

Régression en grande dimension

Julien JACQUES

02 février, 2023

Les data

En régression, on cherche à **expliquer et prédire une variable quantitative**

$$Y = (y_1, \dots, y_n)^t$$

à partir de p variables explicatives, que l'on supposera quantitative (*les variables catégorielles seront binarisées*) :

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

Le modèle linéaire

Le modèle de régression linéaire s'écrit

$$y_i = \beta_0 + \sum_{j=1}^p \beta_j x_{ij} + \epsilon_i$$

où :

- ▶ β_0 est l'intercept
- ▶ β_j sont les coefficients de régression
- ▶ ϵ_i est le résidu, la part de y_i que l'on n'arrive pas à expliquer à partir des x_{ij}
- ▶ les ϵ_i sont supposés i.i.d (indépendants et identiquement distribués $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$)

Estimation

L'estimation de $\beta = (\beta_0, \beta_1, \dots, \beta_p)$ peut être réalisée par **maximum de vraisemblance** (sous l'hypothèse $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$) ou de façon équivalente par **moindre carrés**:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n (y_i - (\beta_0 + \sum_{j=1}^p \beta_j x_{ij}))^2 = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n \epsilon_i^2$$

La solution est alors (sous la contrainte $p < n$, sinon il n'existe pas de solution unique) :

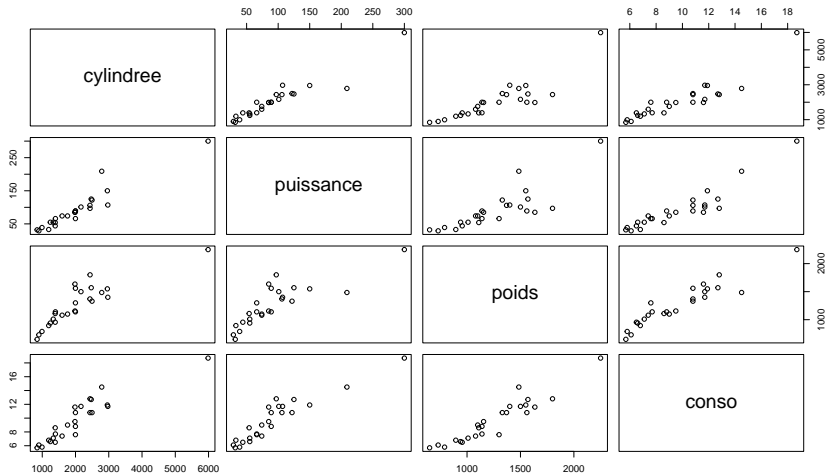
$$\hat{\beta} = (X^t X)^{-1} X^t Y$$

où

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

Exemple (data)

```
auto=read.table('vehicules.txt',header=TRUE)  
plot(auto[,2:5])
```



Les 3 variables semblent corrélées positivement à la consommation.

Exemple (modèle)

```
modele=lm(conso~cylindree+puissance+poids,data=auto)
```

```
summary(modele)
##
## Call:
## lm(formula = conso ~ cylindree + puissance + poids, data = auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.5120 -0.4868  0.1639  0.5513  1.2886
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.2998368   0.6140732   2.117  0.046391 *
## cylindree    -0.0004740   0.0004972  -0.953  0.351305
## puissance     0.0302691   0.0074708   4.052  0.000574 ***
## poids        0.0052011   0.0008067   6.447  2.17e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7006 on 21 degrees of freedom
## Multiple R-squared:  0.9567, Adjusted R-squared:  0.9505
## F-statistic: 154.7 on 3 and 21 DF,  p-value: 1.787e-14
```

La régression indique que la variable cylindree n'est pas significative dans le modèle.

Exemple (modèle)

Par contre, en enlevant la variable puissance elle devient significative. La corrélation entre les variables explicatives brouille le message. . .

```
#auto=as.data.frame(scale(auto[,2:5]))
modele=lm(conso~cylindree+poids,data=auto)
summary(modele)
```

```
##
## Call:
## lm(formula = conso ~ cylindree + poids, data = auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.23095 -0.44033  0.01671  0.47117  2.77739
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.7307901  0.7795958   0.937  0.35873
## cylindree    0.0011776  0.0003713   3.171  0.00442 **
## poids       0.0051903  0.0010520   4.934 6.19e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9137 on 22 degrees of freedom
## Multiple R-squared:  0.9229, Adjusted R-squared:  0.9158
## F-statistic: 131.6 on 2 and 22 DF,  p-value: 5.757e-13
```

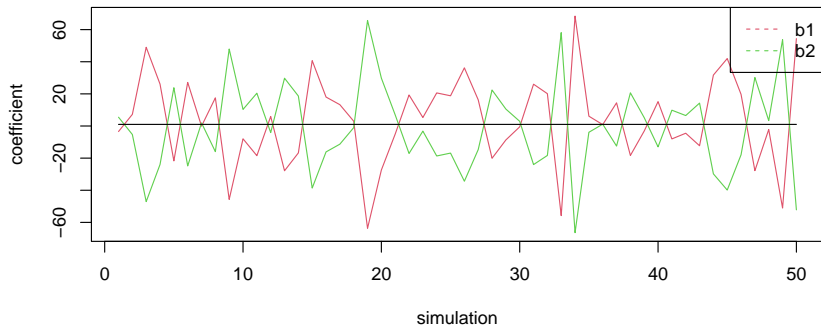
Influence de la corrélation

Soit le modèle

$$y_i = 3 + x_{1i} + x_{2i} + \epsilon_i$$

avec (x_1, x_2) très corrélées : $x_1 \sim \mathcal{N}(0, 1)$, $x_2|x_1 \sim \mathcal{N}(x_1, .01^2)$.

