

Régularisation

Mathieu Pigeon

UQAM

- 1 Introduction
- 2 Transformation des variables explicatives
- 3 Régression Ridge
- 4 Régression Lasso

Problématique - Régression

- Dans une problématique de type *régression*, on dispose d'une base de données de taille n : $(y_i, \mathbf{x}_i)_{i=1,2,\dots,n}$, avec $\mathbf{x}_i = [x_{i1} \ \cdots \ x_{ip}]$.
- y_i est la variable réponse (numérique) et \mathbf{x}_i est un vecteur de variables explicatives (numériques également), ou prédicteurs.
- L'objectif est de *prédire* la valeur de la variable réponse y^* pour une nouvelle observation dont les variables explicatives sont \mathbf{x}^* .

Méthode des K plus proches voisins

- On définit $\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^p x_j^2}$.
- Pour une base de données $(y_i, \mathbf{x}_i)_{i=1,2,\dots,n}$ et une valeur de K , on définit l'ensemble des K plus proches voisins du point \mathbf{x}_0 , $\mathcal{X}_0^{(K)}$, comme étant l'ensemble des K points \mathbf{x}_i de la base de données pour lesquels la valeur de $\|\mathbf{x}_i - \mathbf{x}_0\|_2$ est la plus petite.
- Pour une nouvelle observation \mathbf{x}_0 , la prédiction sera donnée par

$$\tilde{Y} = \frac{1}{K} \sum_{i: \mathbf{x}_i \in \mathcal{X}_0^{(K)}} y_i.$$

Exemple 1

Base de données *freaggnnumber* disponible dans la librairie *CASdatasets*.
Contient la fréquence totale de sinistres pour 12 513 classes d'assuré(e)s.
Les variables sont

- **ClaimNumber** : la fréquence totale pour la classe ;
- **Exposure** : l'exposition totale pour la classe (en année-police) ;
- **VehAge** : l'âge du véhicule ;
- **LicenceAge** : l'âge auquel le conducteur a obtenu son permis de conduire ;
- **DriverAge** : l'âge du conducteur.

Exemple 1 : Préparation des données

```
library(CASdatasets)
```

```
data(freaggnumber)
```

```
dataAGG <- freaggnumber
```

```
head(dataAGG)
```

DriverAge	LicenceAge	VehAge	Exposure	ClaimNumber
39	18	3	1356.402	192
35	18	3	1243.948	172
37	18	1	1263.064	172
38	18	3	1328.589	171
39	18	2	1346.795	170
40	18	3	1263.537	165

Exemple 1 : Modèle des K plus proches voisins

```
library(FNN)

FUNout <- function(x){
  sum((knn.reg(train = matrix(as.matrix(dataAGG)[,1:4],
    ncol = 4), y = dataAGG$ClaimNumber, k = x)$pred
    - dataAGG$ClaimNumber)^2)/length(dataAGG$ClaimNumber)
}

outMSE <- sapply(1:50, function(x) FUNout(x))
(1:50)[which(outMSE == min(outMSE))]
```

[1] 10

Exemple 1 : Modèle des K plus proches voisins

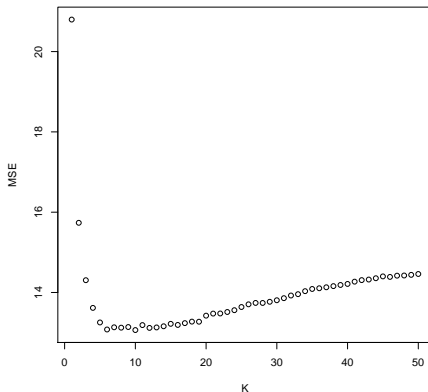


FIGURE – Évolution de l'erreur quadratique moyenne de validation en fonction de la valeur de K (le minimum est à $K = 10$).

Exemple 1 : Modèle des K plus proches voisins

```
modele1 <- knn.reg(train = matrix(as.matrix(dataAGG)[,1:4],  
                                ncol = 4),  
                  test = matrix(as.matrix(dataAGG)[,1:4], ncol = 4),  
                  y = dataAGG$ClaimNumber, k = 10)  
MSE_K_10 <- mean((modele1$pred - dataAGG$ClaimNumber)^2)  
MSE_K_10
```

10.64317

Méthode des K plus proches voisins

En dépit de sa simplicité, cette méthode présente de nombreux inconvénients :

- le modèle est **constant par morceau** \rightarrow variations abruptes ; et
- le modèle **performe mal en dimension élevée**, c'est-à-dire lorsque $p = \dim(\mathbf{X})$ est grand.

