# Analyse des données prostate: la régression ridge

julien.chiquet@genopole.cnrs.fr

Module MPR – option modélisation, 20 novembre 2009

## Table des matières



## 1 Motivations

Soit le modèle linéaire défini par

$$y = \beta_0 + \sum_{i=1}^{p} X_i \beta_i + \varepsilon,$$

où $y$ est la réponse à expliquer par le vecteur de variables $X = (X_1, \ldots, X_p)^\intercal$, avec $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Notons $\mathbf{y}$ le vecteur de taille $n$ contenant $n$ observations de $y$ et $\mathbf{X}$ la matrice de taille $n \times p$ dont la $i^{\text{e}}$ colonne contient les $n$ observations relatives à la variable $X_i$. Contrairement à la formulation utilisée lors de la résolution des moindres carrés ordinaires, nous mettons de côté l'intercept. Ainsi, le modèle s'écrit

$$\mathbf{y} = \beta_0 \mathbf{1}_n + \mathbf{X}\boldsymbol{\beta}_{\setminus 0} + \boldsymbol{\varepsilon},$$

où $\boldsymbol{\beta}_{\setminus 0} = (\beta_1, \ldots, \beta_p)^\intercal$ est le vecteur $\boldsymbol{\beta}$ privé de l'élément $\beta_0$, et $\mathbf{1}_n$ est vecteur de taille $n$ rempli de 1 .

Nous savons que l'estimateur des moindres carrés est sans biais ; nous savons même qu'il est l'estimateur ayant la variance la plus petite possible parmi les estimateurs linéaires sans biais de $\boldsymbol{\beta}$. Ainsi, si l'on cherche un estimateur dont l'erreur de prédiction soit plus petite que celle de l'estimateur des moindres carrés, nous devons nécessairement gagner sur la variance tout en perdant le fait que l'estimateur soit non biaisé[1]. On espère que le gain dû à la réduction de la variance soit plus grand que la perte due au biais.

C'est sous ces motivations qu'ont été proposées les méthodes de régressions pénalisées, qui forcent les éléments de $\boldsymbol{\beta}$ à avoir une certaines forme en vu de réduire l'erreur de prédiction.

## 2 La régression ridge

La régression ridge impose une pénalité sur la taille des coefficients de $\boldsymbol{\beta}_{\setminus 0}$ (*pas* sur l'intercept !). En effet, les $\beta_i$ prenant de grandes valeurs induisent une grande variance. En limitant la taille des coefficients, on espère gagner en terme d'erreur de prédiction. À cet effet, la régression ridge minimise la somme des carrés résiduels à laquelle est adjointe un terme dépendant de la norme 2 du vecteur des coefficients :

$$\widehat{\boldsymbol{\beta}}^{\text{ridge}} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \left\{ \text{RSS}(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}_{\setminus 0}\|_2^2 \right\}, \qquad \text{RSS}(\boldsymbol{\beta}) = \left\| \mathbf{y} - \mathbf{1}_n \beta_0 - \mathbf{X}\boldsymbol{\beta}_{\setminus 0} \right\|_2^2. \quad (1)$$

Le paramètre $\lambda \geq 0$ dose le niveau de pénalité : lorsque $\lambda \to 0$, on s'approche de la solution des moindres carrés, alors que pour $\lambda \to \infty$, le niveau de pénalité devient si fort que toutes les valeurs de $\boldsymbol{\beta}$ sont mises à 0. Le théorème suivant donne la solution de ce problème.

**Théorème.** *On suppose que* $\mathbf{X}$ *est centrée/réduite. La solution* $\widehat{\boldsymbol{\beta}}^{ridge}$ *du problème* (1) *est donnée par*

$$\widehat{\boldsymbol{\beta}}_{\setminus 0}^{\text{ridge}} = \mathbf{S}_\lambda^{-1} \mathbf{X}^\intercal \tilde{\mathbf{y}}, \;\; \widehat{\beta}_0^{\text{ridge}} = \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i, \quad (2)$$

*où* $\mathbf{S}_\lambda = \mathbf{X}^\intercal \mathbf{X} + \lambda \mathbf{I}_p$ *et* $\tilde{\mathbf{y}} \triangleq \mathbf{y} - \bar{y}\mathbf{1}_n$.