Sur 50 simulations on obtient les estimations de β_1 et β_2 suivantes :



Influence de la corrélation

L'estimation de la variance de l'estimateur $\hat{\beta}_j$ de β_j est :

$$\widehat{Var}(\hat{\beta}_j) = \frac{s^2}{(n-1)\widehat{Var}(X_j)} \frac{1}{1-R_j^2}$$

où :

- ▶ R_j^2 est le coefficient de détermination de la variable X_j sur tous les autres prédicteurs X_ℓ ($\ell \neq j$)
- ▶ $\frac{1}{1-R_j^2}$ est le **VIF (facteur d'inflation de la variance)**
- ▶ un VIF supérieur à 10 indiquera un problème de colinéarité
- ▶ $s^2 = \frac{1}{n-p} \sum_{i=1}^p \hat{\epsilon}_i^2$

Ainsi :

- ▶ si les variables sont fortement liées linéairement, R_j^2 est proche de 1, donc $\widehat{Var}(\hat{\beta}_j)$ explose
- ▶ si $p < n$ mais $p \simeq n$, s^2 explose

Influence de la corrélation (VIF)

Avec la librarire car

```
modele=lm(conso~cylindree+puissance+poids,data=auto)
library(car)
vif(modele)
```

```
## cylindree puissance      poids
## 12.992577  9.696485  4.260911
```

Influence de la corrélation (VIF)

Avec la librairie mctest

```
modele=lm(conso~cylindree+puissance+poids,data=auto)
library(mctest)
imcdiag(modele)
```

Régression en présence de corrélation et/ou en grande dimension

Trois solutions sont possibles :

- ▶ sélectionner des variables (forward, stepwise. . .) :
 - ▶ interprétation aisée mais lourd en calcul, parfois grande variance
- ▶ créer de nouvelles variables non corrélées synthétisant l'information : régression sur composantes principales (PCR), régression PLS
 - ▶ bonne capacité de prédiction mais faible en interprétation
- ▶ contraindre l'estimation des paramètres : régressions pénalisées (ridge, LASSO, ElasticNet. . .)
 - ▶ bonne capacité de prédiction et interprétation aisée

Choix de modèles en régression

Décomposition biais-variance

Pour un nouvel x^* , la prédiction par le modèle linéaire s'écrit

$$\hat{y}_i^* = \beta_0 + \sum_{j=1}^p \beta_j x_{ij}^*$$

La qualité de la prédiction $E[(y_i^* - \hat{y}_i^*)^2]$ peut se décomposer en

$$E[(y_i^* - \hat{y}_i^*)^2] = \sigma^2 + \underbrace{(E[\hat{y}_i^*] - y_i^*)^2}_{\text{biais}} + \underbrace{E[(E[\hat{y}_i^*] - \hat{y}_i^*)^2]}_{\text{variance}}$$

où

- ▶ σ^2 : erreur incompressible. Variance de la cible Y
- ▶ $E[\hat{y}_i^*] - y_i^*$: **biais** : écart entre l'espérance de la prédiction et la vraie valeur. Indique les insuffisances intrinsèques du modèle (variables explicatives manquantes, ou forme de la relation non captée, etc.).
- ▶ $E[(\hat{y}_i^* - y_i^*)^2]$: **variance** : dispersion de la prédiction autour de sa propre espérance. Témoigne de l'instabilité du modèle, sa dépendance aux fluctuations de l'échantillon d'apprentissage.

Analyse de variance de la régression et R^2

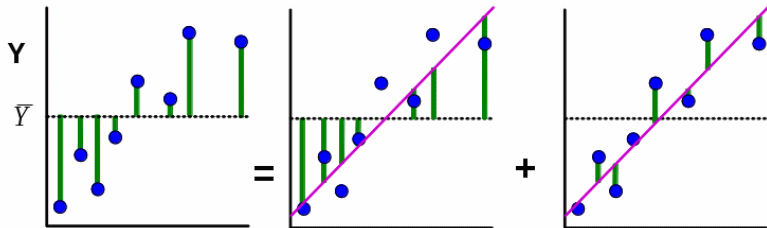


Figure 1: Source <http://www.wikistat.fr>

$$\begin{array}{lll} \text{SST} & = & \text{SSReg} \\ \text{(variance} & & \text{(variance} \\ \text{totale)} & & \text{expliquée)} \\ & + & \text{SSR} \\ & & \text{(variance} \\ & & \text{résiduelle)} \end{array}$$

Analyse de variance de la régression et R^2

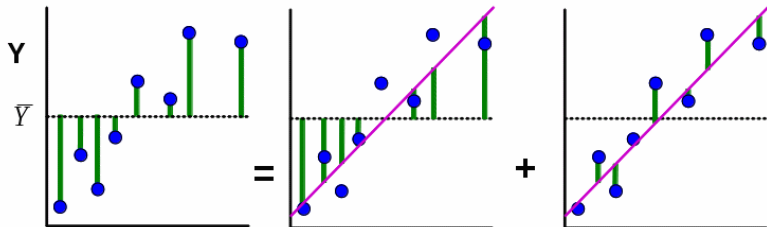


Figure 1: Source <http://www.wikistat.fr>

$$\begin{array}{lll} \text{SST} & = & \text{SSReg} \quad + \quad \text{SSR} \\ \text{(variance totale)} & & \text{(variance expliquée)} \quad \quad \text{(variance résiduelle)} \end{array}$$

Le **coefficient de détermination** indique la qualité de la régression :

$$R^2 = \frac{SSReg}{SST}$$

Coefficients de détermination

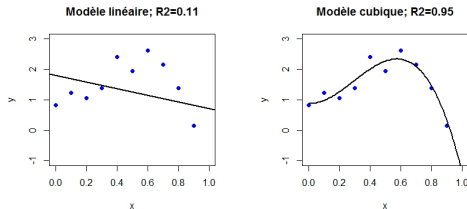


FIGURE 1 – Régression polynomiale : ajustement par, à gauche, $y = \beta_0 + \beta_1 x + \epsilon$, et à droite, $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$

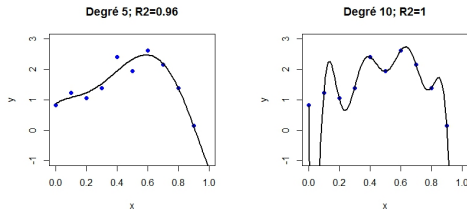


FIGURE 2 – Régression polynomiale : ajustement par, à gauche, $y = \beta_0 + \beta_1 x + \dots + \beta_5 x^5 + \epsilon$, et à droite, $y = \beta_0 + \beta_1 x + \dots + \beta_{10} x^{10} + \epsilon$.

