

Statistique en grande dimension

L. Rouvière

laurent.rouviere@univ-rennes2.fr

Septembre 2020

- **Objectifs** : identifier le problème de la grande dimension et adapter les techniques traditionnelles à ce cadre.
- **Pré-requis** : théorie des probabilités, modélisation statistique, régression (linéaire et logistique). R, niveau avancé.
- **Enseignant** : Laurent Rouvière laurent.rouviere@univ-rennes2.fr
 - **Recherche** : statistique non paramétrique, apprentissage statistique
 - **Enseignements** : statistique et probabilités (Université, école d'ingénieur et de commerce, formation continue).
 - **Consulting** : énergie, finance, marketing.

- 20h : 10h CM + 8 TP + 2h TD.

Programme

- 20h : 10h CM + 8 TP + 2h TD.
- Matériel :
 - slides à l'url https://lrouviere.github.io/stat_grand_dim/
 - tutoriel (compléments de cours + exercices) à l'url https://lrouviere.github.io/TUTO_GRANDE_DIM/

- 20h : 10h CM + 8 TP + 2h TD.
- Matériel :
 - slides à l'url https://lrouviere.github.io/stat_grand_dim/
 - tutoriel (compléments de cours + exercices) à l'url https://lrouviere.github.io/TUTO_GRANDE_DIM/
- 4 parties :
 1. Introduction : le problème de la grande dimension
 2. Régression sur composantes (PCR et PLS) et sélection de variables
 3. Approches régularisées (ridge/lasso)
 4. Modèle additif ou introduction au graph mining

Première partie I

Introduction : le problème de la grande dimension

Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie

Quelques citations

[Giraud, 2015]

- Over the last twenty years (or so), the dramatic development of data acquisition technologies has enabled devices able to take **thousands (up to million) of measurements simultaneously**.
- Having access to such massive data sounds like a **blessing**.
- Indeed, **separating the useful information from the noise** is generally almost impossible in high dimensional settings.
- This issue is often referred as **the curse of dimensionality**.

[Bühlmann and van de Geer, 2011]

- High-dimensional data are nowadays **rule rather than exception** in areas like information technology, bioinformatics or astronomy...
- The word "high-dimensional" refers to the situation where the **number of unknown parameters** which are to be estimated is one or several orders of magnitude larger than the **number of samples in the data**.
- Classical statistical inference **cannot be used** for high dimensional problems.

- **Constat** : de plus en plus de données à disposition.

En résumé

- **Constat** : de plus en plus de données à disposition.

Positif

Beaucoup d'information pour répondre au problème posé.

En résumé

- **Constat** : de plus en plus de données à disposition.

Positif

Beaucoup d'information pour répondre au problème posé.

Négatif

- Difficile de dissocier l'information pertinente du bruit.
- Modèle de plus en plus complexe \implies de plus en plus de paramètres \implies difficile de bien estimer.

Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie

Détection de spam

- Sur 4 601 mails, on a pu identifier 1813 spams.
- On a également mesuré sur chacun de ces mails la présence ou absence de 57 mots.

```
> spam %>% select(c(1:8,58)) %>% head()
##   make address  all num3d  our over remove internet type
## 1 0.00    0.64 0.64    0 0.32 0.00    0.00    0.00 spam
## 2 0.21    0.28 0.50    0 0.14 0.28    0.21    0.07 spam
## 3 0.06    0.00 0.71    0 1.23 0.19    0.19    0.12 spam
## 4 0.00    0.00 0.00    0 0.63 0.00    0.31    0.63 spam
## 5 0.00    0.00 0.00    0 0.63 0.00    0.31    0.63 spam
## 6 0.00    0.00 0.00    0 1.85 0.00    0.00    1.85 spam
```

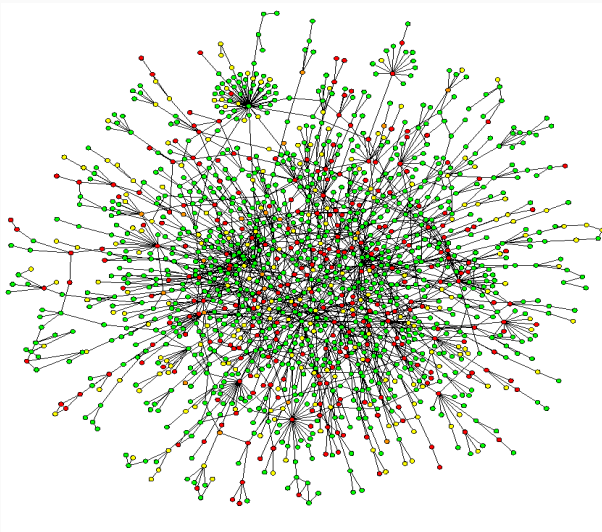
Détection de spam

- Sur 4 601 mails, on a pu identifier 1813 spams.
- On a également mesuré sur chacun de ces mails la présence ou absence de 57 mots.

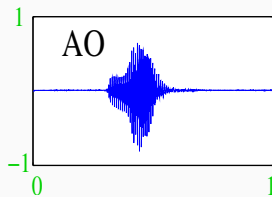
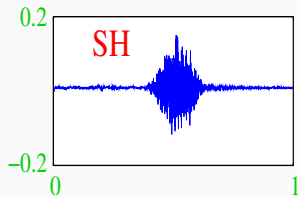
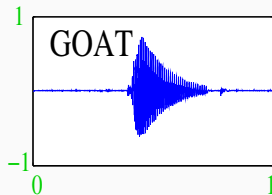
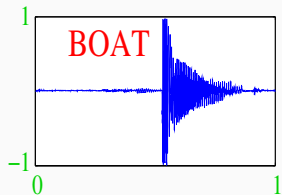
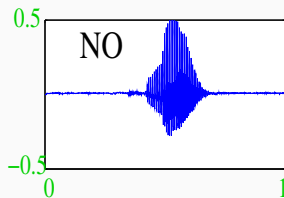
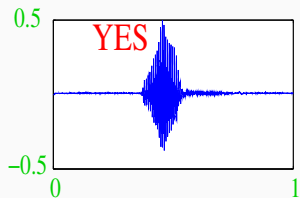
```
> spam %>% select(c(1:8,58)) %>% head()
##   make address  all num3d  our over remove internet type
## 1 0.00    0.64 0.64    0 0.32 0.00    0.00    0.00 spam
## 2 0.21    0.28 0.50    0 0.14 0.28    0.21    0.07 spam
## 3 0.06    0.00 0.71    0 1.23 0.19    0.19    0.12 spam
## 4 0.00    0.00 0.00    0 0.63 0.00    0.31    0.63 spam
## 5 0.00    0.00 0.00    0 0.63 0.00    0.31    0.63 spam
## 6 0.00    0.00 0.00    0 1.85 0.00    0.00    1.85 spam
```

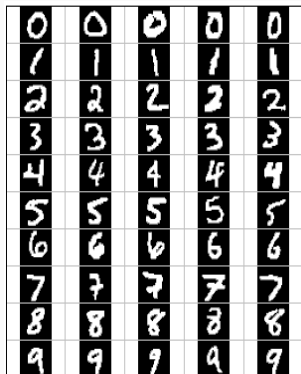
Le problème

Expliquer **type** par $p = 57$ variables.



Données fonctionnelles





Quelques chiffres sur les capacités de stockage

[Besse and Laurent,]

Période	Mémoire	Ordre de grandeur
1940-70	Octet	$n = 30, p \leq 10$
1970	kO	$n = 500, p \leq 10$
1980	MO	Machine Learning
1990	GO	Data-Mining
2000	TO	$p > n$, apprentissage statistique
2010	PO	n explose, cloud, cluster...
2013	??	Big data
2017	??	Intelligence artificielle...