*Démonstration.* Concernant l'intercept, la dérivée par rapport à $\beta_0$ donne

$$\frac{\partial}{\partial \beta_0} \text{RSS}(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}_{\setminus 0}\|_2^2 = 0 \Leftrightarrow \mathbf{1}_n^\intercal (\mathbf{y} - \beta_0 \mathbf{1}_n - \mathbf{X}\boldsymbol{\beta}) = 0 \Leftrightarrow \sum_i y_i - n\beta_0 - \mathbf{1}_n^\intercal \mathbf{X}\boldsymbol{\beta} = 0.$$

En notant $\bar{X}^j = 1/n \sum_i X_i^j$ la moyenne de la $j^{\text{e}}$ colonne de $\mathbf{X}$ et en divisant par $n$, nous avons

$$\bar{\mathbf{y}} - \beta_0 - \sum_{j=1}^{p} \bar{X}_j \beta_j = 0,$$

[1] Il s'agit du fameux compromis *biais/variance*

soit, lorsque $\mathbf{X}$ est centrée en colonne, $\hat{\beta}_0 = \bar{\mathbf{y}}$. Pour les autres paramètres $\boldsymbol{\beta}_{\setminus 0}$, l'hypothèse de gradient nul donne[2]

$$\nabla_{\boldsymbol{\beta}_{\setminus 0}}(\mathrm{RSS}(\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}_{\setminus 0}\|_2^2) = 0 \Leftrightarrow 2\mathbf{X}^\intercal(\mathbf{y} - \beta_0\mathbf{1}_n - \mathbf{X}\boldsymbol{\beta}_{\setminus 0}) + 2\lambda\boldsymbol{\beta}_{\setminus 0} = 0.$$

En estimant $\beta_0$ par $\bar{y}$, nous avons $\mathbf{y} - \beta_0\mathbf{1}_n = \tilde{\mathbf{y}}$. Le gradient nul implique donc que

$$\mathbf{X}^\intercal\tilde{\mathbf{y}} - \mathbf{X}^\intercal\mathbf{X}\boldsymbol{\beta}_{\setminus 0} + \lambda\boldsymbol{\beta}_{\setminus 0} = 0 \Leftrightarrow (\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I}_p)\boldsymbol{\beta}_{\setminus 0} = \mathbf{X}^\intercal\tilde{\mathbf{y}},$$

d'où la solution (2). □

**Théorème.** *Espérance et variance de* $\hat{\boldsymbol{\beta}}_{\setminus 0}^{\mathrm{ridge}}$ *sont données par*

$$\mathbb{E}\left(\hat{\boldsymbol{\beta}}_{\setminus 0}^{\mathrm{ridge}}\right) = \mathbf{S}_\lambda^{-1}\mathbf{X}^\intercal\mathbf{X}\boldsymbol{\beta}, \quad \mathrm{Var}\left(\hat{\boldsymbol{\beta}}_{\setminus 0}^{\mathrm{ridge}}\right) = \sigma^2\mathbf{S}_\lambda^{-1}\left(\mathbf{I}_p - \lambda\mathbf{S}_\lambda^{-1}\right).$$

*Démonstration.* Pour l'espérance, il suffit d'exprimer l'estimateur de la régression ridge en fonction de celui des moindres carrés :

$$\hat{\boldsymbol{\beta}}_{\setminus 0}^{\mathrm{ridge}} = \mathbf{S}_\lambda^{-1}\mathbf{X}^\intercal\tilde{\mathbf{y}} = \mathbf{S}_\lambda^{-1}\mathbf{X}^\intercal\mathbf{X}\hat{\boldsymbol{\beta}}^{\mathrm{ols}}$$

On constate que $\hat{\boldsymbol{\beta}}_{\setminus 0}^{\mathrm{ridge}}$ est effectivement biaisé, à moins que $\lambda = 0$ puisque dans ce cas $\mathbf{S}_\lambda = \mathbf{X}^\intercal\mathbf{X}$.