Choix de modèles en régression

Le R^2 n'est pas un bon indicateur pour comparer des modèles, il choisira toujours les modèles les plus complexes (ayant le plus de paramètres).

Nous utiliserons plutôt :

- ▶ la Root Mean Square Error (RMSE) : $\sqrt{\frac{1}{n} \sum_i (\hat{y}_i - y_i)^2}$
évaluée sur des données **indépendantes** :
 - ▶ sur un échantillon test : 1/3 des données disponibles sont mis de cotés pour pouvoir *tester* la qualité des modèles
 - ▶ par validation croisée Leave One Out (LOO) : \hat{y}_i est estimé en utilisant toutes les données sauf la i -ème
 - ▶ par validation croisée K-fold : les données disponibles sont découpées en K groupes, chacun des groupes servant tour à tour d'échantillon test

Choix de modèles en régression

Tout comme le R^2 , la vraisemblance \mathcal{L} n'est pas non plus un bon indicateur pour comparer des modèles, elle choisira toujours les modèles les plus complexes (ayant le plus de paramètres).

Nous utiliserons plutôt :

- ▶ des indicateurs de **vraisemblances pénalisées** :

- ▶ $BIC = -2 \ln \mathcal{L} + k \ln n$ où k est le nombre de paramètres du modèle
- ▶ $AIC = -2 \ln \mathcal{L} + 2k$

On pourra également utilisé le :

- ▶ R^2 ajusté : $R_{adj}^2 = 1 - \left(\frac{(1-R^2)(n-1)}{n-k-1} \right)$

Sélection de variables en régression

Sélection de variables en régression

Le nombre de sous-ensembles de variables parmi les p covariables est énorme $\sum_{j=0}^p C_p^j = 2^p$.

Par exemple pour $p = 10$:

```
nb=0
for (j in 0:10) nb=nb+ncol(combn(10,j))
cat('Nb de sous-ensembles = ',nb)
```

```
## Nb de sous-ensembles = 1024
```

Sélection de variables en régression

Le nombre de sous-ensembles de variables parmi les p covariables est énorme $\sum_{j=0}^p C_p^j = 2^p$.

Par exemple pour $p = 10$:

```
nb=0
for (j in 0:10) nb=nb+ncol(combn(10,j))
cat('Nb de sous-ensembles = ',nb)
```

```
## Nb de sous-ensembles = 1024
```

Il nous faut une stratégie pour n'explorer qu'un **nombre restreint** de ces sous-ensembles

Sélection de variables en régression

- ▶ Méthode ascendante : **forward**
 - ▶ on part du modèle constant sans covariable
 - ▶ on ajoute la covariable qui apporte le plus selon un certain critère (ex : plus forte augmentation du R^2 / diminution du AIC)
 - ▶ puis on ajoute une seconde variable selon le même critère, sans remettre en question celles déjà introduite,
 - ▶ etc jusqu'à ce que toutes les variables soient dans le modèle
- ▶ Méthode descendante : **backward**
 - ▶ on part du modèle complet (possible si $p < n$)
 - ▶ on enlève la variable qui apporte la plus faible diminution du R^2 / augmentation du AIC
 - ▶ puis on enlève une seconde variable selon le même critère, sans remettre en question celles déjà supprimées,
 - ▶ etc jusqu'à ce qu'il n'y ait plus de variable dans le modèle
- ▶ on choisira parmi les p modèles avec les critères habituels :
 - ▶ RMSE par CV ou apprentissage/test
 - ▶ AIC

Sélection de variables en régression sous R

- ▶ Méthode **stepwise / both**

- ▶ recherche ascendante ou descendante
- ▶ on ajoute un test de significativité des variables incluses dans le modèle à chaque étape, et suppression de celles non significatives
- ▶ on s'arrête quand on ne peut plus ni supprimer ni ajouter de variables

Le modèle à conserver est le dernier obtenu.

Sélection de variables en régression

```
modele=lm(conso~cylindree+puissance+poids,data=auto)
library(MASS)
stepAIC(modele,direction='both',trace=FALSE)

##
## Call:
## lm(formula = conso ~ puissance + poids, data = auto)
##
## Coefficients:
## (Intercept)      puissance          poids
##      1.359305       0.024431       0.004814
```

Rq : on peut aussi utiliser la fonction *step*

Sélection de variables en régression

Les algorithmes de sélection de variables sont particulièrement intéressants lorsqu'on a un nombre réduit de variables, sinon le temps de calcul peut-être long, et ils peuvent être relativement sensibles aux fluctuations d'échantillonnage.

Exercice

Estimer un modèle de régression avec sélection de variables sur les données **vehicles** du package **plsdepot**

```
library(plsdepot)
data(vehicles)
str(vehicles)
```

```
## 'data.frame':    30 obs. of  16 variables:
## $ diesel      : int  0 0 0 0 0 0 0 0 0 0 0 ...
## $ turbo       : int  0 0 0 0 1 0 0 0 0 0 0 ...
## $ two.doors   : int  1 0 1 0 1 0 1 0 0 0 0 ...
## $ hatchback   : int  1 0 0 0 1 1 1 0 0 0 0 ...
## $ wheel.base  : num  94.5 105.8 101.2 94.5 93.7 ...
## $ length      : num  171 193 177 159 157 ...
## $ width       : num  65.5 71.4 64.8 63.6 63.8 63.8 64 65
## $ height      : num  52.4 55.7 54.3 52 50.8 50.6 52.6 54
## $ curb.weight : int  2823 2844 2395 1909 2128 1967 1956
## $ eng.size    : int  152 136 108 90 98 90 92 110 111 258
## $ horsepower  : int  154 110 101 70 102 68 76 86 78 176
```