Quelques chiffres sur les capacités de stockage

[Besse and Laurent,]

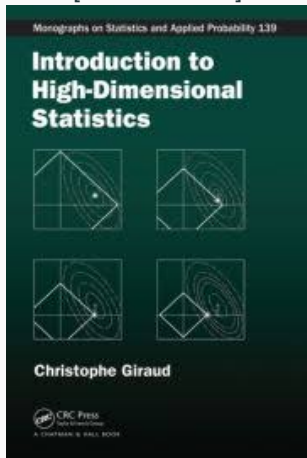
Période	Mémoire	Ordre de grandeur
1940-70	Octet	$n = 30, p \leq 10$
1970	kO	$n = 500, p \leq 10$
1980	MO	Machine Learning
1990	GO	Data-Mining
2000	TO	$p > n$, apprentissage statistique
2010	PO	n explose, cloud, cluster...
2013	??	Big data
2017	??	Intelligence artificielle...

Conclusion

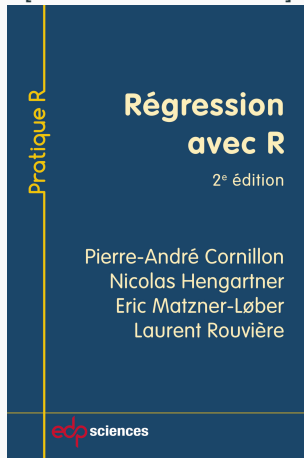
Nécessité d'adapter les techniques traditionnelles à ces données volumineuses.

Références (1)

[Giraud, 2015]

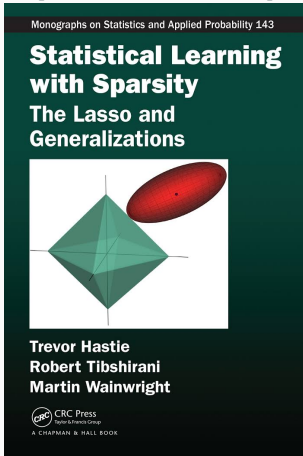


[Cornillon et al., 2019]

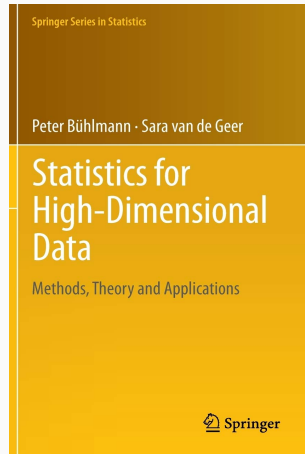


Références (2)

[Hastie et al., 2015]



[Bühlmann and van de Geer, 2011]



Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie

Le problème de régression

- Les données : $(x_1, y_1), \dots, (x_n, y_n)$ avec $x_i \in \mathbb{R}^p$ et $y_i \in \mathbb{R}$.
- Le modèle

$$y_i = m(x_i) + \varepsilon_i \quad \text{avec} \quad \mathbf{E}[\varepsilon_i] = 0 \text{ et } \mathbf{V}[\varepsilon_i] = \sigma^2.$$

Le problème de régression

- Les données : $(x_1, y_1), \dots, (x_n, y_n)$ avec $x_i \in \mathbb{R}^p$ et $y_i \in \mathbb{R}$.
- Le modèle

$$y_i = m(x_i) + \varepsilon_i \quad \text{avec} \quad \mathbf{E}[\varepsilon_i] = 0 \text{ et } \mathbf{V}[\varepsilon_i] = \sigma^2.$$

Le problème

Estimer $m : \mathbb{R}^p \rightarrow \mathbb{R}$.

Le problème de régression

- Les données : $(x_1, y_1), \dots, (x_n, y_n)$ avec $x_i \in \mathbb{R}^p$ et $y_i \in \mathbb{R}$.
- Le modèle

$$y_i = m(x_i) + \varepsilon_i \quad \text{avec} \quad \mathbf{E}[\varepsilon_i] = 0 \text{ et } \mathbf{V}[\varepsilon_i] = \sigma^2.$$

Le problème

Estimer $m : \mathbb{R}^p \rightarrow \mathbb{R}$.

Différentes approches

- Paramétrique : modèle linéaire et estimation par moindres carrés...
- Non paramétrique : noyau, plus proches voisins...

Quelques exemples

Grande dimension en régression

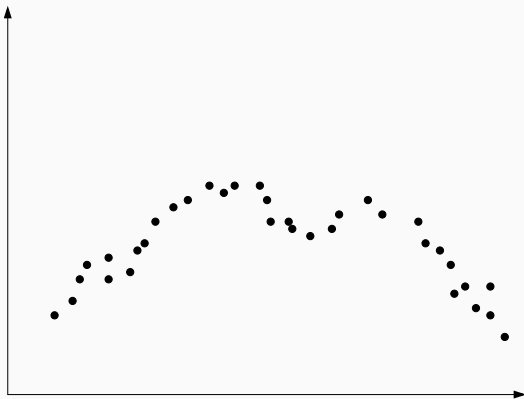
Approche non paramétrique

Approche paramétrique

Bibliographie

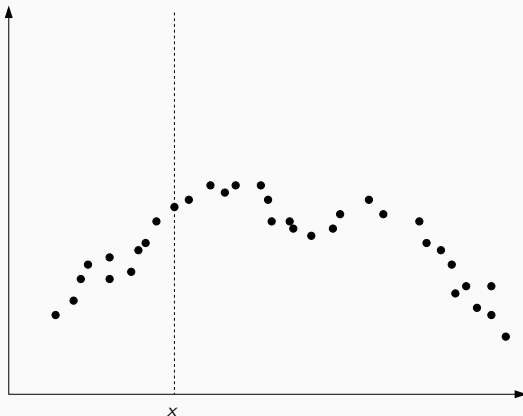
Estimateurs à noyau

- Non paramétriques : moyennes locales $\hat{m}_n(x) = \sum_{i=1}^n W_{ni}(x)y_i$ où les poids $W_{ni}(x)$ vont varier selon les algorithmes.



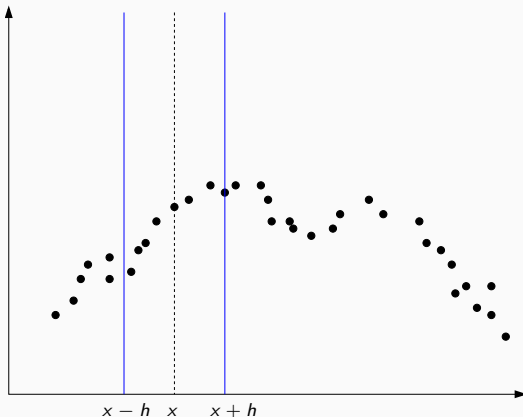
Estimateurs à noyau

- Non paramétriques : moyennes locales $\hat{m}_n(x) = \sum_{i=1}^n W_{ni}(x)y_i$ où les poids $W_{ni}(x)$ vont varier selon les algorithmes.



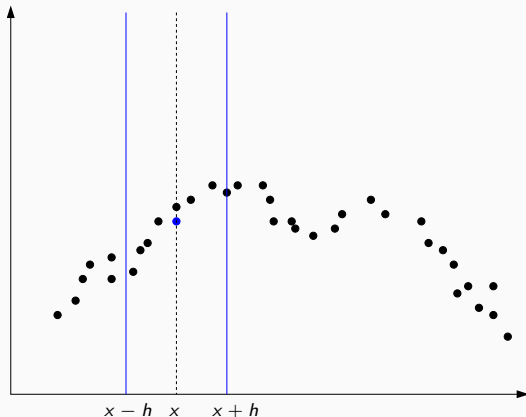
Estimateurs à noyau

- Non paramétriques : moyennes locales $\hat{m}_n(x) = \sum_{i=1}^n W_{ni}(x)y_i$ où les poids $W_{ni}(x)$ vont varier selon les algorithmes.