Régression linéaire

On a le modèle paramétrique suivant :

$$Y_i = \beta_0 + \sum_{j=1}^p \beta_j X_{ij} + \epsilon_i, \quad i = 1, \dots, n.$$

Les paramètres β_j , $j = 0, 1, \dots, p$ du modèle peuvent être estimés en minimisant l'erreur quadratique moyenne (MSE) sur un échantillon d'entraînement :

$$\left(\hat{\beta}_0, \dots, \hat{\beta}_p \right) = \operatorname{argmin}_{\beta_0, \dots, \beta_p} \frac{1}{n} \sum_{i=1}^n \left(Y_i - \hat{Y}_i \right)^2$$
$$\hat{y}_i = \hat{\beta}_0 + \sum_{j=1}^p \hat{\beta}_j X_{ij}.$$

Régression linéaire

Sous forme matricielle, on a

$$\mathbf{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & X_{11} & \dots & X_{1p} \\ \vdots & & \ddots & \vdots \\ 1 & X_{n1} & \dots & X_{np} \end{bmatrix}$$
$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_p \end{bmatrix}.$$

Régression linéaire

Le modèle est alors

$$\mathbb{E}[\mathbf{Y}] = \mathbf{X}\boldsymbol{\beta}$$

avec les paramètres donnés par

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \arg \min_{\boldsymbol{\beta}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) \\ &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}.\end{aligned}$$

Régression linéaire

Pour une nouvelle observation dont les variables explicatives sont $X_{(n+1)1}, \dots, X_{(n+1)p}$, la prédiction est donnée par

$$\tilde{y}_{n+1} = \hat{\beta}_0 + \sum_{j=1}^p \hat{\beta}_j X_{(n+1)j}.$$

On obtient ainsi un modèle qui est très **lisse** mais qui manque souvent de flexibilité pour capturer la relation entre la variable réponse et les variables explicatives.

Exemple 1 : Régression linéaire

```
modele2 <- lm(ClaimNumber ~ DriverAge + LicenceAge + VehAge,
              data = dataAGG, offset = Exposure)
```

```
summary(modele2)
```

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-463.07988	7.25980	-63.787	< 2e-16	***
DriverAge	0.62110	0.09266	6.703	2.13e-11	***
LicenceAge	11.77947	0.22877	51.491	< 2e-16	***
VehAge	6.15840	0.21764	28.296	< 2e-16	***

```
MSE_LIN <- mean((predict(modele2) - dataAGG$ClaimNumber)^2)
```

```
MSE_LIN
```

```
[1] 22769.49
```

Régression linéaire généralisée (GLM)

On a le modèle paramétrique suivant :

$$g(\mathbb{E}[Y_i]) = \beta_0 + \sum_{j=1}^p \beta_j X_{ij}, \quad i = 1, \dots, n,$$

ou, sous forme matricielle,

$$\mathbf{g}(\mathbb{E}[\mathbf{Y}]) = \mathbf{X}\boldsymbol{\beta}.$$

On retrouve une relation linéaire entre une transformation de l'espérance et les variables explicatives.

Exemple 1 : Régression linéaire généralisée (Poisson)

```
modele3 <- glm(ClaimNumber ~ DriverAge + LicenceAge + VehAge,
               family = poisson(link = "log"),
               data = dataAGG, offset = log(Exposure))
summary(modele3)
```

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-2.5930588	0.0183868	-141.03	<2e-16 ***
DriverAge	-0.0157666	0.0002351	-67.05	<2e-16 ***
LicenceAge	0.0522104	0.0009091	57.43	<2e-16 ***
VehAge	-0.0078364	0.0006060	-12.93	<2e-16 ***

```
MSE_Poisson <- mean((predict(modele3, type = "response")
                        - dataAGG$ClaimNumber)^2)
```

```
MSE_Poisson
[1] 12.62325
```

