors tracer le joli chemin de régularisation pour les différentes valeurs de k lorsqu'il augmente (de gauche à droite) :

```r
# Ajuster un modèle Ridge
ridge_model <- glmnet(X_scaled, Y, alpha = 0)

# Trajectoire des coefficients Ridge
plot(ridge_model, xvar = "lambda", label = TRUE)
```



Après avoir déterminé la meilleure valeur de lambda, les coefficients correspondants (les β) sont extraits. Les variables les plus importantes, c'est-à-dire celles dont les coefficients ont les valeurs absolues les plus élevées, sont ensuite identifiées. Cela fournit un aperçu des prédicteurs qui ont le plus d'impact dans le modèle Ridge.

On construit ainsi notre ensemble $\mathcal{M} \subset [\![1, p]\!]$.

```
ridge_model <- glmnet(X_scaled, Y, alpha = 0)
coef_ridge <- predict(ridge_model, type = "coefficients", s = cv_ridge$lambda.min)
important_vars_ridge <- order(abs(coef_ridge), decreasing = TRUE)

head(important_vars_ridge)
```

```
## [1]   1 403 404 402 405 406
```

Méthode de régression Lasso

On souhaite ici minimiser la fonction Φ donnée par :

$$\Phi(\beta) = \|Y - X\beta\|_2^2 + k\,\|\beta\|_1 \qquad (\text{avec } k \in \mathbb{R}_+^*)$$

$$\Longrightarrow \quad \Phi(\beta) = (Y - X\beta)^T(Y - X\beta) + k\sum_{j=1}^{p}|\beta_j| \qquad (\text{avec } k \in \mathbb{R}_+^*)$$

Les données sont chargées de la même manière que pour Ridge. Comme pour la régression Ridge, les données pour la régression Lasso doivent également être centrées et réduites.

```
data_cookies <- read.table(file = "cookies.csv", header = TRUE, sep = ",")
Y <- data_cookies[, 1]
X <- as.matrix(data_cookies[, -1])
X_scaled <- scale(X)
```

La fonction `cv.glmnet()` est utilisée pour ajuster un modèle Lasso (en fixant `alpha` $= 1$) et pour effectuer la validation croisée afin de déterminer le meilleur lambda. Encore une fois, lambda correspond au paramètre de régularisation k, mais dans le contexte de Lasso, il contrôle la force de la pénalité qui peut réduire certains coefficients totalement à zéro !

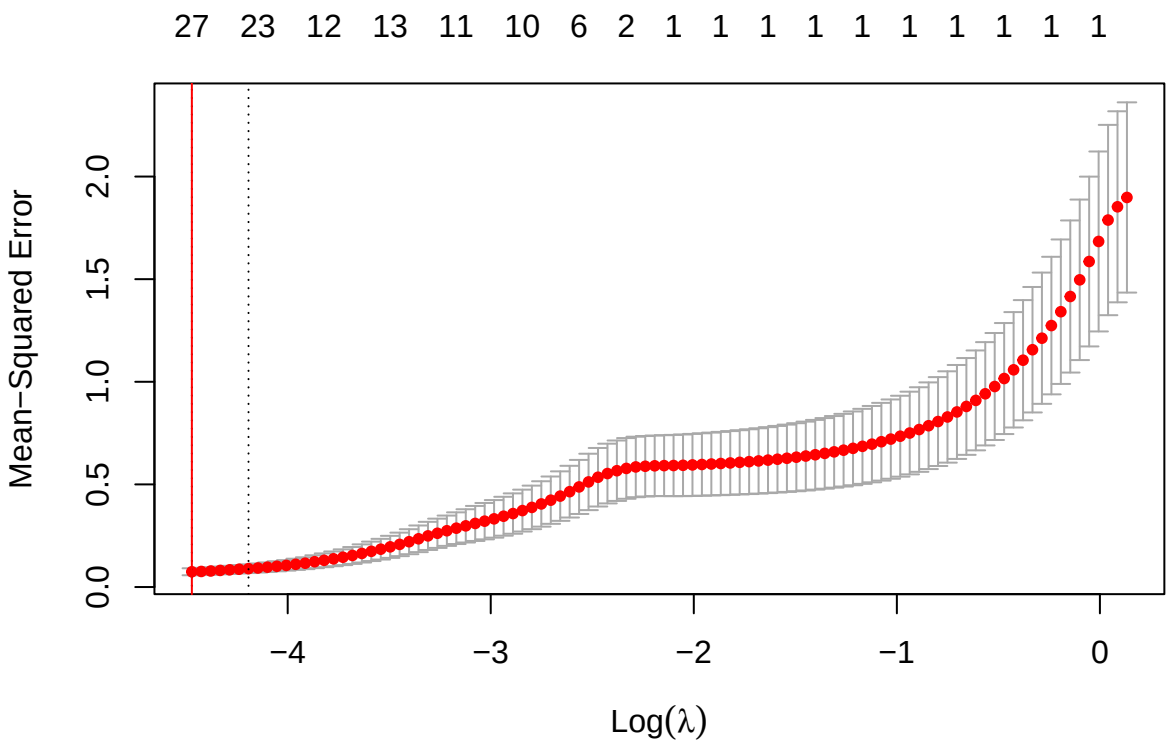
```
cv_lasso <- cv.glmnet(X_scaled, Y, alpha = 1)  # alpha = 1 pour lasso

# Affiche le meilleur lambda (k) trouvé
print(cv_lasso$lambda.min)
```

```
## [1] 0.01142432
```

```
final_lasso_model <- glmnet(X_scaled, Y, alpha = 1, lambda = cv_lasso$lambda.min)
```