Estimateurs à noyau

- Non paramétriques : moyennes locales $\hat{m}_n(x) = \sum_{i=1}^n W_{ni}(x)y_i$ où les poids $W_{ni}(x)$ vont varier selon les algorithmes.



- L'estimateur s'écrit

$$\hat{m}_n(x) = \frac{\sum_{i=1}^n \mathbf{1}_{x-h \leq x_i \leq x+h} y_i}{\sum_{i=1}^n \mathbf{1}_{x-h \leq x_i \leq x+h}} = \frac{\sum_{i=1}^n \mathbf{1}_{\left| \frac{x_i - x}{h} \right| \leq 1} y_i}{\sum_{i=1}^n \mathbf{1}_{\left| \frac{x_i - x}{h} \right| \leq 1}}.$$

- L'estimateur s'écrit

$$\hat{m}_n(x) = \frac{\sum_{i=1}^n \mathbf{1}_{x-h \leq x_i \leq x+h} y_i}{\sum_{i=1}^n \mathbf{1}_{x-h \leq x_i \leq x+h}} = \frac{\sum_{i=1}^n \mathbf{1}_{\left| \frac{x_i - x}{h} \right| \leq 1} y_i}{\sum_{i=1}^n \mathbf{1}_{\left| \frac{x_i - x}{h} \right| \leq 1}}.$$

Définition

Soit $h > 0$ et $K : \mathbb{R}^p \rightarrow \mathbb{R}^+$. L'estimateur à noyau de **fenêtre** h et de **noyau** K est défini par

$$\hat{m}_n(x) = \frac{\sum_{i=1}^n K\left(\frac{x_i - x}{h}\right) y_i}{\sum_{i=1}^n K\left(\frac{x_i - x}{h}\right)}.$$

- Noyau usuel dans \mathbb{R}^p :

1. Uniforme : $K(x) = \mathbf{1}_{\|x\| \leq 1}$;
2. Gaussien : $K(x) = \exp(-\|x\|^2)$;
3. Epanechnikov : $K(x) = \frac{3}{4}(1 - \|x\|^2)\mathbf{1}_{\|x\| \leq 1}$.

- Noyau usuel dans \mathbb{R}^p :
 1. Uniforme : $K(x) = \mathbf{1}_{\|x\| \leq 1}$;
 2. Gaussien : $K(x) = \exp(-\|x\|^2)$;
 3. Epanechnikov : $K(x) = \frac{3}{4}(1 - \|x\|^2)\mathbf{1}_{\|x\| \leq 1}$.
- Le choix de h est crucial pour la qualité de l'estimation :
 1. h grand : estimateur « constant », variance faible, biais fort ;
 2. h petit : « interpolation », variance forte, biais faible.

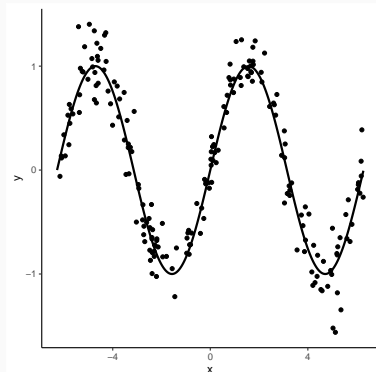
Un exemple

- On génère un échantillon $(X_i, Y_i), i = 1, \dots, n = 200$ selon

$$Y_i = \sin(X_i) + \varepsilon_i, \quad i = 1, \dots, n$$

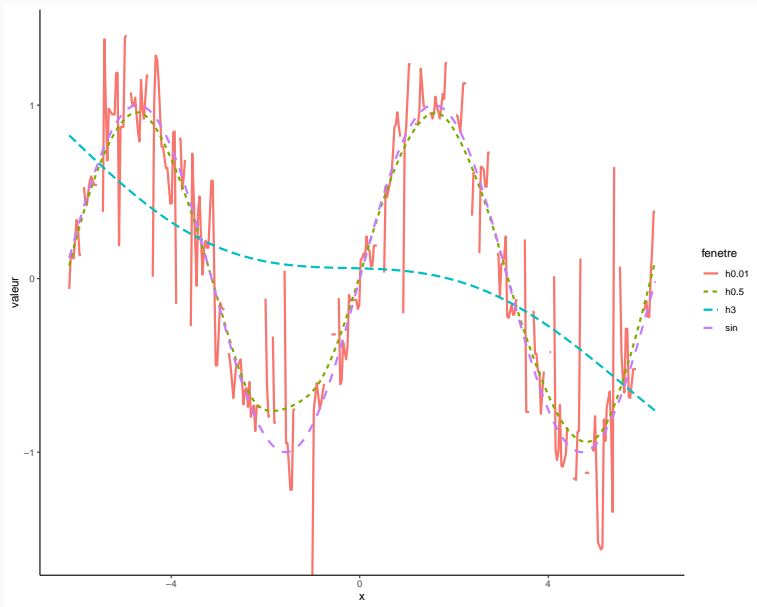
avec X_i uniformes sur $[-2\pi, 2\pi]$, ε_i de loi gaussienne $\mathcal{N}(0, 0.2^2)$.

```
> n <- 200; set.seed(1234)
> X <- runif(n, -2*pi, 2*pi)
> eps <- rnorm(n, 0, 0.2)
> Y <- sin(X) + eps
> df <- data.frame(X=X, Y=Y)
> x <- seq(-2*pi, 2*pi, by=0.01)
> df1 <- data.frame(x=x, y=sin(x))
> ggplot(df1) + aes(x=x, y=y) +
+   geom_line(size=1) +
+   geom_point(data=df, aes(x=X, y=Y)) +
+   theme_classic()
```



Tracé des estimateurs

```
> library("KernSmooth") #package a charger pour la fonction locpoly
> h1 <- 0.5;h2 <- 3;h3 <- 0.01
> fx1 <-locpoly(X,Y,bandwidth=h1)
> fx2 <-locpoly(X,Y,bandwidth=h2)
> fx3 <-locpoly(X,Y,bandwidth=h3)
> df2 <- data.frame(x=fx1$x,"h0.5"=fx1$y,"h3"=fx2$y,"h0.01"=fx3$y) %>%
+   mutate(sin=sin(x)) %>%
+   gather(key="fenetre",value="valeur",-x)
> ggplot(df2)+aes(x=x,y=valeur,color=fenetre,lty=fenetre)+
+   geom_line(size=1)+theme_classic()
```



Algorithme de plus proches voisins

Définition

Soit $k \leq n$ un entier. L'estimateur des k plus proches voisins est défini par

$$\hat{m}_n(x) = \frac{1}{k} \sum_{i \in \text{kppv}(x)} y_i$$

où pour $x \in \mathcal{X}$

$$\text{kppv}(x) = \{i : x_i \text{ fait partie des kppv de } x \text{ parmi } \{x_1, \dots, x_n\}\}.$$

Algorithme de plus proches voisins

Définition

Soit $k \leq n$ un entier. L'estimateur des **k plus proches voisins** est défini par

$$\hat{m}_n(x) = \frac{1}{k} \sum_{i \in \text{kppv}(x)} y_i$$

où pour $x \in \mathcal{X}$

$$\text{kppv}(x) = \{i : x_i \text{ fait partie des kppv de } x \text{ parmi } \{x_1, \dots, x_n\}\}.$$