Transformation des variables explicatives

- Pour gagner en flexibilité dans le modèle, on peut considérer une transformation de l'ensemble des variables explicatives (\mathbf{X}).
- Pour une valeur entière positive K , on définit une transformation $h_k : \mathbb{R}^p \rightarrow \mathbb{R}$, $k = 1, \dots, K$.
- On aura alors

$$f(X_{ji}) = \gamma_0 + \sum_{k=1}^K \gamma_k h_k(X_{ij}), \quad j = 1, \dots, p$$
$$Y_i = \beta_0 + \sum_{j=1}^p \beta_j f(X_{ij}), \quad i = 1, \dots, n.$$

Transformation des variables explicatives

Plusieurs choix sont possibles pour les fonctions $h()$:

- polynomiales : $h_k(x) = x^k$ par exemple (généralement quand $p = 1$) ;
- trigonométrique : $h_k(x) = \sin(kx/2\pi)$ ou $h_k(x) = \cos(kx/2\pi)$;
- séries de Taylor, etc.

Exemple 1 : Liens

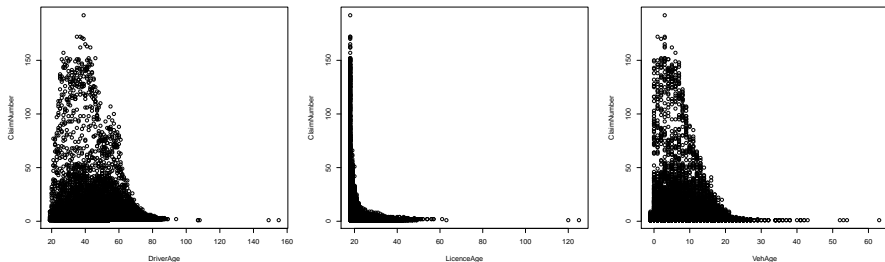


FIGURE – Difficile d'utiliser ces graphiques pour déterminer la forme de la fonction $h()$.

Régression polynomiale

Lorsque $p = 1$, on considère souvent

$$Y_i = \beta_0 + \beta_1 f(X_i) + \epsilon_i$$

$$f(X_i) = \gamma_0 + \sum_{k=1}^K \gamma_k X_i^k$$

$$\rightarrow Y_i = \beta_0 + \beta_1 \left(\gamma_0 + \sum_{k=1}^K \gamma_k X_i^k \right)$$

$$\rightarrow Y_i = \beta_0^* + \sum_{k=1}^K \beta_k^* X_i^k + \epsilon_i, \quad i = 1, \dots, n.$$

Régression polynomiale

- Si on choisit une petite valeur de K (par exemple $K = 1$), on obtient un modèle peu flexible : sous-ajustement.
- Si on choisit une grande valeur de K (par exemple $K = 20$), on obtient un modèle trop flexible : surajustement.
- $\rightarrow K$ peut être considéré comme un **hyperparamètre** du modèle et sa valeur obtenue par validation croisée.

Validation croisée

- Il y a plusieurs années (!), le manque de puissance des ordinateurs rendait l'utilisation de la validation croisée difficile, voire impossible.
- On a alors développé plusieurs critères dont la valeur était basée uniquement sur l'échantillon d'entraînement : AIC, BIC, AICc, R^2 , R^2 ajusté, etc.
- Ces approches (1) pouvaient ne pas conduire à des décisions cohérentes et (2) ne mesuraient pas vraiment la force prédictive d'un modèle.
- Aujourd'hui, les ordinateurs sont nettement plus puissants...

Exemple 1 : Régression Poisson polynomiale

```
dataAGG$DriverAge2 <- dataAGG$DriverAge^2  
dataAGG$DriverAge3 <- dataAGG$DriverAge^3  
dataAGG$LicenceAge2 <- dataAGG$LicenceAge^2  
dataAGG$LicenceAge3 <- dataAGG$LicenceAge^3  
dataAGG$VehAge2 <- dataAGG$VehAge^2  
dataAGG$VehAge3 <- dataAGG$VehAge^3
```


Exemple 1 : Régression Poisson polynomiale

```
modele4 <- glm(ClaimNumber ~ DriverAge + LicenceAge +
  VehAge + DriverAge2 + DriverAge3 +
  LicenceAge2 + LicenceAge3 + VehAge2 +
  VehAge3, family = poisson(link = "log"),
  data = dataAGG, offset = log(Exposure))
summary(modele4)
```