Pour la variance, on utilise les règles de calcul standards pour les vecteurs aléatoires :

$$\begin{aligned}\mathrm{Var}\left(\hat{\boldsymbol{\beta}}_{\setminus 0}^{\mathrm{ridge}}\right) &= \mathrm{Var}\left(\mathbf{S}_\lambda^{-1}\mathbf{X}^\intercal\tilde{\mathbf{y}}\right) = \mathbf{S}_\lambda^{-1}\mathbf{X}^\intercal\mathrm{Var}(\tilde{\mathbf{y}})\mathbf{X}\mathbf{S}_\lambda^{-1} = \sigma^2\mathbf{S}_\lambda^{-1}\mathbf{X}^\intercal\mathbf{X}\mathbf{S}_\lambda^{-1} \\ &= \sigma^2\mathbf{S}_\lambda^{-1}(\mathbf{S}_\lambda + \lambda\mathbf{I}_p)\mathbf{S}_\lambda^{-1} = \sigma^2\mathbf{S}_\lambda^{-1}(\mathbf{I}_p + \lambda\mathbf{S}_\lambda^{-1}).\end{aligned}$$

□

## 2.1 Calcul pratique

En pratique, on est amené à calculer la solution de la régression ridge pour un grand nombre de valeurs du paramètres $\lambda$, afin de choisir la valeur $\lambda^\star$ qui convient : le calcul de l'estimateur $\hat{\boldsymbol{\beta}}^{\mathrm{ridge}}$ réclamant une inversion matricielle, la résolution du problème (2) pour une large grille de $\lambda$ peut rapidement s'avérer coûteuse numériquement. À cet effet, le résultat suivant est précieux, accélérant drastiquement la procédure.

**Théorème.** *Soit la décomposition en valeurs singulières de* $\mathbf{X}$

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\intercal,$$

*où* $\mathbf{U}, \mathbf{V}$ *sont orthogonales et* $\mathbf{D}$ *diagonale de termes diagonaux* $(d_1, \ldots, d_i, \ldots, d_p)$. *Alors,*

$$\hat{\boldsymbol{\beta}}_{\setminus 0}^{\mathrm{ridge}} = \mathbf{V}\boldsymbol{\Delta}_\lambda\mathbf{U}^\intercal\tilde{\mathbf{y}},$$

*où* $\boldsymbol{\Delta}_\lambda$ *est diagonale de termes diagonaux* $\Delta_i$ *donnés par*

$$\Delta_i = \frac{d_i}{d_i^2 + \lambda}.$$

[2] Pour dériver le carrée de la norme 2, il suffit de l'écrire sous la forme $\|\boldsymbol{\beta}\|_2^2 = \boldsymbol{\beta}^\intercal\boldsymbol{\beta}$.

*Démonstration.* Tout d'abord, notons que la décomposition spectrale de la matrice carré et symétrique $\mathbf{X}^\intercal\mathbf{X}$ est donnée par

$$\mathbf{X}^\intercal\mathbf{X} = (\mathbf{U}\mathbf{D}\mathbf{V}^\intercal)^\intercal\mathbf{U}\mathbf{D}\mathbf{V}^\intercal = \mathbf{V}\mathbf{D}\mathbf{U}^\intercal\mathbf{U}\mathbf{D}\mathbf{V}^\intercal = \mathbf{V}\mathbf{D}^2\mathbf{V}^\intercal,$$

ainsi $\mathbf{V}$ est la matrice des vecteurs propres de $\mathbf{X}^\intercal\mathbf{X}$ et $\text{diag}(\mathbf{D})$ les valeurs propres correspondantes. En utilisant l'orthogonalité de $\mathbf{V}$, nous avons

$$\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I}_p = \mathbf{V}\mathbf{D}^2\mathbf{V}^\intercal + \lambda\mathbf{I}_p = \mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{V}^\intercal\mathbf{V})\mathbf{V}^\intercal = \mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I}_p)\mathbf{V}^\intercal$$

Finalement,

$$\begin{aligned}\boldsymbol{\beta}_{\backslash 0} &= (\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\mathbf{X}^\intercal\tilde{\mathbf{y}} = \left(\mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I}_p)\mathbf{V}^\intercal\right)^{-1}(\mathbf{U}\mathbf{D}\mathbf{V}^\intercal)^\intercal\tilde{\mathbf{y}} \\ &= \mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I}_p)^{-1}\mathbf{V}^\intercal\mathbf{V}\mathbf{D}\mathbf{U}^\intercal\tilde{\mathbf{y}} = \mathbf{V}\boldsymbol{\Delta}_\lambda\mathbf{U}^\intercal\tilde{\mathbf{y}}.\end{aligned}$$

□

Ainsi, le calcul des solutions de la régression ridge pour $K$ valeurs de $\lambda$ se ramène à deux opérations matricielles : la décomposition en valeurs singulières de $\mathbf{X}$ et *un* produit matricielle entre $\mathbf{V}$ de taille $p \times p$ et une matrice de taille $p \times K$ contenant en colonne les valeurs de $\boldsymbol{\Delta}_\lambda\mathbf{U}^\intercal\tilde{\mathbf{y}}$ pour chaque valeur de $\lambda$[3].