Remarque

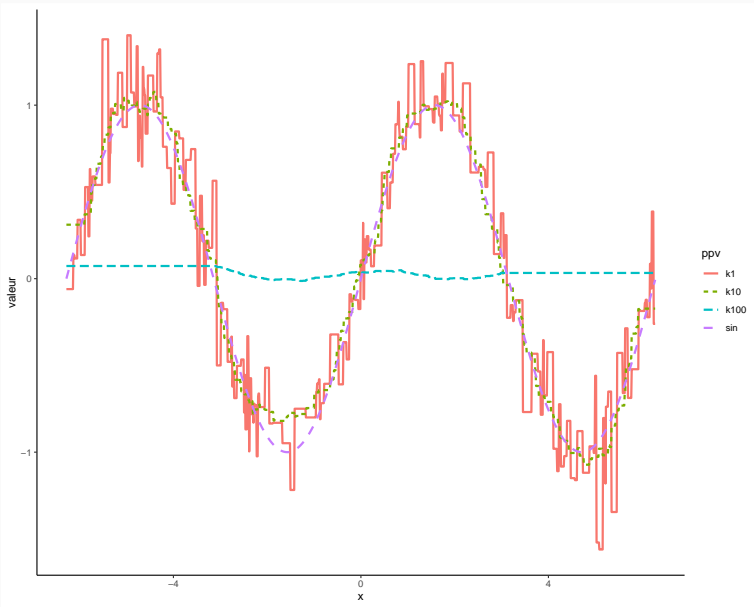
Cette fois, c'est le paramètre k qui est **bleu** crucial pour la qualité de l'estimation :

1. **k grand** : estimateur « constant », variance faible, biais fort ;
2. **k petit** : « sur-ajustement », variance forte, biais faible ;

Exemple

- La fonction `knn.reg` du package `FNN` permet de construire des estimateurs de type k plus proches voisins.

```
> library(FNN)
> k1 <- 10; k2 <- 100; k3 <- 1
> fx1 <- knn.reg(train=X, test=as.matrix(x), y=Y, k=k1)
> fx2 <- knn.reg(train=X, test=as.matrix(x), y=Y, k=k2)
> fx3 <- knn.reg(train=X, test=as.matrix(x), y=Y, k=k3)
> df3 <- data.frame(x=x, "k10"=fx1$pred, "k100"=fx2$pred, "k1"=fx3$pred) %>%
+   mutate(sin=sin(x)) %>%
+   gather(key="ppv", value="valeur", -x)
> ggplot(df3)+aes(x=x, y=valeur, color=ppv, lty=ppv)+
+   geom_line(size=1)+theme_classic()
```



Et que dit la théorie ?

- L'étude des propriétés des estimateurs peut s'effectuer en contrôlant le **risque quadratique**

$$\mathbf{E} \|\hat{m}_n - m\|^2 = \mathbf{E} \left\{ \int (\hat{m}_n(x) - m(x))^2 \mu(dx) \right\}$$

qui se décompose en un terme de **biais** et de **variance**.

- Le contrôle du terme de biais nécessite des **hypothèses sur la régularité** de la fonction à estimer m .
- Nous donnons dans la suite des résultats pour les fonctions **Lipschitziennes** :

$$|m(x) - m(z)| \leq C \|x - z\|, \quad \forall x, z \in \mathbb{R}^p.$$

- Pour l'estimateur à **noyau** de fenêtre $h > 0$, on a

$$\mathbf{E}\|\hat{m}_n - m\|^2 \leq C_1^2 h^2 + \frac{C_2}{nh^p}.$$

- Pour l'estimateur des **k ppv**, on a

$$\mathbf{E}\|\hat{m}_n - m\|^2 \leq \frac{C_3}{k} + C_4 \left(\frac{k}{n}\right)^{2/p}.$$

Théorème [Györfi et al., 2002]

- Pour l'estimateur à **noyau** de fenêtre $h > 0$, on a

$$\mathbf{E}\|\hat{m}_n - m\|^2 \leq C_1^2 h^2 + \frac{C_2}{nh^p}.$$

- Pour l'estimateur des **k ppv**, on a

$$\mathbf{E}\|\hat{m}_n - m\|^2 \leq \frac{C_3}{k} + C_4 \left(\frac{k}{n}\right)^{2/p}.$$

Commentaire

- On retrouve bien **l'importance de h et k** dans les vitesses de convergence.
- On voit également que la **dimension p intervient** dans les vitesses de convergence.

Corollaire

- La fenêtre et le nombre de ppv **optimaux** sont de l'ordre de

$$h^* = C_5 n^{-\frac{1}{p+2}} \quad \text{et} \quad k^* = C_6 n^{\frac{2}{p+2}}.$$

- Pour ces valeurs optimales, le **risque quadratique** vérifie

$$\mathbf{E} \|\hat{m}_n - m\|^2 \leq C_7 n^{-\frac{2}{p+2}}.$$

Corollaire

- La fenêtre et le nombre de ppv **optimaux** sont de l'ordre de

$$h^* = C_5 n^{-\frac{1}{p+2}} \quad \text{et} \quad k^* = C_6 n^{\frac{2}{p+2}}.$$

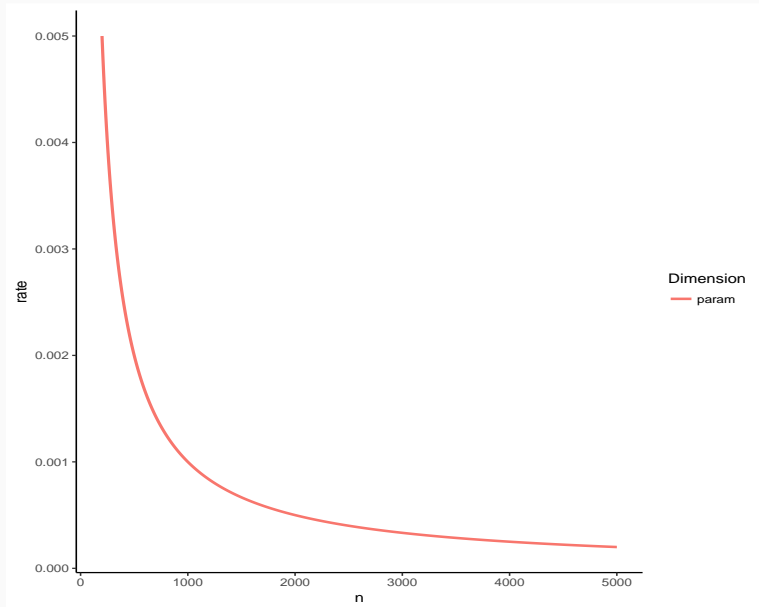
- Pour ces valeurs optimales, le **risque quadratique** vérifie

$$\mathbf{E} \|\hat{m}_n - m\|^2 \leq C_7 n^{-\frac{2}{p+2}}.$$

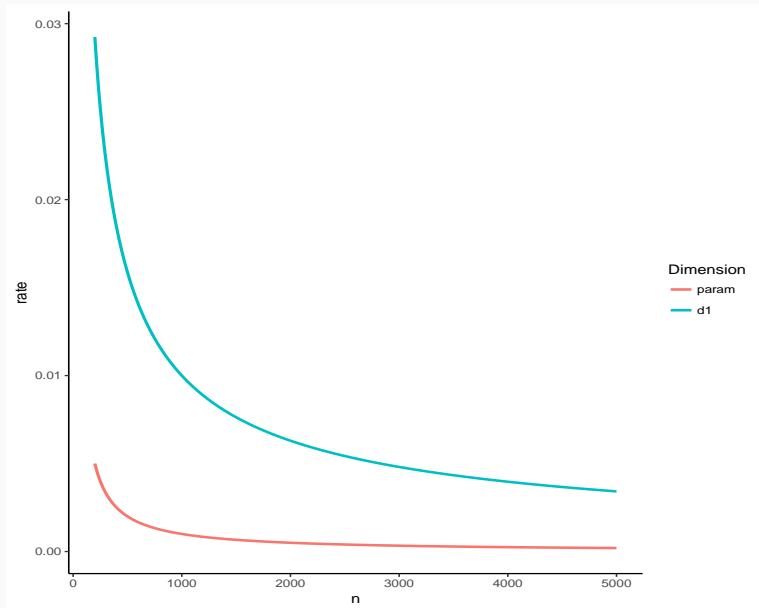
Conséquence

- Lorsque $p \nearrow$, les estimateurs convergent **moins vite** et sont donc **moins précis**.
- C'est le **fléau de la dimension**.