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	-4.159e-01	1.100e-01	-3.781	0.000156	***
DriverAge	-2.018e-02	5.942e-03	-3.397	0.000681	***
LicenceAge	-1.447e-01	8.576e-03	-16.873	< 2e-16	***
VehAge	-3.221e-03	2.225e-03	-1.448	0.147608	
DriverAge2	6.433e-05	1.272e-04	0.506	0.612992	
DriverAge3	-1.272e-07	8.668e-07	-0.147	0.883359	
...					

Exemple 1 : Régression Poisson polynomiale

```
MSE_Poisson_poly <- mean((predict(modele4, type = "response")  
                          - dataAGG$ClaimNumber)^2)
```

```
MSE_Poisson_poly
```

```
[1] 11.45383
```

Régression polynomiale

- Dans certains cas, les termes sont « ordonnés » : x, x^2, x^3, \dots, x^K .
- Il suffit alors d'ajuster et de calculer le tMSE pour les modèles

$$Y = \beta_0 + \beta_1 X$$

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2$$

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$$

...

et d'arrêter lorsqu'un certain critère est rencontré.

Transformation des variables explicatives

- Si les variables explicatives ne sont pas « ordonnées » (ce qui est généralement le cas), la procédure peut être beaucoup plus longue.
- En présence de p variables explicatives, on a 2^p modèles à ajuster et à tester.
- Avec $p = 5$, on a 32 modèles à tester, avec $p = 12$, on a 4 096 modèles à tester, avec $p = 100$, on a 1.26×10^{30} modèles à tester, etc.
- Lorsque le nombre de modèles est trop élevé, on doit utiliser une approche séquentielle (*stepwise approach*) qui considère uniquement un sous-ensemble des 2^p modèles. Cette approche est similaire à celle utilisée avec les critères AIC et BIC.

Réduction de la variance

- On a vu que l'erreur quadratique moyenne de validation pouvait se décomposer en trois termes : **(1)** variance de la prédiction, **(2)** biais de la prédiction au carré et **(3)** l'erreur stochastique (irréductible).
- Un grand nombre de variables explicatives dans un modèle vient augmenter la **flexibilité** du modèle, ce qui peut cause du **surajustement** (ou surapprentissage) et **augmenter** la variance de la prédiction.
- Sélectionner correctement les variables → diminuer la variabilité des prédictions → diminuer l'erreur quadratique moyenne de validation.
- Il existe d'autres méthodes que la sélection de variables permettant la réduction de la variance.

Régression Ridge

- Il s'agit d'un modèle qui ajoute une contrainte aux valeurs que peuvent prendre les paramètres (β).
- Cela permet de réduire la variance des paramètres et donc, la variance des prédictions.
- En contrepartie, cela va causer une augmentation du biais : il faut que la diminution de la variance soit plus importante que l'augmentation du biais pour obtenir de meilleures estimations.

Régression Ridge

À la base, il s'agit d'un modèle de régression linéaire

$$Y_i = \beta_0 + \sum_{j=1}^p \beta_j X_{ij} + \epsilon_i,$$

mais pour lequel les estimateurs sont donnés par

$$\hat{\beta}^R = \arg \min_{\beta_0, \dots, \beta_p} \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2,$$

pour une valeur de $\lambda > 0$.

Régression Ridge

Sous forme matricielle, on a (pour simplifier la théorie, on suppose que les variables explicatives sont centrées et réduites et que la variable réponse est centrée)

$$\mathbb{E}[\mathbf{Y}] = \mathbf{X}\boldsymbol{\beta}$$

où les coefficients sont donnés par

$$\begin{aligned}\hat{\boldsymbol{\beta}}^R &= \arg \min_{\boldsymbol{\beta}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \boldsymbol{\beta} \\ &= \left(\mathbf{X}^{**T} \mathbf{X}^{**} + \lambda \mathbf{I}_p \right)^{-1} \mathbf{X}^{**T} \mathbf{Y},\end{aligned}$$

où \mathbf{I}_p est une matrice identité ($p \times p$) et

$$\mathbf{X}^{**} = \begin{bmatrix} X_{11} & \dots & X_{1p} \\ \dots & \ddots & \dots \\ X_{n1} & \dots & X_{np} \end{bmatrix}.$$

Régression Ridge

- Le terme supplémentaire

$$\lambda \sum_{j=1} \beta_j^2$$

permet d'ajouter une pénalité liée à la valeur des paramètres.