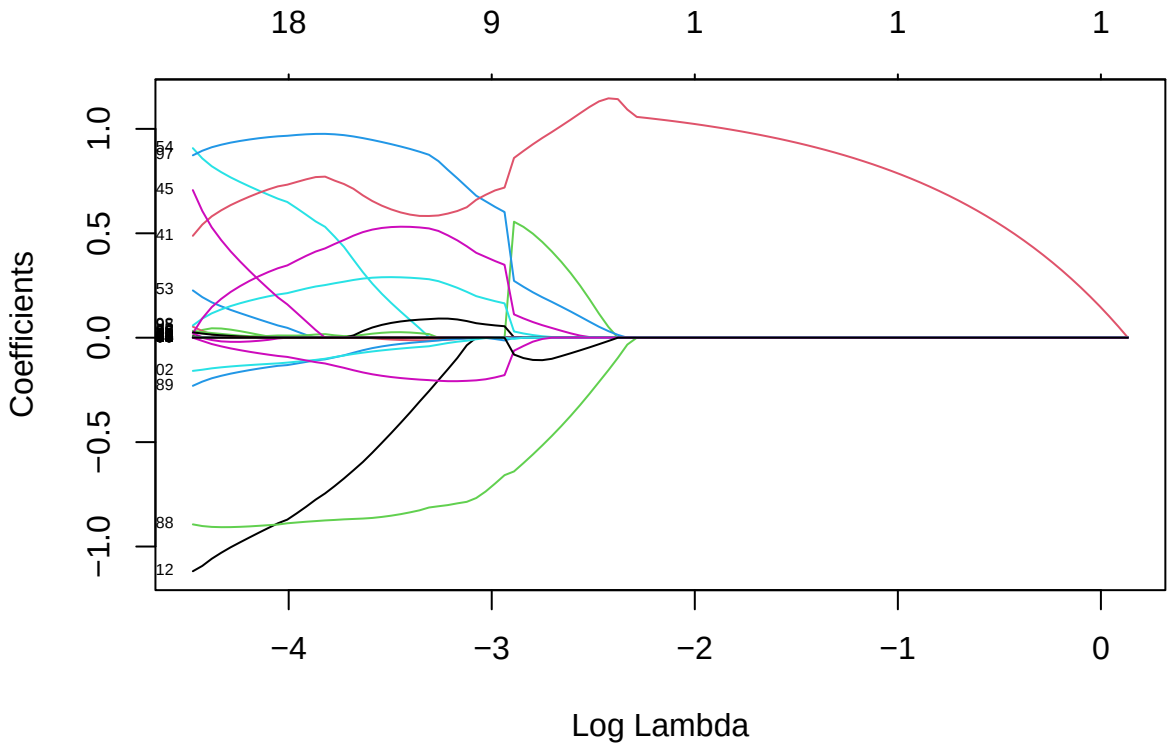
Un graphique de validation croisée est également utile pour le Lasso, permettant de voir l'effet du lambda sur la performance du modèle. Ici, c'est le trait rouge qui indique l'emplacement du meilleur lambda.

```
plot(cv_lasso)
abline(v = log(cv_lasso$lambda.min), col = "red")
```



Nous pouvons alors tracer le fameux chemin de régularisation. Le chemin du Lasso montre l'évolution des coefficients et la sélection de variables réalisée par le Lasso à mesure que le lambda augmente.

```
lasso_model <- glmnet(X_scaled, Y, alpha = 1)
plot(lasso_model, xvar = "lambda", label = TRUE)
```



Les coefficients les plus significatifs à la meilleure valeur de lambda sont extraits et ordonnés pour identifier les prédicteurs les plus influents.