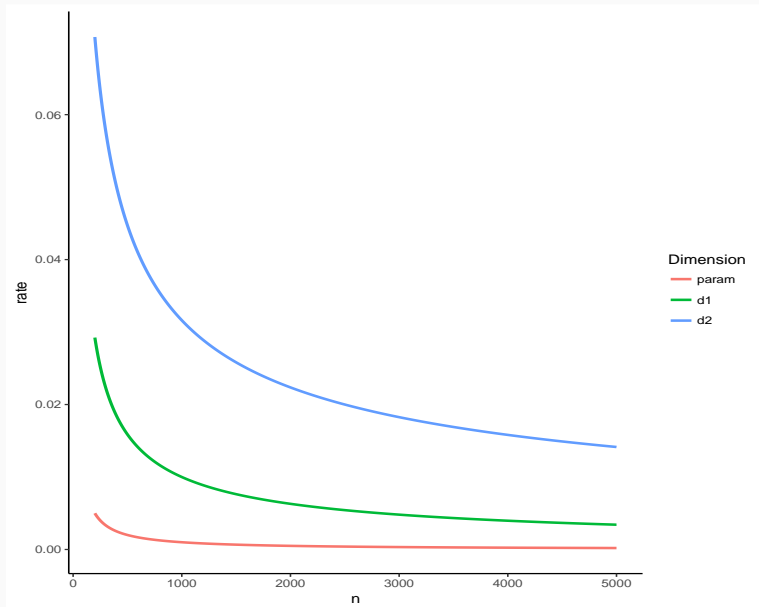
Fléau de la dimension (Illustration)



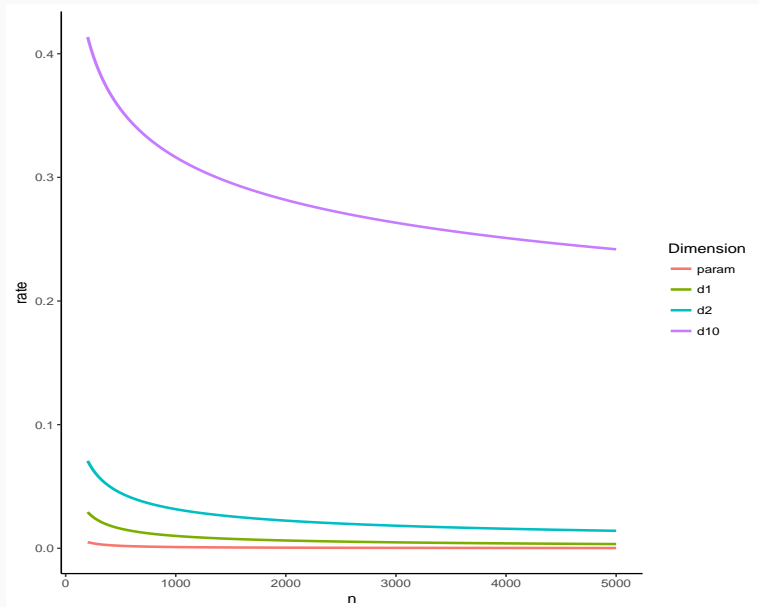
Fléau de la dimension (Illustration)



Fléau de la dimension (Illustration)



Fléau de la dimension (Illustration)



Pourquoi le fléau de la dimension ?

- Les estimateurs non paramétriques présentés sont basés sur des moyennes locales.
- On calcule des moyennes à partir d'observations proches du point où on veut estimer la fonction.

Pourquoi le fléau de la dimension ?

- Les estimateurs non paramétriques présentés sont basés sur des moyennes locales.
- On calcule des moyennes à partir d'observations proches du point où on veut estimer la fonction.
- Lorsque la dimension p augmente, la notion de proximité perd de son sens \implies difficile de trouver des observations proches.
- On dit que les voisinages se vident.

Pourquoi le fléau de la dimension ?

- Les estimateurs non paramétriques présentés sont basés sur des **moyennes locales**.
- On calcule des moyennes à partir d'observations **proches** du point où on veut estimer la fonction.
- Lorsque la dimension p augmente, la notion de proximité perd de son sens \implies difficile de trouver des **observations proches**.
- On dit que les **voisinages se vident**.

Exemple [Giraud, 2015]

Soit $X = (X_1, \dots, X_p)$ et $Y = (Y_1, \dots, Y_p)$ 2 vecteurs aléatoires indépendants de distribution uniforme sur $[0, 1]^p$, alors

$$\mathbb{E}\|X - Y\|^2 = p/6 \quad \text{et} \quad \sigma[\|X - Y\|^2] \approx 0.2\sqrt{p}.$$

Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie

- Le modèle **linéaire** est le modèle **paramétrique** de référence.
- Ce modèle fait **l'hypothèse** que la fonction de régression m est linéaire en ses composantes :

$$y_i = m(x_i) + \varepsilon_i = \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i$$

avec $\mathbf{E}[\varepsilon_i] = 0$ et $\mathbf{V}[\varepsilon_i] = \sigma^2$.

Modèle linéaire

- Le modèle **linéaire** est le modèle **paramétrique** de référence.
- Ce modèle fait **l'hypothèse** que la fonction de régression m est linéaire en ses composantes :

$$y_i = m(x_i) + \varepsilon_i = \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i$$

avec $\mathbf{E}[\varepsilon_i] = 0$ et $\mathbf{V}[\varepsilon_i] = \sigma^2$.

Estimation

Estimer m revient à **estimer** $\beta \in \mathbb{R}^p$ (dimension finie \implies **paramétrique**).

- L'approche **moindres carrés** consiste à minimiser

$$\sum_{i=1}^n \varepsilon_i^2 = \sum_{i=1}^n (y_i - \beta_1 x_{i1} + \dots + \beta_p x_{ip})^2$$

qui fournit l'estimateur des moindres carrés

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

- La fonction des régression m^* est alors estimée par

$$\hat{m}_n(x) = \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p.$$

Propriété

Sous les hypothèses du modèle linéaire, on a

- $\mathbf{E}[\hat{\beta}] = \beta$ et $\mathbf{V}[\hat{\beta}] = (\mathbb{X}^t \mathbb{X})^{-1} \sigma^2$.
- On déduit (sous certaines hypothèses supplémentaires sur le design)

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = O\left(\frac{1}{n}\right) \quad \text{et} \quad \mathbf{E}[(\hat{m}_n(x) - m^*(x))^2] = O\left(\frac{1}{n}\right).$$

Propriété

Sous les **hypothèses du modèle linéaire**, on a

- $\mathbf{E}[\hat{\beta}] = \beta$ et $\mathbf{V}[\hat{\beta}] = (\mathbb{X}^t \mathbb{X})^{-1} \sigma^2$.
- On déduit (sous certains hypothèses supplémentaires sur le design)

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = \mathcal{O}\left(\frac{1}{n}\right) \quad \text{et} \quad \mathbf{E}[(\hat{m}_n(x) - m^*(x))^2] = \mathcal{O}\left(\frac{1}{n}\right).$$

Remarque

- On dit que l'estimateur des moindres carrés converge à la **vitesse paramétrique** $(1/n)$.
- Si on suppose de plus que les **erreurs** $\varepsilon_i, i = 1 \dots, n$ sont **gaussiennes**, on déduit la **loi des estimateurs des moindres carrés** (qui nous permet d'obtenir des intervalles de confiance, des procédures de test...).
- Pour plus de précisions, on pourra se référer à [Grob, 2003, Cornillon et al., 2019].