## 2.2 Notion de degrés de liberté effectifs

D'une manière générale, le nombre de degré de liberté d'une méthode permet de décrire son niveau de complexité : ainsi, pour les moindres carrés ordinaires, les degrés de liberté sont au nombre de $p$, le nombre de variables à estimer. Une définition propre au méthodes de régularisation s'impose pour traduire le niveau de complexité de la méthode. Celle-ci dépend du paramètre de pénalisation $\lambda$ :

**Définition.** *Soit une méthode de régularisation transformant les valeurs observées de $y$ en valeurs prédites selon une matrice $\hat{\mathbf{H}}_\lambda$, c'est-à-dire, que*

$$\hat{y} = \hat{\mathbf{H}}_\lambda\tilde{y}$$

*Le nombre de degrés de liberté effectifs de cette méthode est définie par*

$$\text{df}(\lambda) = \text{Tr}(\hat{\mathbf{H}}_\lambda).$$

La trace d'une matrice, égale à la somme de ses termes diagonaux (où, de manière équivalente, à la somme de ses valeurs propres), traduit la *complexité* de cette matrice, et donc de la méthode de régression dans ce cas. Dans le cas de la régression ridge, il n'est pas difficile de voir[4] que

$$\text{df}(\lambda) = \text{Tr}(\hat{\mathbf{H}}_\lambda) = \text{Tr}(\mathbf{X}\mathbf{V}\boldsymbol{\Delta}_\lambda\mathbf{U}^\intercal) = \text{Tr}(\boldsymbol{\Delta}_\lambda\mathbf{U}^\intercal\mathbf{X}\mathbf{V}) = \text{Tr}(\boldsymbol{\Delta}_\lambda\mathbf{D}) = \sum_{i=1}^{p}\frac{d_i^2}{d_i^2 + \lambda}. \quad (3)$$

[3] Confer en annexe la fonction `ridge.regression`

[4] On rappelle que $\text{Tr}(A^\intercal) = \text{Tr}(A)$ et que $\text{Tr}(AB) = \text{Tr}(BA)$

# 3 Code R commenté

On rappelle que les données prostate contiennent un vecteur `x`, un vecteur `y` et un vecteur de booléens `set` indiquant les données de l'ensemble d'apprentissage.

## 3.1 Initialisation

On charge les données et les fonctions qui sont stockées dans le fichier `functions.R`, puis on centre et on réduit la matrice $\mathbf{X}$. On construit les données de test et d'apprentissage.

```
> source("functions.R")
> load("prostate.rda")
> x <- as.matrix(x)
> x <- scale(x)
> test <- !set
> n <- sum(set)
> n.test <- sum(test)
> x.test <- x[test, ]
> y.test <- y[test]
> x <- x[set, ]
> y <- y[set]
```

## 3.2 Estimation

La fonction `ridge.regression`, qui se trouve pour ma part dans le fichier `functions.R`, fait *grosso-modo* la même chose que la fonction `lm.ridge` : elle prend en argument une matrice `x` de prédicteurs, un vecteur de réponse `y` et un vecteur de paramètres `lambda`. En sortie, cette fonction renvoie une structure de liste contenant

1. une matrice `beta` à $p$ colonnes et autant de lignes que d'éléments dans `lambda` : la ligne $i$ correspond à l'estimateur de la régression ridge pour une valeur de pénalité égale à `lambda[i]` ;
2. un scalaire `beta0` contenant l'estimation de l'intercept ;
3. un vecteur `df` contenant les degrés de liberté tels que définis en (3), calculés pour les valeurs contenues dans `lambda`.