Régression PCR

Régression PCR

La régression sur composantes principales (PCR) consiste à :

- ▶ réaliser une ACP sur les prédicteurs X
- ▶ réaliser une régression sur les composantes principales issues de l'ACP

Le choix du nombre de composantes principales pourra se faire :

- ▶ soit par des techniques classiques d'ACP,
- ▶ ou mieux en fonction de la qualité du modèle de régression obtenu

Régression PCR sous R

```
library(pls)
modele=pcr(conso~cylindree+puissance+poids,data=auto,
           validation='L00')
summary(modele)
```

```
## Data:      X dimension: 25 3
## Y dimension: 25 1
## Fit method: svdpc
## Number of components considered: 3
##
## VALIDATION: RMSEP
## Cross-validated using 25 leave-one-out segments.
##      (Intercept)  1 comps  2 comps  3 comps
## CV              3.215    1.509    1.050    0.8866
## adjCV           3.215    1.501    1.045    0.8806
##
## TRAINING: % variance explained
##      1 comps  2 comps  3 comps
## X        97.61    99.97    100.00
## conso     86.23    92.30    95.67
```

Exercice

Estimer un modèle de régression (avec sélection de variables) pour le prix sur les données **vehicles** du package **plsdepot**

```
library(plsdepot)
data(vehicles)
str(vehicles)
```

```
## 'data.frame':    30 obs. of  16 variables:
## $ diesel      : int  0 0 0 0 0 0 0 0 0 0 0 ...
## $ turbo       : int  0 0 0 0 1 0 0 0 0 0 0 ...
## $ two.doors    : int  1 0 1 0 1 0 1 0 0 0 0 ...
## $ hatchback    : int  1 0 0 0 1 1 1 0 0 0 0 ...
## $ wheel.base  : num  94.5 105.8 101.2 94.5 93.7 ...
## $ length      : num  171 193 177 159 157 ...
## $ width       : num  65.5 71.4 64.8 63.6 63.8 63.8 64 65
## $ height      : num  52.4 55.7 54.3 52 50.8 50.6 52.6 54
## $ curb.weight : int  2823 2844 2395 1909 2128 1967 1956
## $ eng.size    : int  152 136 108 90 98 90 92 110 111 258
## $ horsepower  : int  154 110 101 70 102 68 76 86 78 176
```

Régression PLS

Régression PLS

La régression sur composantes principales permet de solutionner à la fois les problèmes de corrélation entre covariables et les problèmes de dimension.

Par contre rien n'assure que les composantes principales construites, retraçant un maximum de variabilité contenue dans X , soient intéressantes pour prédire Y .

La **régression PLS (Partial Least Square)** consiste à construire de nouvelles variables (composante PLS) avec comme critère d'être fortement corrélées à Y .

Régression PLS

Recherche des composantes PLS (algorithme NIPALS)

- ▶ normalisation des covariables (moyenne 0 et variance 1)
- ▶ 1ère composante PLS : on recherche $Z_1 = \sum_{j=1}^p \rho_{j1} X_j$ qui maximise $\text{cov}(Z_1, Y)$. La solution est $\rho_j \propto \text{corr}(Y, X_j)$
- ▶ on estime $Y = \beta_1 Z_1 + \epsilon_1$
- ▶ 2ème composante PLS : on recherche $Z_2 = \sum_{j=1}^p \rho_{j2} X_j$ qui maximise $\text{cov}(\epsilon_1, Z_2)$
- ▶ on estime $Y = \beta_1 Z_1 + \beta_2 Z_2 + \epsilon_2$
- ▶ et ainsi de suite

Sélection du nombre de composantes

- ▶ on s'arrête généralement avant que le critère $Q_h^2 = 1 - \frac{PRESS_h}{RSS_{h-1}} < 0.0975$ (!) où
 - ▶ $RSS_h = \sum_{i=1}^n (y_i - \sum_{j=1}^h \rho_{jh} z_{ih})^2$
 - ▶ $PRESS_h$ est la même chose que RSS_h mais évalué par validation croisée leave-one-out

Régression PLS

Importance de la variable X_j dans le modèle des m composantes :

$$VIP_{jm} = \sqrt{\frac{p}{\sum_{h=1}^m R^2(Y, Z_h)} \sum_{h=1}^m R^2(Y, Z_h) \rho_{jh}^2}$$

Régression PLS sous R avec package *pls*

```
library(pls)
modele=plsr(conso~cylindree+puissance+poids,data=auto,
            validation='L00')
summary(modele)
```

```
## Data:      X dimension: 25 3
## Y dimension: 25 1
## Fit method: kernelpls
## Number of components considered: 3
##
## VALIDATION: RMSEP
## Cross-validated using 25 leave-one-out segments.
##      (Intercept)  1 comps  2 comps  3 comps
## CV           3.215   1.491   1.043   0.8866
## adjCV        3.215   1.482   1.038   0.8806
##
## TRAINING: % variance explained
##      1 comps  2 comps  3 comps
## X       97.60   99.97  100.00
## conso   86.52   92.39   95.67
```

Exercice

Estimer un modèle de régression avec sélection de variables sur les données **vehicles** du package **plsdepot**

```
library(plsdepot)
data(vehicles)
str(vehicles)
```

```
## 'data.frame':    30 obs. of  16 variables:
## $ diesel      : int  0 0 0 0 0 0 0 0 0 0 0 ...
## $ turbo       : int  0 0 0 0 1 0 0 0 0 0 0 ...
## $ two.doors    : int  1 0 1 0 1 0 1 0 0 0 0 ...
## $ hatchback    : int  1 0 0 0 1 1 1 0 0 0 0 ...
## $ wheel.base  : num  94.5 105.8 101.2 94.5 93.7 ...
## $ length      : num  171 193 177 159 157 ...
## $ width       : num  65.5 71.4 64.8 63.6 63.8 63.8 64 65
## $ height      : num  52.4 55.7 54.3 52 50.8 50.6 52.6 54
## $ curb.weight : int  2823 2844 2395 1909 2128 1967 1956
## $ eng.size    : int  152 136 108 90 98 90 92 110 111 258
## $ horsepower  : int  154 110 101 70 102 68 76 86 78 176
```