- Ainsi, les estimations $\hat{\beta}_1^R, \dots, \hat{\beta}_p^R$ seront plus petites (en valeur absolue) que celles obtenus avec le modèle classique $\hat{\beta}_1, \dots, \hat{\beta}_p$.
- La contrainte supplémentaire peut affecter un ou plusieurs des paramètres du modèle mais **ne doit pas** toucher l'ordonnée à l'origine.

Régression Ridge

- L'hyperparamètre λ permet de contrôler l'intensité de la contrainte appliquée : plus la valeur de λ sera élevée, plus la pénalité sera importante et plus les estimations obtenues seront près de 0.
- Si on choisit $\lambda = 0$, alors $\hat{\beta}_j^R = \hat{\beta}_j, j = 1, \dots, p$.
- On utilise la validation croisée pour déterminer la valeur de λ qui minimise l'erreur quadratique moyenne de validation (vMSE).

Régression Ridge

De façon équivalente, on peut obtenir les paramètres du modèle de régression Ridge en résolvant

$$\hat{\beta}^R = \arg \min_{\beta_0, \dots, \beta_p} \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2$$

sous la contrainte que $\sum_{j=1}^p \beta_j^2 \leq s$, pour un hyperparamètre $s > 0$. En présence d'une matrice de schéma (\mathbf{X}) orthonormale, on peut vérifier que

$$\hat{\beta}^R = \frac{\hat{\beta}}{1 + \lambda}$$

Exemple 2 : Multicolinéarité

```
### Création d'une base avec  $\beta_1 = 1$  et  $\beta_2 = 1$   
set.seed(100)  
x1 <- rnorm(20)  
x2 <- rnorm(20, mean=x1, sd=.01)  
x11 <- (x1 - mean(x1))/sd(x1)  
x22 <- (x2 - mean(x2))/sd(x2)  
y <- rnorm(20, mean = 3 + x11 + x22)  
yy <- y - mean(y)
```

Exemple 2 : Multicolinéarité

```
M1 <- lm(yy ~ x11 + x22)
```

```
round(coef(M1), 3)
```

(Intercept)	x11	x22
0.000	4.931	-2.766

```
X <- matrix(c(rep(1,20), x11, x22), ncol = 3)
```

```
Y <- matrix(yy, ncol = 1)
```

```
round(t(solve(t(X) %*% X) %*% t(X) %*% Y), 3)
```

(Intercept)	x11	x22
0.000	4.931	-2.766

Exemple 2 : Multicolinéarité

```
### Fonction lm.ridge de la librairie MASS
library(MASS)
M2 <- lm.ridge(yy ~ x11 + x22, lambda = 1)
round(coef(M2), 3)
      x11    x22
0.000 1.059 1.052

lambda <- 1
Z <- X[,2:3]
round(t(solve(t(Z) %*% Z + lambda*diag(2))
      %*% t(Z) %*% Y), 3)
      x11    x22
0.000 1.058 1.051
```

Régression Ridge

On peut démontrer que la variance des estimateurs est donnée par

$$\text{Var} \left[\hat{\beta} \right] = \sigma^2 \mathbf{W} \mathbf{X}^{**T} \mathbf{X}^{**} \mathbf{W},$$

où $\mathbf{W} = \left(\mathbf{X}^{**T} \mathbf{X}^{**} + \lambda \mathbf{I}_p \right)^{-1}$. On peut également vérifier que le biais est donné par $-\lambda \mathbf{W} \beta$.