La dimension en régression linéaire

- La dimension p ne **semble** pas intervenir dans les résultats précédents !

La dimension en régression linéaire

- La dimension p ne **semble** pas intervenir dans les résultats précédents !
- Elle est **bien présente** en réalité (cachée dans les "constantes").

La dimension en régression linéaire

- La dimension p ne **semble** pas intervenir dans les résultats précédents !
- Elle est **bien présente** en réalité (cachée dans les "constantes").

Exemple

- Sous le modèle linéaire, on a

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = \text{Tr}((\mathbb{X}^t \mathbb{X})^{-1}) \sigma^2.$$

- Si on suppose de plus que \mathbb{X} est orthonormale, alors

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = p \sigma^2.$$

La dimension en régression linéaire

- La dimension p ne **semble** pas intervenir dans les résultats précédents !
- Elle est **bien présente** en réalité (cachée dans les "constantes").

Exemple

- Sous le modèle linéaire, on a

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = \text{Tr}((\mathbb{X}^t \mathbb{X})^{-1}) \sigma^2.$$

- Si on suppose de plus que \mathbb{X} est orthonormale, alors

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = p \sigma^2.$$

Conséquence

L'erreur d'estimation **augmente** avec la dimension !

Illustration

- On génère des données $(x_i, y_i), i = 1, \dots, 500$ selon le modèle

$$Y = 1X_1 + 0X_2 + \dots + 0X_{q+1} + \varepsilon$$

où $X_2, X_{q+1}, \dots, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

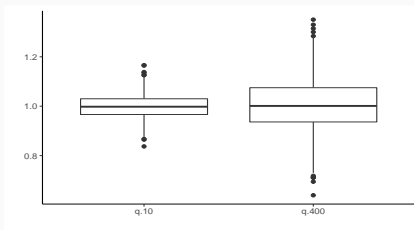
Illustration

- On génère des données $(x_i, y_i), i = 1, \dots, 500$ selon le modèle

$$Y = 1X_1 + 0X_2 + \dots + 0X_{q+1} + \varepsilon$$

où $X_2, X_{q+1}, \dots, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

- On calcule l'estimateur de MCO de β_1 sur 1000 répétitions. On trace les boxplot de ces estimateurs pour $q = 10$ et $q = 400$.



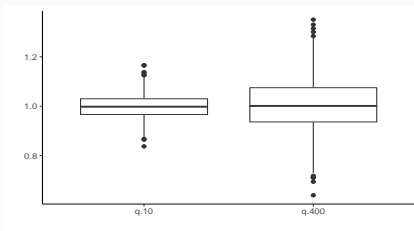
Illustration

- On génère des données $(x_i, y_i), i = 1, \dots, 500$ selon le modèle

$$Y = 1X_1 + 0X_2 + \dots + 0X_{q+1} + \varepsilon$$

où $X_2, X_{q+1}, \dots, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

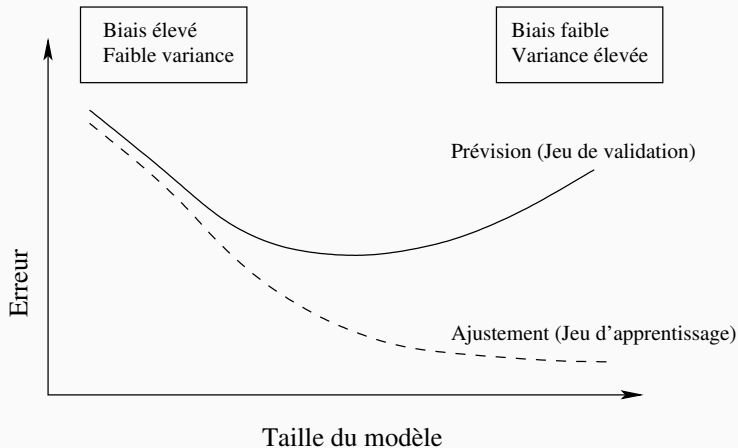
- On calcule l'estimateur de MCO de β_1 sur 1000 répétitions. On trace les boxplot de ces estimateurs pour $q = 10$ et $q = 400$.



Conclusion

Plus de variance (donc moins de précision) lorsque le nombre de variables inutiles augmente.

Taille de modèle



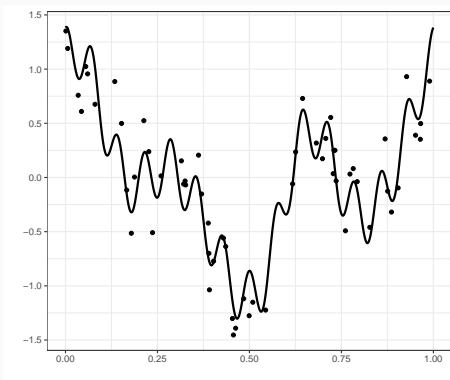
Idem erreur d'estimation (variance) / erreur d'approximation (biais).

Autre exemple : régression fonctionnelle

- On souhaite reconstruire un **signal** à l'aide d'un **échantillon bruité** :

$$y_i = m(x_i) + \varepsilon_i.$$

- L'échantillon et la vraie fonction se trouvent sur la figure ci-dessous.



- La théorie du signal nous dit qu'une fonction (suffisamment) régulière peut être approchée dans le **domaine de Fourier**.
- Pour p assez grand, on a

$$m(x) = \alpha_0 + \sum_{j=1}^p (\beta_j \cos(2\pi jx) + \gamma_j \sin(2\pi jx)) .$$

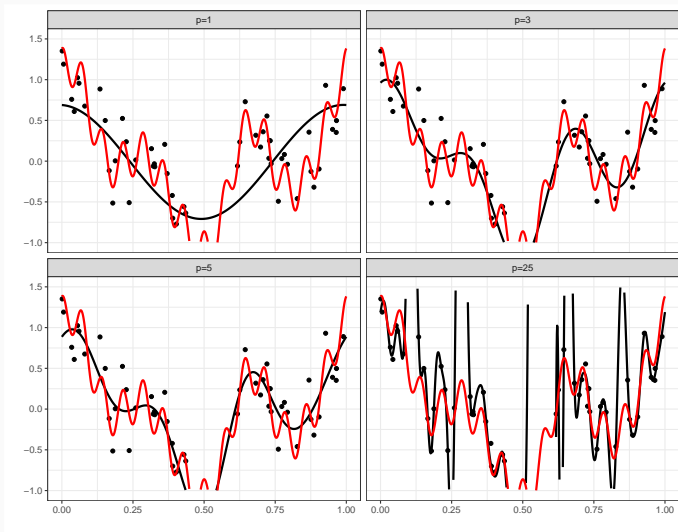
- La théorie du signal nous dit qu'une fonction (suffisamment) régulière peut être approchée dans le **domaine de Fourier**.
- Pour p assez grand, on a

$$m(x) = \alpha_0 + \sum_{j=1}^p (\beta_j \cos(2\pi jx) + \gamma_j \sin(2\pi jx)) .$$

- On propose donc de considérer le **modèle linéaire de dimension $2p + 1$** :

$$y_i = \alpha_0 + \sum_{j=1}^p (\beta_j \cos(2\pi jx_i) + \gamma_j \sin(2\pi jx_i)) + \varepsilon_i .$$

Résultats pour 4 valeurs de p



On **interpole** si p est trop grand.

Conclusion

- En **grande dimension** les approches traditionnelles sont souvent **peu performantes**.
- Nécessité de les corriger.

Conclusion

- En **grande dimension** les approches traditionnelles sont souvent **peu performantes**.
- Nécessité de les corriger.

Comment ?

- Approches **machine learning** : trouver des modèles qui apprennent directement sur les données.
- Méthodes de **réduction de la dimension** : choix de variables ou de combinaisons de variables.
- Approches **régularisées** pour diminuer la variance...