Régression PLS sous R avec package *plsdepot*

Le package **plsdepot** propose également des sorties graphiques type ACP. Examinons ce que cela donne sur les données **vehicles**

Il faut mettre la variable à prédire (prix) dans la dernière colonne

```
cars = vehicles[,c(1:12,14:16,13)]
```

Choisissons dans un premier temps un large nombre de composantes (5), puis nous examinerons les Q^2_h pour sélectionner le bon nombre

```
tmp=plsreg1(cars[,1:15],cars[,16],drop=F,comps=5,crosval=T,  
print(tmp$Q2)
```

##		PRESS	RSS	Q2	LimQ2	Q2cum
##	1	11.540084	29.000000	0.6020661	0.0975	0.6020661
##	2	7.529169	8.562281	0.1206585	0.0975	0.6500802
##	3	3.883748	5.166929	0.2483449	0.0975	0.7369810
##	4	2.903405	2.304000	-0.2601586	0.0975	0.6685543
##	5	2.600414	1.938508	-0.3414517	0.0975	0.5553816

On choisit 3 composantes car Q^2_L passe en dessous de la limite

Régression PLS sous R

Le modèle avec 3 composantes donne

```
pls1=plsreg1(cars[,1:15],cars[,16,drop=F],comps=3)
```

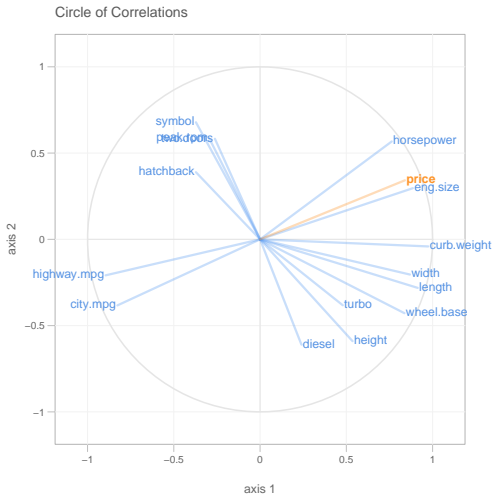
La sortie du modèle contient les objets suivants

```
##
## PLS Regression 1
## -----
## $x.scores      X-scores (T-components)
## $x.loads       X-loadings
## $y.scores      Y-scores (U-components)
## $y.loads       Y-loadings
## $cor.xyt       score correlations
## $raw.wgs       raw weights
## $mod.wgs       modified weights
## $std.coefs     standard coefficients
## $reg.coefs     regular coefficients
## $R2            R-squared
## $R2Xy          explained variance of X-y by T
## $y.pred        y-predicted
## $resid         residuals
## $T2            T2 hotelling
## $Q2            Q2 cross validation
## -----
```

Résultats de la PLS

Traçons le cercle des corrélations pour observer les liens entre variables initiales et composantes PLS :

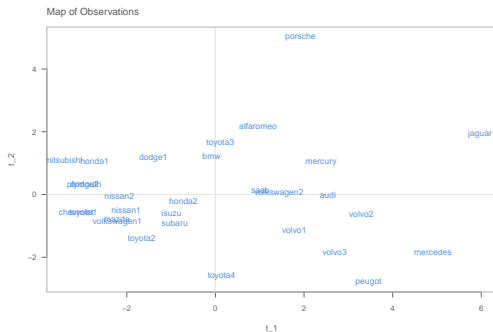
```
plot(pls1,what='variables',comps=c(1,2))
```



Résultats de la PLS

On peut représenter les observations également

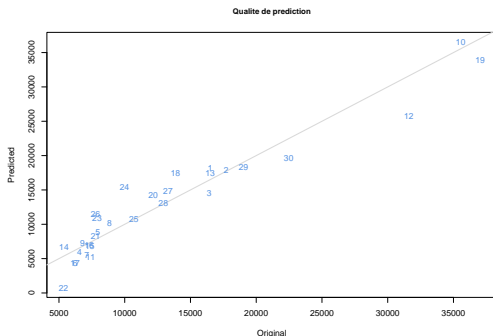
```
plot(pls1, what='observations', comps=c(1,2), show.names=T)
```



Résultats de la PLS

On peut également illustrer la qualité de prediction les valeurs prédites en fonction des valeurs réelles

```
plot(cars$price, pls1$y.pred, type='n', xlab='Original', ylab='Predicted',  
title('Qualite de prediction', cex.main = 0.9)  
abline(a = 0, b = 1, col = 'gray85', lwd = 2)  
text(cars$price, pls1$y.pred, col = '#5592e3')
```



Importance des variables en PLS

Les VIPs ne sont pas implémentés dans ce package, on peut les calculer à la main

Un $VIP > .8$ est généralement signe que la variable a une importance sur la prediction

##	[,1]	[,2]	[,3]
## diesel	0.2784769	0.2857481	0.8227674
## turbo	0.5073281	0.5693036	0.5412825
## two.doors	0.0913152	0.4670275	0.4650202
## hatchback	0.5526105	0.5117730	0.7878440
## wheel.base	0.9967898	1.0027557	0.9475462
## length	1.1619058	1.1211620	1.0633458
## width	1.2033004	1.1241463	1.0646373
## height	0.3698322	0.7443507	0.7294240
## curb.weight	1.4859406	1.3766264	1.3079641
## eng.size	1.6230848	1.5698906	1.5150749
## horsepower	1.4730322	1.4542874	1.3747565
## peak.rpm	0.1165452	0.5088424	0.4886079
## symbol	0.1537610	0.6316262	0.5975477
## city.mpg	1.2893546	1.1965327	1.2307053
## highway.mpg	1.3080983	1.2123358	1.2216591

Régression pénalisée

Régression pénalisée

En présence de corrélation entre covariables, ou lorsque p est proche de n (ce qui induit naturellement de la corrélation ou quasi-colinéarité), la variance de la prédiction est très grande.