Ainsi, pour obtenir les estimations de 1000 valeurs de $\lambda$ variant de 0 à 1000 plus la valeur que R obtient en $\lambda = \infty$, on saisit le code suivant :

```
> n.lambda <- 1000
> lambda <- c(seq(0, 1000, length = n.lambda - 1), Inf)
> out <- ridge.regression(x, y, lambda)
```

On peut par la suite obtenir l'erreur de prédiction pour toutes les valeurs de $\lambda$ en une seule opération matricielle :

```
> err <- 1/n.test * colSums((y.test - out$beta0 - x.test %*% t(out$beta))^2)
```

En effet, l'opération `x.test %*% t(out$beta)` créé une matrice dont la $i^{\text{e}}$ colonne correspond à $\mathbf{X}\hat{\boldsymbol{\beta}}_{\lambda}^{\text{ridge}}$ pour la valeur $\lambda =$ `lambda[i]`. Puis, `R` comprend par lui même qu'il faut répéter le vecteur `y.test` autant de fois qu'il y a de colonnes dans `x.test %*% t(out$beta)` pour que l'opération de soustraction matricelle soit consistante. Après avoir mis le tout au carré, on veille à faire une somme en colonne pour obtenir l'erreur correspondant à chacune des valeurs de $\lambda$.

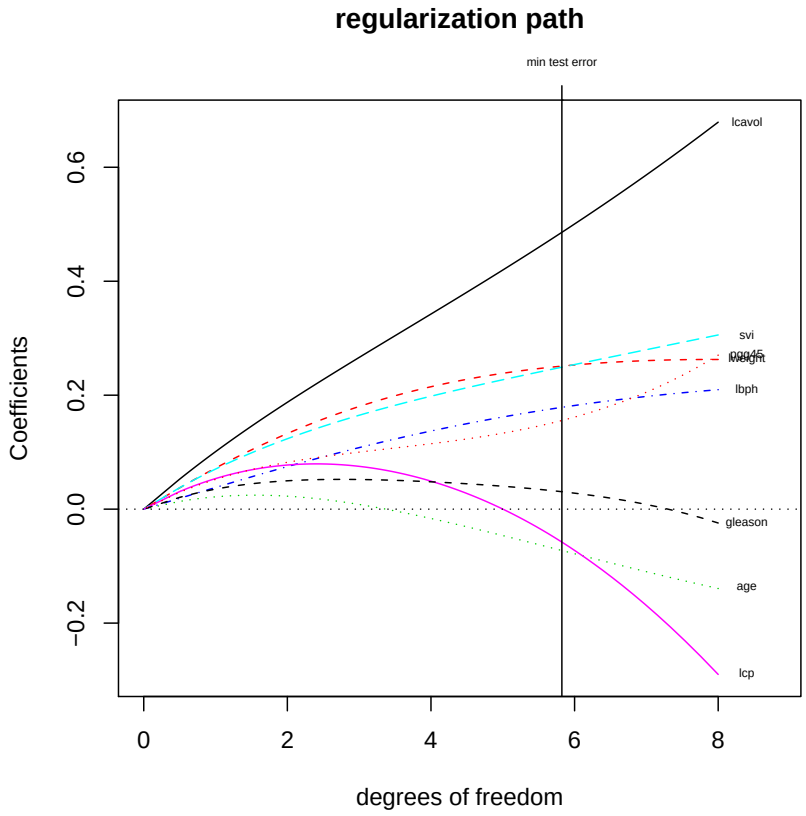
Enfin, on obtient $\lambda^\star$ et $\text{df}(\lambda)^\star$ en cherchant les valeurs minimisant l'erreur de prédiction :