Quelques exemples

Grande dimension en régression

Approche non paramétrique

Approche paramétrique

Bibliographie



Besse, P. and Laurent, B.

Apprentissage Statistique modélisation, prévision, data mining.

INSA - Toulouse.

http://www.math.univ-toulouse.fr/~besse/pub/Appren_stat.pdf.



Bühlmann, P. and van de Geer, S. (2011).

Statistics for High-Dimensional Data : Methods, Theory and Applications.

Springer.



Cornillon, P., Hengartner, N., Matzner-Løber, E., and Rouvière, L. (2019).

Régression avec R.

EDP Sciences.



Giraud, C. (2015).

Introduction to High-Dimensional Statistics.

CRC Press.



Grob, J. (2003).

Linear regression.

Springer.



Györfi, L., Kohler, M., Krzyzak, A., and Harro, W. (2002).
A Distribution-Free Theory of Nonparametric Regression.
Springer.



Hastie, T., Tibshirani, R., and Wainwright, M. (2015).
Statistical Learning with Sparsity : The Lasso and Generalizations.
CRC Press.

[https:](https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf)

[//web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf.](https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf)

Deuxième partie II

Réduction de la dimension

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

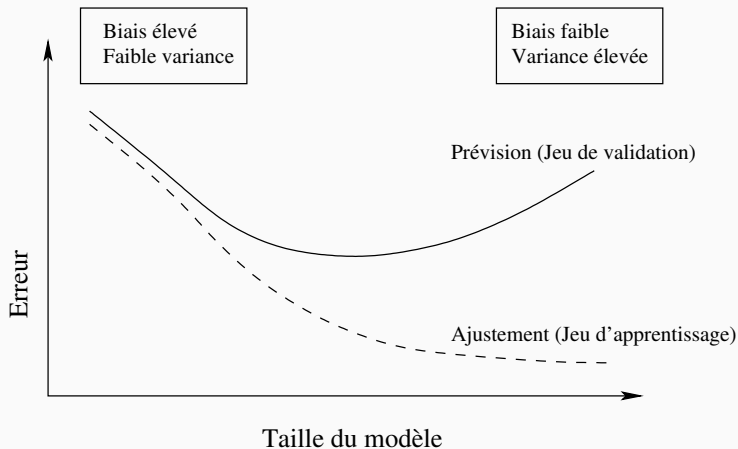
Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

Rappels



Idem erreur d'estimation (variance) / erreur d'approximation (biais).

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. de même loi que (X, Y) à valeurs dans $\mathbb{R}^p \times \mathcal{Y}$;
- Dans ce chapitre, on suppose que $\mathcal{Y} = \mathbb{R}$ ou $\{-1, 1\}$;

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. de même loi que (X, Y) à valeurs dans $\mathbb{R}^p \times \mathcal{Y}$;
- Dans ce chapitre, on suppose que $\mathcal{Y} = \mathbb{R}$ ou $\{-1, 1\}$;

Modèle linéaire et logistique

1. Si $\mathcal{Y} = \mathbb{R}$,

$$m(x) = \mathbf{E}[Y|X = x] = \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d = x^t \beta.$$

2. Si $\mathcal{Y} = \{-1, 1\}$,

$$\text{logit } p(x) = \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d = x^t \beta$$

où $p(x) = \mathbf{P}(Y = 1|X = x)$.

- Ces deux modèles font partie des modèles de référence.

Limites

- Principalement 2 motifs d'insatisfaction :

- Principalement 2 motifs d'insatisfaction :
 1. Précision d'estimation : les estimateurs des MCO pour la régression et du MV pour la logistique ont souvent un biais relativement faible mais une variance élevée (notamment lorsque le nombre de variables p est grand).

- Principalement 2 motifs d'insatisfaction :
 1. **Précision d'estimation** : les estimateurs des MCO pour la régression et du MV pour la logistique ont souvent un biais relativement faible mais une variance élevée (notamment lorsque le nombre de variables p est grand).
 2. **Interprétation** : lorsque le nombre de variables p est grand, on ne connaît pas les variables "importantes".

- Principalement 2 motifs d'insatisfaction :
 1. **Précision d'estimation** : les estimateurs des MCO pour la régression et du MV pour la logistique ont souvent un biais relativement faible mais une variance élevée (notamment lorsque le nombre de variables p est grand).
 2. **Interprétation** : lorsque le nombre de variables p est grand, on ne connaît pas les variables "importantes".

Objectifs

- Avec l'augmentation du volume des données ces dernières années, ces deux inconvénients sont de plus en plus visibles.
- Nécessité de développer des procédures de sélection de sous-groupes de variables ou de combinaison de variables.

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

Best subset selection

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. à valeurs dans $\mathbb{R}^p \times \mathbb{R}$;
- p variables explicatives $\implies 2^p$ modèles concurrents.

Best subset selection

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. à valeurs dans $\mathbb{R}^p \times \mathbb{R}$;
- p variables explicatives $\implies 2^p$ modèles concurrents.

Approche exhaustive

1. Construire les 2^p modèles ;
2. Choisir celui qui optimise un critère donné.

Best subset selection

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. à valeurs dans $\mathbb{R}^p \times \mathbb{R}$;
- p variables explicatives $\implies 2^p$ modèles concurrents.

Approche exhaustive

1. Construire les 2^p modèles ;
2. Choisir celui qui optimise un critère donné.

Algorithme : best subset selection

1. Pour $k = 0, \dots, p$:
 - 1.1 Construire les $\binom{p}{k}$ modèles linéaires à k variables ;
 - 1.2 Choisir parmi ces modèles celui qui a le plus grand R^2 . ; On note \mathcal{M}_k le modèle sélectionné.
2. Choisir, parmi $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_p$, le meilleur modèle au sens d'un critère donné.

Exemples de critères (voir [Cornillon et al., 2019])

- **AIC** : Akaike Information Criterion

$$-2\mathcal{L}_n(\hat{\beta}) + 2p.$$

- **BIC** : Bayesian Information Criterion

$$-2\mathcal{L}_n(\hat{\beta}) + \log(n)p.$$

- **R^2 ajusté** :

$$R_a^2 = 1 - \frac{n-1}{n-p+1}(1 - R^2) \quad \text{où} \quad R^2 = \frac{SSR}{SST} = \frac{\|\hat{Y} - \bar{Y}\mathbf{1}\|^2}{\|Y - \bar{Y}\mathbf{1}\|^2}.$$

- **C_p de Mallow** :

$$C_p = \frac{1}{n} \left(\sum_{i=1}^n (Y_i - \hat{Y}_i)^2 + 2p\hat{\sigma}^2 \right).$$

- Ces critères sont constitués de deux parties :
 1. une qui mesure la **qualité d'ajustement** du modèle ;
 2. une autre qui mesure sa **complexité**.

Ajustement complexité

- Ces critères sont constitués de deux parties :
 1. une qui mesure la **qualité d'ajustement** du modèle ;
 2. une autre qui mesure sa **complexité**.

Exemple AIC

- $-2\mathcal{L}_n(\hat{\beta})$ mesure l'ajustement ;
- $2p$ mesure la complexité.

Ajustement complexité

- Ces critères sont constitués de deux parties :
 1. une qui mesure la **qualité d'ajustement** du modèle ;
 2. une autre qui mesure sa **complexité**.

Exemple AIC

- $-2\mathcal{L}_n(\hat{\beta})$ mesure l'ajustement ;
- $2p$ mesure la complexité.

⇒ l'idée est de choisir un modèle de **complexité minimale** qui **ajuste bien** les données.