Le principe de la régression pénalisée est de **contraindre les valeurs des paramètres** de régression β_j , qui peuvent prendre des valeurs erratique en présence de corrélation ou quasi co-linéarité, et ainsi **réduire la variance**, quitte à induire une légère **augmentation du biais**.

Régression ridge

La régression ridge (Hoerl & Kennard, 1970) consiste à contraindre les β_j à ne pas prendre de valeur trop grandes

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 \right\} \text{ subject to } \sum_{j=1}^p \beta_j^2 < t_\lambda$$

Ce problème s'écrit de façon équivalente :

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$

La solution de ce problème est

$$\hat{\beta}^{ridge} = (X^t X + \lambda I)^{-1} X^t Y$$

L'inverse instable de $X^t X$ est régularisée en ajoutant un terme constant sur la diagonale.

Propriétés de la régression ridge

Attention : on normalisera les prédictors (centrage réduction) pour éviter que des variables à forte variance n'aient trop d'influence.

Propriétés :

- ▶ les coefficients de régression sont *shrinkés* vers 0
- ▶ plus grand est λ , plus le shrinkage est important
- ▶ si les covariables sont orthogonales, on a $\hat{\beta}^{ridge} = \frac{1}{1+\lambda} \hat{\beta}^{OLS}$

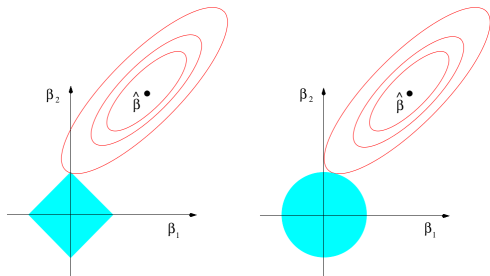
Le choix du paramètre de shrinkage λ pourra être fait par validation croisée

Régression LASSO

Le principe est le même que la régression ridge, sauf que l'on change la pénalisation L_2 par une pénalisation L_1 :

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 \right\} \text{ subject to } \sum_{j=1}^p |\beta_j| < t_\lambda$$

Contrairement à la norme L_2 , les *coins* de la norme L_1 tendent à obtenir des solutions avec un certains nombre de $\hat{\beta}_j$ nul: la régression LASSO permet de réaliser une **sélection de variables**



Régression LASSO

L'estimation des coefficients du LASSO nécessite un algorithme itératif, **Least Angle Regression** (LAR-LASSO), qui fonctionne de la façon suivante :

1. on commence avec tous les $\beta_j = 0$
2. on choisit la variable X_k la plus corrélée avec Y , et on fait évoluer petit à petit le coefficient β_k en direction du β_k^{OLS} , jusqu'à ce qu'une autre variable X_ℓ devienne plus corrélée avec les résidus de la régression de Y sur X_k
3. on fait alors évoluer à la fois β_k et β_ℓ en direction de leur solution OLS, ... et ainsi de suite
4. lorsqu'un coefficient β_j heurte 0, on enlève alors la variable correspondante des variables actives et on continue

On obtient alors des scénarios de solutions (LASSO path) à mesure que $\|\beta\|_1$ augmente.

Ainsi, toutes les solutions pour toutes valeurs de λ sont obtenues en une seule fois avec cet algorithme.

Exemple de LASSO path

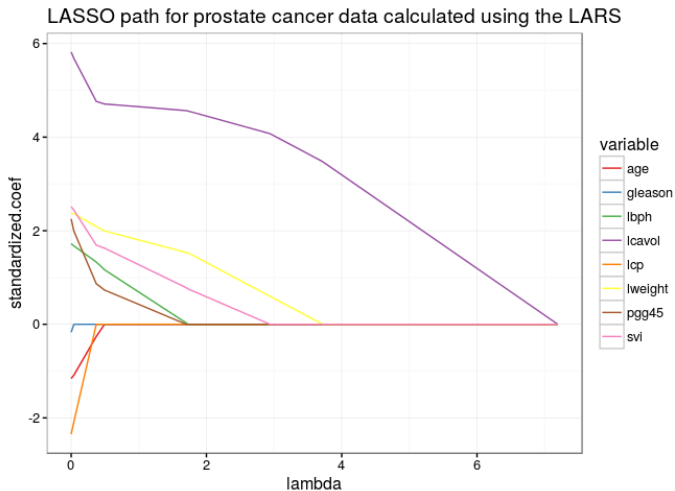


Figure 3: Exemple de chemin LASSO

Régression Elastic-Net

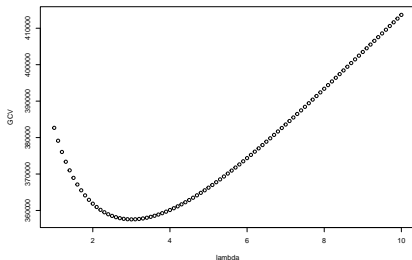
La régression **Elastic-Net** est un mélange entre une régression ridge et une régression LASSO

$$\hat{\beta}^{EN} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda_1 \sum_{j=1}^p |\beta_j| + \lambda_2 \sum_{j=1}^p \beta_j^2 \right\}$$

Régression ridge sous R

Effectuons une régression ridge sur les données cars

```
library(MASS)
model_ride <- lm.ride(price~.,data=cars,lambda=seq(1,10,0.1))
plot(seq(1,10,0.1),model_ride$GCV,xlab='lambda',ylab='GCV')
```



Le lambda minimisant l'erreur GCV est

```
print(model_ride$lambda[which.min(model_ride$GCV)])
```