Exemple 1 : Régression Ridge

```
### Fonction lm.ridge de la librairie MASS
### Attention: l'échelle des paramètres n'est pas l'originale
library(MASS)
modele5 <- lm.ridge(ClaimNumber ~ DriverAge + LicenceAge
                    + VehAge + offset(Exposure),
                    data = dataAGG, lambda = 0:4)

modele5
```

		DriverAge	LicenceAge	VehAge
0	-463.0799	0.6211036	11.77947	6.158403
1	-463.0499	0.6211080	11.77846	6.157810
2	-463.0199	0.6211123	11.77745	6.157217
3	-462.9899	0.6211166	11.77645	6.156625
4	-462.9600	0.6211209	11.77544	6.156032

Exemple 1 : Régression Ridge

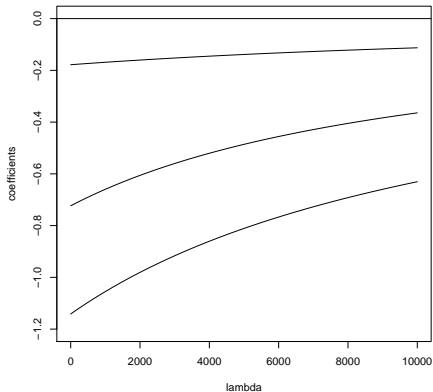


FIGURE – Évolution des valeurs des paramètres en fonction de la valeur de λ .

Exemple 1 : Régression Ridge

```
### Fonction glmnet de la librairie du même nom
```

```
library(glmnet)  
head(dataAGG, 3)
```

	DriverAge	LicenceAge	VehAge	Exposure	ClaimNumber
1	39	18	3	1356.402	192
2	35	18	3	1243.948	172
6	40	18	3	1263.537	165

```
X <- as.matrix(dataAGG[, (1:3)])  
Y <- as.matrix(dataAGG[, 5])  
OS <- as.matrix(dataAGG[, 4])  
fit <- glmnet(X, Y, alpha = 0, nlambda = 5, offset = OS)
```

Exemple 1 : Régression Ridge

```
round(fit$lambda, 2)
```

```
68895.54  6889.55   688.96    68.90     6.89
```

```
round(fit$beta, 2)
```

	s0	s1	s2	s3	s4
DriverAge	0	0.03	0.23	0.58	0.62
LicenceAge	0	0.28	2.26	8.24	11.29
VehAge	0	0.12	1.01	4.12	5.87

```
plot(fit, xvar = "lambda")
```

Exemple 1 : Régression Ridge

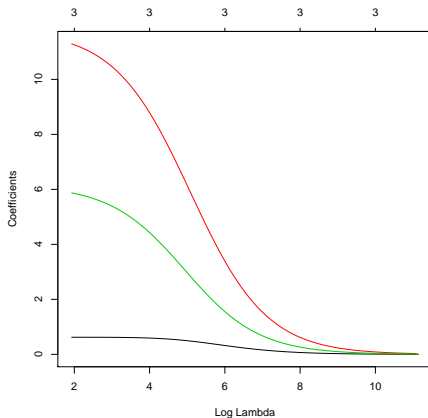


FIGURE – Évolution des valeurs des paramètres en fonction de la valeur de $\ln(\lambda)$.

Exemple 1 : Régression Ridge

```
set.seed(1)
cv.fit <- cv.glmnet(X, Y, nfolds = 10, alpha = 0,
                   offset = OS)
cv.fit$lambda.min
7.561278

coef(cv.fit, s = "lambda.min")

(Intercept) -447.1411740
DriverAge    0.6221213
LicenceAge   11.2457049
VehAge       5.8451944

plot(cv.fit)
```

Exemple 1 : Régression Ridge

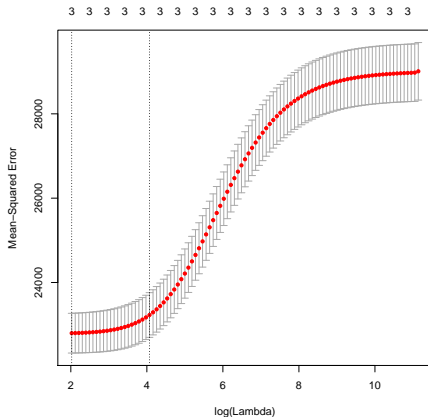


FIGURE – Évolution de l'erreur quadratique moyenne en fonction de la valeur de $\ln(\lambda)$.