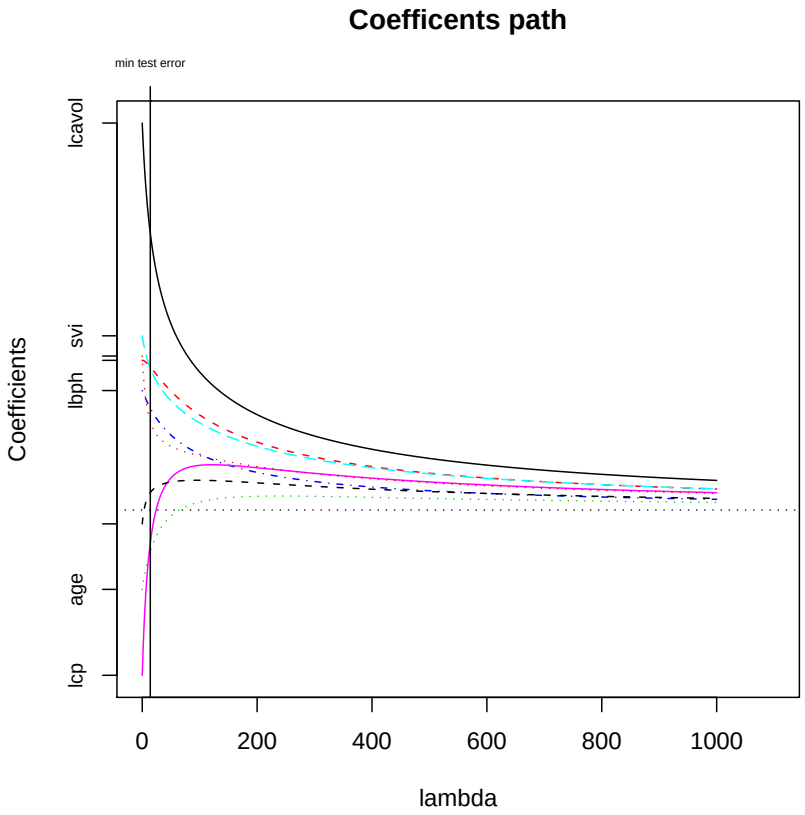
```
> lambda.star <- lambda[which.min(err)]
> df.star <- out$df[which.min(err)]
```

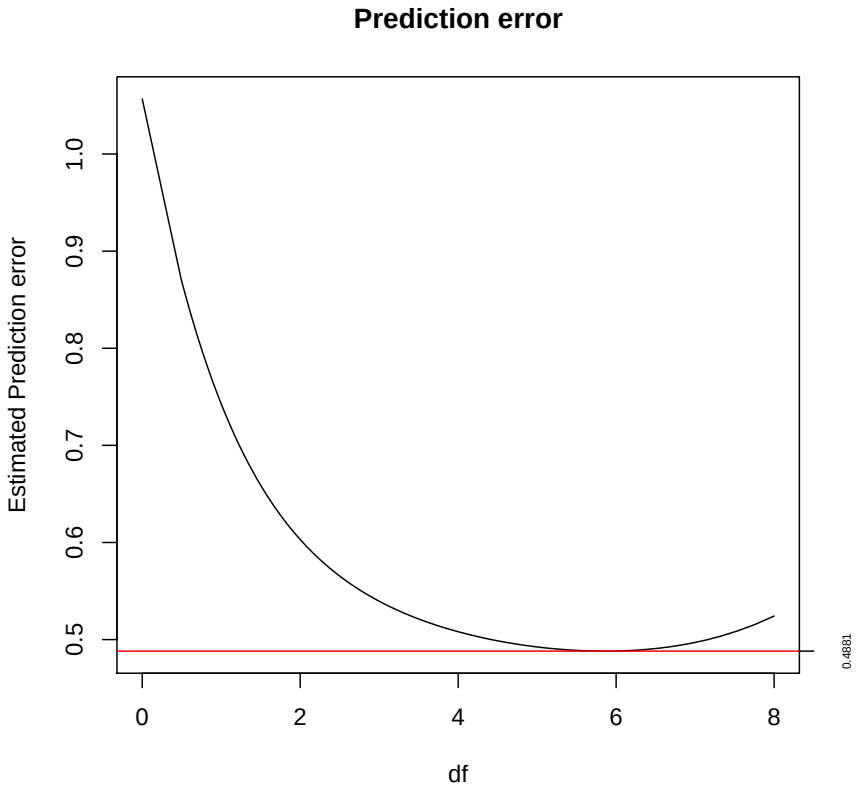
### 3.3 Représentation des résultats

Ce code permet de générer les 3 figures suivantes, résumant les résultats.

```
> par(mfrow = c(1, 3))
> plot.path(out$beta, out$df, err)
> plot.coef(out$beta[-n.lambda, ], lambda[-n.lambda], err)
> plot.err(err, out$df)
```




Coefficents path
min test error
lcavol
svi
lbph
age
lcp
Coefficients
0
200
400
600
800
1000
lambda


Prediction error
Estimated Prediction error
0.5
0.6
0.7
0.8
0.9
1.0
0.4881
0
2
4
6
8
df