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

Régression ridge sous R

Les valeurs des coefficients de régression sont alors

```
model_ride <- lm.ride(price~.,data=cars,lambda=3)
print(model_ride$coef)
```

```
##      diesel      turbo  two.doors  hatchback  wheel.base    length
## 1319.07847  322.30923  403.06638 -1924.13998   286.84161  -573.16314
##      width      height  curb.weight    eng.size  horsepower    peak.rpm
## 1196.52729 -1349.51087  1827.72607   3242.42975   2381.26372   1205.24651
##      symbol    city.mpg  highway.mpg
##    792.42010   -57.75644   -35.31223
```

que l'on peut comparer aux coefficients de la régression moindres carrés classiques

```
model_ols <- lm.ride(price~.,data=cars,lambda=0)
print(model_ols$coef)
```

```
##      diesel      turbo  two.doors  hatchback  wheel.base    length
##    538.0448   652.7103   253.2335  -3262.8088   204.0221  -5139.8544
##      width      height  curb.weight    eng.size  horsepower    peak.rpm
## 2349.8250  -972.9552   5813.5268   3124.1172   1595.1574   1086.3392
##      symbol    city.mpg  highway.mpg
##   1305.1569  -570.1281   839.4466
```

Prédiction avec ridge

Il n'existe pas de méthode *predict* associé à la fonction *lm.ridge*.

Il nous faut le construire la prédiction à la main (pour l'exemple on le fait sur toutes les données, en pratique il faudrait séparer échantillons d'apprentissage et de test):

- ▶ on commence par centrer et réduire les données test avec les moyennes et variances de l'échantillon d'apprentissage :

```
x.test=scale(cars[,-16],center=model_ridge$xm,  
              scale=model_ridge$scales)
```

Attention : dans cars le prix est la variable 16 (réorganisé précédemment pour PLS)

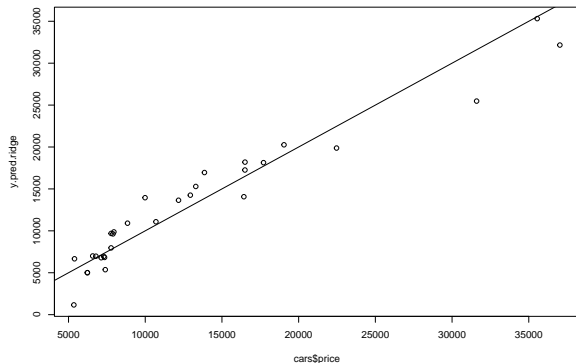
- ▶ puis on reconstruit la prédiction :

```
y.pred.ridge = x.test%% model_ridge$coef + model_ridge$ym
```

Prédiction avec ridge

On peut vérifier la qualité de la prédiction

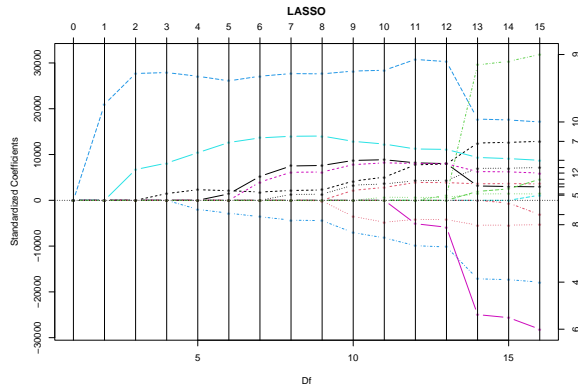
```
plot(cars$price,y.pred.ridge);abline(a=0,b=1)
```



Régression LASSO sous R

Effectuons une régression LASSO sur les données cars

```
library('lars')  
model_lasso=lars(as.matrix(cars[,1:15]),cars$price,type="lasso",  
                 trace=F,normalize=TRUE)  
plot(model_lasso,xvar='df', plottype='coeff')
```



Régression LASSO sous R

On peut afficher les valeurs des coefficients à chaque étape de notre algorithme LARS-LASSO

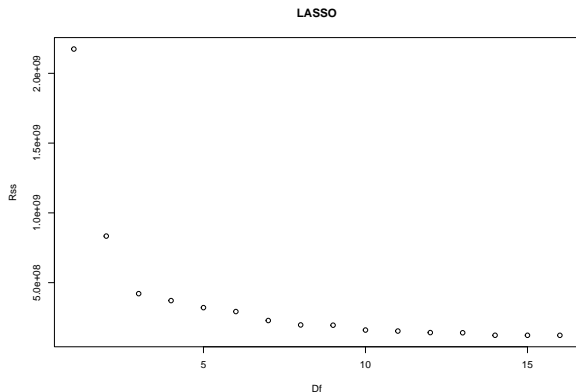
```
print(model_lasso$beta)
```

```
##      diesel      turbo two.doors hatchback wheel.base      length      width
## 0      0.0000      0.000      0.0000      0.0000      0.00000      0.00000      0.0000
## 1      0.0000      0.000      0.0000      0.0000      0.00000      0.00000      0.0000
## 2      0.0000      0.000      0.0000      0.0000      0.00000      0.00000      0.0000
## 3      0.0000      0.000      0.0000      0.0000      0.00000      0.00000     111.8724
## 4      0.0000      0.000      0.0000     -789.5347      0.00000      0.00000     176.0849
## 5     642.1658      0.000      0.0000    -1122.8677      0.00000      0.00000     160.5224
## 6    2387.3986      0.000      0.0000    -1427.4690      0.00000      0.00000     137.3243
## 7    3423.5014      0.000      0.0000    -1728.8728      0.00000      0.00000     163.5281
## 8    3472.4719      0.000      0.0000    -1767.6045      0.00000      0.00000     171.9324
## 9    3975.5130     938.967      0.0000    -2811.9448      0.00000      0.00000     318.9629
## 10   4045.6870    1215.831     261.5374    -3266.8182      0.00000      0.00000     384.6408
## 11   3736.2192    1678.544     240.2845    -3949.4488      0.00000     -67.50439     589.2773
## 12   3657.0753    1669.712     252.9974    -4059.3016      0.00000     -77.54383     604.0831
## 13   1428.5408    1556.148     526.2831    -6832.0639      0.00000    -329.40365     952.2763
## 14   1375.3582    1576.617     536.1358    -6903.9045      0.00000    -338.20113     963.4281
## 15   1345.1121    1543.223     525.4958    -7120.0325     29.73835    -371.48374     979.5767
##      height curb.weight  eng.size horsepower peak.rpm      symbol      city.mpg
## 0      0.0000      0.0000000      0.00000      0.00000      0.000000      0.0000      0.00000
## 1      0.0000      0.0000000    102.74446      0.00000      0.000000      0.0000      0.00000
## 2      0.0000      0.0000000    135.63204     31.44918      0.000000      0.0000      0.00000
## 3      0.0000      0.0000000    136.59424     38.06575      0.000000      0.0000      0.00000
## 4      0.0000      0.0000000    132.65883     48.68140      0.000000      0.0000      0.00000
## 5      0.0000      0.0000000    127.72535     59.12405      0.000000      0.0000      0.00000
## 6      0.0000      0.0000000    132.71437     63.81603      1.515639      0.0000      0.00000
## 7      0.0000      0.0000000    135.49894     65.48059      2.330967     210.4033      0.00000
## 8     -14.0585      0.0000000    135.19833     65.71291      2.352341     219.4773      0.00000
## 9     -327.4630      0.0000000    138.20365     60.28710      2.964816     543.0628      0.00000
## 10    -443.1169      0.0000000    139.05942     57.51567      3.114888     586.6742      0.00000
## 11    -386.5205      0.0000000    150.64756     52.49213      3.066809     700.1839      0.00000
```