Exemple 1 : Régression Ridge

```
set.seed(1)
cv.fit <- cv.glmnet(X, Y, nfolds = 10, alpha = 0,
                    family = "poisson", offset = log(OS))

coef(cv.fit, s = "lambda.min")

(Intercept) -2.556832848
DriverAge    -0.015202477
LicenceAge   0.049053574
VehAge       -0.007563799
```

Exemple 1 : Régression Ridge

```
MSE_RIDGE <- mean((predict(cv.fit, s = "lambda.min",  
                        newx = X, newoffset = log(OS),  
                        type = "response")  
                    - dataAGG$ClaimNumber)^2)
```

```
MSE_RIDGE  
12.42184
```


Exemple 1 : Régression Ridge

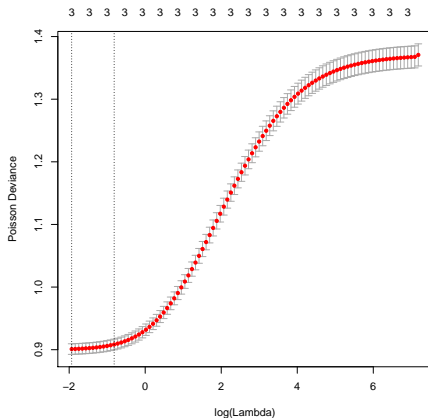


FIGURE – Évolution de la déviance Poisson en fonction de la valeur de $\ln(\lambda)$.

Régression Ridge

- + L'obtention du modèle final est beaucoup plus rapide que de tester tous les 2^p modèles ou d'utiliser une approche *stepwise*.
- + La variance des prédictions sera réduite.
 - Ne fait pas de sélection de variables : même si on choisit une grande valeur pour λ , toutes les variables explicatives sont conservées dans le modèle.
 - L'interprétation peut être complexe à faire.

Régression Lasso

Il s'agit d'un modèle de régression linéaire

$$Y_i = \beta_0 + \sum_{j=1}^p \beta_j X_{ij} + \epsilon_i,$$

mais pour lequel les estimateurs sont donnés par

$$\hat{\beta}^L = \arg \min_{\beta_0, \dots, \beta_p} \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j|,$$

pour une valeur de l'hyperparamètre $\lambda > 0$ obtenue par validation croisée.

Régression Lasso

- L'effet de la régression Lasso est similaire à celui de la régression Ridge : les paramètres prennent de plus petites valeurs (en valeur absolue).
- La régression Lasso **force** certains paramètres à prendre exactement la valeur 0 si la valeur de λ est assez élevée : sélection de variables.
- Le modèle final obtenu est plus facile à interpréter.

Régression Lasso

De façon équivalente, on peut obtenir les paramètres du modèle de régression Lasso en résolvant

$$\hat{\beta}^L = \arg \min_{\beta_0, \dots, \beta_p} \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2$$

sous la contrainte que $\sum_{j=1}^p |\beta_j| \leq s$, pour un hyperparamètre $s > 0$.

Régression Lasso

```
fit <- glmnet(X, Y, alpha = 1, nlambda = 5, offset = 0S)
round(fit$lambda, 2)
68.90  6.89  0.69  0.07  0.01
```

```
round(fit$beta, 2)
```

	s0	s1	s2	s3	s4
DriverAge	.	0.19	0.58	0.62	0.62
LicenceAge	.	10.71	11.67	11.77	11.78
VehAge	.	4.84	6.03	6.15	6.16

```
plot(fit, xvar = "lambda")
```

Exemple 1 : Régression Lasso

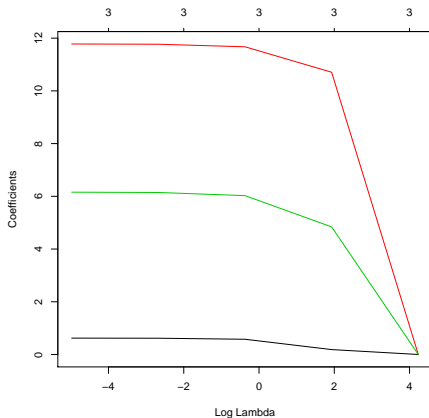


FIGURE – Évolution de la valeur des paramètres en fonction de la valeur de $\ln(\lambda)$.