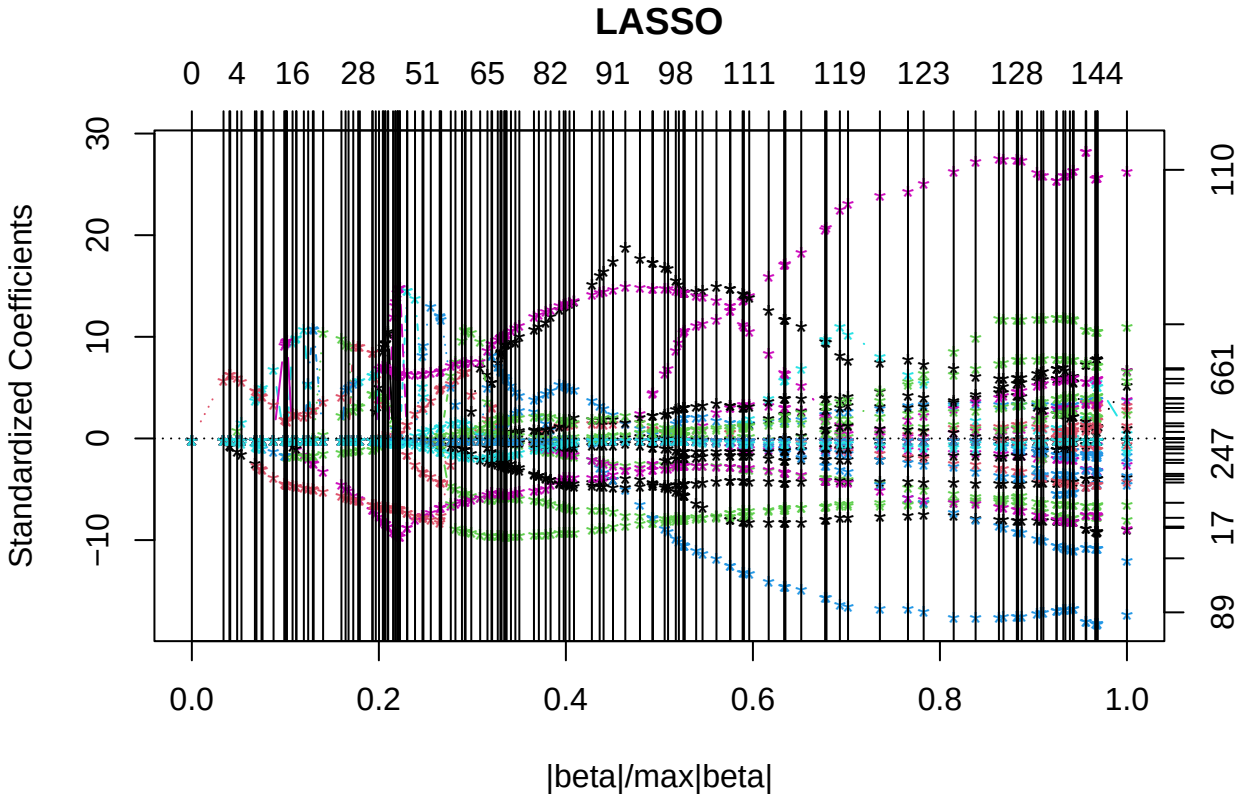
Contrairement à Ridge, certains coefficients peuvent être exactement **nuls**, ce qui indique que le "Lasso" a choisi de ne pas inclure ces variables dans le modèle.

```
lasso_model <- glmnet(X_scaled, Y, alpha = 1)
coef_lasso <- predict(lasso_model, type = "coefficients", s = cv_lasso$lambda.min)
important_vars_lasso <- order(abs(coef_lasso), decreasing = TRUE)

head(important_vars_lasso)
```

```
## [1]   1 313 155 489 398 446
```

À noter qu'on pouvait aussi obtenir tous ces résultats avec la librairie "lars" !

```
Y <- data_cookies[, 1]
X <- as.matrix(data_cookies[, -1])

lasso_model <- lars(X, Y, type = "lasso", use.Gram = FALSE)

plot(lasso_model)
```


LASSO
0 4 16 28 51 65 82 91 98 111 119 123 128 144
Standardized Coefficients
30
20
10
0
−10
110
661
247
17
89
0.0
0.2
0.4
0.6
0.8
1.0
|beta|/max|beta|

Conclusion

Dans cette étude, les modèles de régression **Ridge** et **Lasso** ont démontré leur valeur dans la prédiction de la teneur en graisse des cookies à partir des spectres initiaux. D'abord, la régression Ridge a révélé, à travers le tracé des chemins de régularisation, comment elle minimise l'impact de la multicollinéarité, alors que la régression Lasso, sur le même type de graphique, a permis de distinguer clairement les variables les plus significatives en réduisant certains coefficients à zéro.

Ces observations visuelles ont confirmé l'efficacité de la **pénalisation** pour gérer les données spectrales de haute dimension et ont mis en évidence l'importance des méthodes de **sélection de variables** dans l'amélioration de la précision prédictive des modèles analytiques.

Dans le TP1, nous nous étions contenté de créer 5 nouvelles variables découlant de ces 700 variables, tandis que dans ce TP2, nous nous sommes vraiment concentré sur la sélection des variables importantes dans notre travail de régression linéaire.