Régression LASSO sous R

On peut représenter la valeur du RSS en fonction de nombre de variables actives

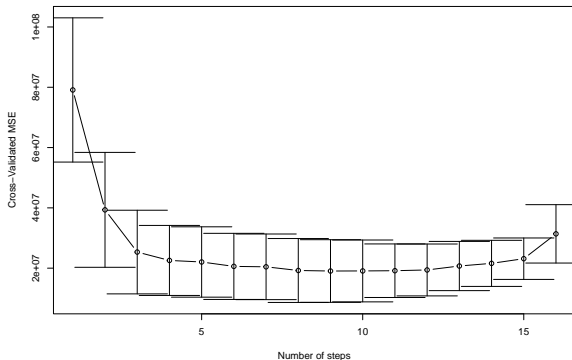
```
plot(summary(model_lasso)$Df,summary(model_lasso)$Rss,  
      xlab='Df',ylab='Rss',main='LASSO')
```



Régression LASSO sous R

On peut aussi calculer l'erreur quadratique moyenne (MSE) par validation croisée (10-fold ici)

```
cv=cv.lars(as.matrix(cars[,1:15]),cars$price,K=10,trace=F,  
           plot.it=T,se=T,type=c("lasso"),mode='step')
```



Régression LASSO sous R

Plutôt que de choisir le λ qui minimise la CV-MSE, on va choisir celui conduisant à la solution la plus parcimonieuse tout en ayant une CV-MSE dans l'intervalle de variable de la CV-MSE minimale.

Le λ à l'étape 3 et les coefficients correspondants sont obtenus par :

```
print(model_lasso$lambda[3])
```

```
## [1] 9261.16
```

```
print(model_lasso$beta[3,])
```

```
##      diesel      turbo  two.doors  hatchback  wheel.base  
##      0.00000      0.00000      0.00000      0.00000      0.00000  
##      width      height  curb.weight  eng.size  horsepower  
##      0.00000      0.00000      0.00000      135.63204      31.44918  
##      symbol  city.mpg  highway.mpg  
##      0.00000      0.00000      0.00000
```

Régression LASSO sous R

La régression LASSO doit être utilisée comme une méthode de sélection de variables. Pour interpréter le modèle, il est préférable de ré-estimer un modèle de régression classique sur les variables sélectionnées par le LASSO

```
model_final<- lm(price~eng.size+horsepower,data=cars)
summary(model_final)
```

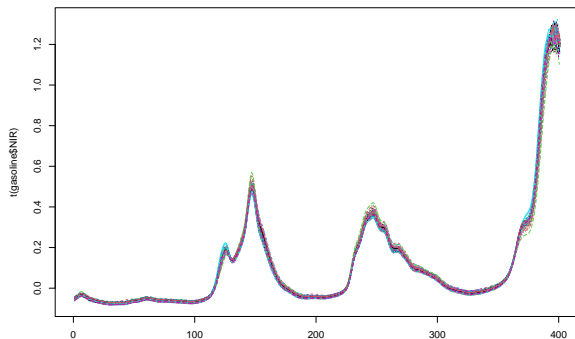
```
##
## Call:
## lm(formula = price ~ eng.size + horsepower, data = cars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7439.5 -2062.9  -216.3  1247.1  7840.3
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -12447.01    2214.63  -5.620 5.79e-06 ***
## eng.size      160.64      29.25   5.492 8.15e-06 ***
## horsepower     55.36      27.97   1.979  0.0581 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3478 on 27 degrees of freedom
## Multiple R-squared:  0.8498, Adjusted R-squared:  0.8386
## F-statistic: 76.36 on 2 and 27 DF,  p-value: 7.698e-12
```

Exercices

Données gasoline

Chercher le meilleur modèle de régression pour prédire l'indice d'octane à partir des spectres NIR du data set gasoline

```
library(pls)
data("gasoline")
matplot(t(gasoline$NIR), type='l')
```



Données criminalité

Chercher le meilleur modèle de régression pour prédire le taux de crimes violents aux Etats-Unis (*ViolentCrimesPerPop*) du dataset :

<https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime+Unnormalized>

Les modèles seront comparés sur la base de la RMSE évaluée par 10-fold CV