# A Code des fonctions

## A.1 La fonction `ridge.regression`

Calcule l'estimateur du ridge et les degrés de libertés associés pour une matrice de prédicteurs `x` centrée/réduite, un vecteur de réponse `y` et une séquence de paramètres `lambda`.

```
> ridge.regression <- function(x, y, lambda = 0) {
+     n.lambda <- length(lambda)
+     y.tilde <- y - mean(y)
+     variables <- colnames(x)
+     p <- length(variables)
+     beta0 <- mean(y)
+     SVD <- svd(x)
+     attach(SVD)
+     Delta <- rep(d, n.lambda)/(rep(d^2, n.lambda) + rep(lambda,
+         each = p))
+     beta <- t(v %*% matrix((rep(t(u) %*% y.tilde, n.lambda) *
+         Delta), nrow = p))
+     df <- matrix(rep(d^2, n.lambda)/(rep(d^2, n.lambda) + rep(lambda,
+         each = p)), nrow = p)
+     df <- colSums(df)
+     detach(SVD)
+     colnames(beta) <- variables
+     return(list(beta = beta, beta0 = beta0, df = df))
+ }
```

## A.2 La fonction `plot.path`

Trace le chemin de régularisation pour une matrice contenant les estimateurs de $\beta$, chaque ligne correspondant à une valeur du paramètre $\lambda$. L'utilisateur fournit un vecteur `df` de degrés de libertés de la taille du nombre de ligne de la matrice `beta`, et un éventuellement vecteur d'erreur de prédiction de la taille de `df`.

```
> plot.path <- function(beta, df = NULL, err = NULL, mytitle = "regularization path",
+     grid.on = FALSE) {
+     beta.max <- which.max(rowSums(abs(beta)))
+     coord <- beta[beta.max, ]
+     if (is.null(df)) {
+         df <- rowSums(abs(beta))/sum(abs(beta[beta.max, ]))
+     }
+     matplot(df, beta, xlab = "degrees of freedom", type = "l",
+         ylab = "Coefficients", xlim = c(0, 1.1 * max(df)))
+     title(mytitle, line = 2.5)
+     abline(h = 0, lty = 3)
+     text(rep(1.05 * max(df), ncol(beta)), coord, colnames(beta),
```

```
+         cex = 0.5)
+     if (grid.on) {
+         stepid = trunc(as.numeric(dimnames(beta)[[1]]))
+         axis(3, at = df, labels = paste(stepid), cex.axis = 0.8)
+         abline(v = df)
+     }
+     if (!is.null(err)) {
+         df.star <- df[which.min(err)]
+         abline(v = df.star)
+         axis(3, at = df.star, labels = "min test error", cex.axis = 0.5)
+     }
+     return(df)
+ }
```

### A.3 La fonction `plot.coef`

Trace les valeurs des coefficients de $\beta$ en fonction de $\lambda$ en prenant en argument `beta`, la matrice contenant les estimateurs où chaque ligne correspondant à une valeur du paramètre $\lambda$, et le vecteur `lambda` correspondant. Éventuellement , prend en argument vecteur d'erreur de prédiction de la taille de `df`.

```
> plot.coef <- function(beta, lambda, err = NULL, mytitle = "Coefficents path") {
+     matplot(lambda, beta, xlab = "lambda", type = "l", yaxt = "n",
+         ylab = "Coefficients", xlim = c(0, 1.1 * max(lambda)))
+     title(mytitle, line = 2.5)
+     abline(h = 0, lty = 3)
+     axis(2, at = beta[which.max(rowSums(abs(beta))), ], labels = colnames(beta),
+         cex.axis = 0.8)
+     if (!is.null(err)) {
+         lambda.star <- lambda[which.min(err)]
+         abline(v = lambda.star)
+         axis(3, at = lambda.star, labels = "min test error",
+             cex.axis = 0.5)
+     }
+ }
```

### A.4 La fonction `plot.err`

Trace les valeurs de l'erreur de prédiction `err` en fonction des degrés de liberté `df`.

```
> plot.err <- function(err, df, mytitle = "Prediction error") {
+     plot(df, err, main = mytitle, type = "l", xlab = "df", ylab = "Prediction error")
+     abline(h = min(err), col = "red")
+     axis(4, at = min(err), labels = toString(round(min(err),
+         4)), cex.axis = 0.5)
+ }
```