Régression Lasso

```
set.seed(1)
cv.fit <- cv.glmnet(X, Y, nfolds = 10, alpha = 1,
                   offset = OS)
cv.fit$lambda.min
0.3428948

coef(cv.fit, s = "lambda.min")

(Intercept) -460.0745341
DriverAge    0.5994006
LicenceAge   11.7260140
VehAge       6.0927586

plot(cv.fit)
```


Exemple 1 : Régression Lasso

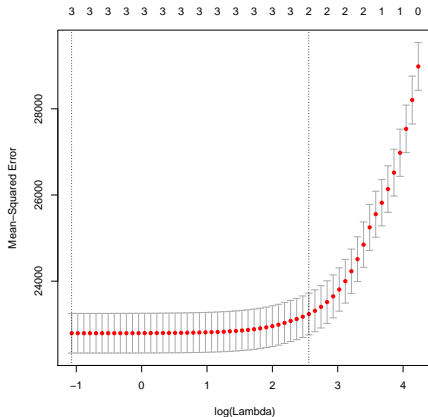


FIGURE – Évolution de l'erreur quadratique moyenne en fonction de la valeur de $\ln(\lambda)$.

Régression Lasso

```
set.seed(1)
cv.fit <- cv.glmnet(X, Y, nfolds = 10, alpha = 1, family = "po
                  offset = log(OS))

coef(cv.fit, s = "lambda.min")

(Intercept) -2.588302202
DriverAge    -0.015652738
LicenceAge   0.051645133
VehAge       -0.007618157
```

Régression Lasso

```
MSE_LASSO <- mean((predict(cv.fit, s = "lambda.min",  
                        newx = X, newoffset = log(OS),  
                        type = "response")  
                    - dataAGG$ClaimNumber)^2)
```

MSE_LASSO

12.60702

Exemple 1 : Régression Lasso

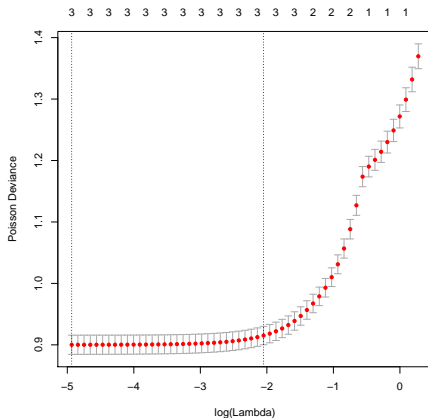


FIGURE – Évolution de la déviance Poisson en fonction de la valeur de $\ln(\lambda)$.

Régression

Les estimateurs sont obtenus en résolvant

$$\hat{\beta}^L = \arg \min_{\beta_0, \dots, \beta_p} \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2$$

sous la contrainte

- $\sum_{j=1}^p \beta_j^2 \leq s$, $s > 0$, pour la régression Ridge ;
- $\sum_{j=1}^p |\beta_j| \leq s$, $s > 0$, pour la régression Lasso ; et
- $\sum_{j=1}^p \mathbb{I}_{\beta_j \neq 0} \leq s$, $s \in \mathbb{N}$, pour la régression classique avec sélection de variables.

Régression

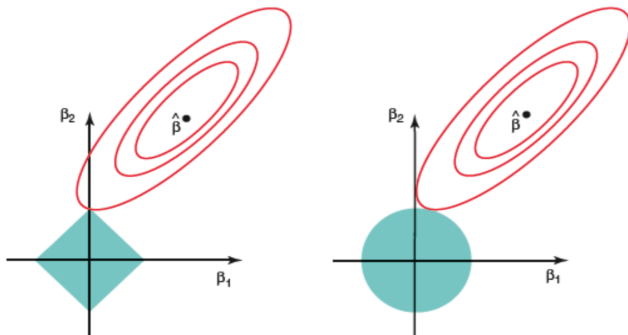


FIGURE – Deux paramètres : régression Lasso (gauche) et régression Ridge (droite). Image issue de *An Introduction to Statistical Learning : with Applications in R*.

Exemple 1 : Résultats

Modèle	Régularisation	MSE	vMSE
10 plus proches voisins	Non	10.64	14.67
Régression linéaire	Non	22 769.49	23 588.88
Régression Poisson	Non	12.62	13.06
Régression Poisson polynomiale	Non	11.45	12.32
Régression Poisson Ridge	Oui	12.42	12.89
Régression Poisson Lasso	Oui	12.61	13.04