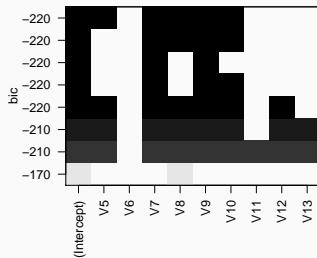
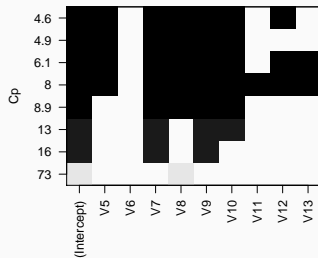
Le coin R

- La fonction `regsubsets` du package `leaps` permet d'utiliser cette approche exhaustive.

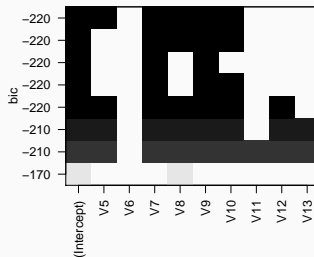
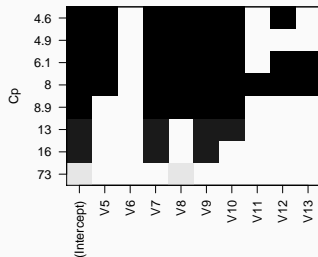
```
> library(mlbench)
> data(Ozone)
> library(leaps)
> reg.fit <- regsubsets(V4~V5+V6+V7+V8+V9+V10+V11+V12+V13,data=Ozone)
> summary(reg.fit)$outmat
```

##		V5	V6	V7	V8	V9	V10	V11	V12	V13
## 1	(1)	"	"	"	"	"	"	"	"	"
## 2	(1)	"	"	"	"	"	"	"	"	"
## 3	(1)	"	"	"	"	"	"	"	"	"
## 4	(1)	"	"	"	"	"	"	"	"	"
## 5	(1)	"	"	"	"	"	"	"	"	"
## 6	(1)	"	"	"	"	"	"	"	"	"
## 7	(1)	"	"	"	"	"	"	"	"	"
## 8	(1)	"	"	"	"	"	"	"	"	"


```
> plot(reg.fit,scale="Cp")
> plot(reg.fit,scale="bic")
```



```
> plot(reg.fit,scale="Cp")
> plot(reg.fit,scale="bic")
```



Conclusion

- C_p sélectionne :

$$Y = \beta_0 + \beta_1 V_5 + \beta_2 V_7 + \beta_3 V_8 + \beta_4 V_9 + \beta_5 V_{10} + \beta_6 V_{12} + \varepsilon.$$

- BIC sélectionne :

$$Y = \beta_0 + \beta_1 V_5 + \beta_2 V_7 + \beta_3 V_8 + \beta_4 V_9 + \beta_5 V_{10} + \varepsilon.$$

Approche pas à pas (stepwise)

- L'avantage de l'approche exhaustive est qu'elle balaie tous les modèles.
- L'inconvénient est que le temps de calcul devient vite très important (résultat long au delà de $p = 30$).

Approche pas à pas (stepwise)

- L'avantage de l'approche exhaustive est qu'elle balaie tous les modèles.
- L'inconvénient est que le temps de calcul devient vite très important (résultat long au delà de $p = 30$).
- Lorsque le nombre de variables est grand, on privilégie souvent les méthodes pas à pas qui consistent à construire les modèles de façon récursive, en ajoutant (ou supprimant) une variable explicative à chaque étape.

Ascendant (forward)

1. Construire \mathcal{M}_0 le **modèle null** (avec uniquement la constante) ;
2. Pour $k = 0, \dots, p - 1$:
 - 2.1 Construire les $p - k$ modèles consistant à ajouter une variable dans \mathcal{M}_k ;
 - 2.2 Choisir, parmi ces $p - k$ modèles, celui qui maximise le $R^2 \rightarrow \mathcal{M}_{k+1}$.
3. Choisir, parmi $\mathcal{M}_0, \dots, \mathcal{M}_p$, le meilleur modèle au sens d'un **critère donné**.

Ascendant (forward)

1. Construire \mathcal{M}_0 le **modèle null** (avec uniquement la constante) ;
2. Pour $k = 0, \dots, p - 1$:
 - 2.1 Construire les $p - k$ modèles consistant à ajouter une variable dans \mathcal{M}_k ;
 - 2.2 Choisir, parmi ces $p - k$ modèles, celui qui maximise le $R^2 \rightarrow \mathcal{M}_{k+1}$.
3. Choisir, parmi $\mathcal{M}_0, \dots, \mathcal{M}_p$, le meilleur modèle au sens d'un **critère donné**.

Descendant (backward)

1. Construire \mathcal{M}_p le **modèle complet** (avec les p variables) ;
2. Pour $k = p, \dots, 1$:
 - 2.1 Construire les k modèles consistant à supprimer une variable dans \mathcal{M}_k ;
 - 2.2 Choisir, parmi ces k modèles, celui qui maximise le $R^2 \rightarrow \mathcal{M}_{k-1}$.
3. Choisir, parmi $\mathcal{M}_0, \dots, \mathcal{M}_p$, le meilleur modèle au sens d'un **critère donné**.

Le coin R

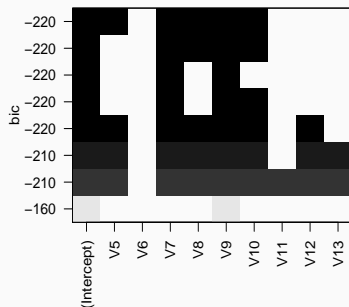
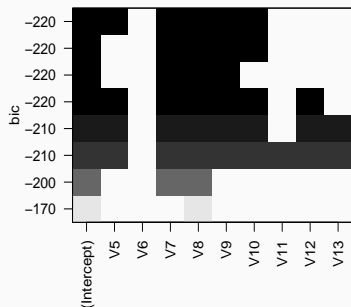
- Il suffit d'ajouter l'argument `method="forward"` ou `method="backward"` dans `regsubsets` pour utiliser ces procédures pas à pas.

```
> reg.fit.for <- regsubsets(V4~V5+V6+V7+V8+V9+V10+V11+V12+V13,  
+                           data=Ozone,method="forward")  
> reg.fit.back <- regsubsets(V4~V5+V6+V7+V8+V9+V10+V11+V12+V13,  
+                             data=Ozone,method="backward")
```

```
> summary(reg.fit.for)$outmat  
##           V5  V6  V7  V8  V9  V10 V11 V12 V13  
## 1 ( 1 ) " " " " " " "*" " " " " " " " "  
## 2 ( 1 ) " " " " "*" "*" " " " " " " " "  
## 3 ( 1 ) " " " " "*" "*" "*" " " " " " "  
## 4 ( 1 ) " " " " "*" "*" "*" "*" " " " " "  
## 5 ( 1 ) "*" " " "*" "*" "*" "*" " " " " "  
## 6 ( 1 ) "*" " " "*" "*" "*" "*" " " "*" " "  
## 7 ( 1 ) "*" " " "*" "*" "*" "*" " " "*" "*" "  
## 8 ( 1 ) "*" " " "*" "*" "*" "*" "*" "*" "
```

```
> summary(reg.fit.back)$outmat  
##           V5  V6  V7  V8  V9  V10 V11 V12 V13  
## 1 ( 1 ) " " " " " " "*" " " " " " " "  
## 2 ( 1 ) " " " " "*" " " "*" " " " " " "  
## 3 ( 1 ) " " " " "*" " " "*" "*" " " " " "  
## 4 ( 1 ) " " " " "*" "*" "*" "*" " " " " "  
## 5 ( 1 ) "*" " " "*" "*" "*" "*" " " " " "  
## 6 ( 1 ) "*" " " "*" "*" "*" "*" " " "*" " "  
## 7 ( 1 ) "*" " " "*" "*" "*" "*" " " "*" "*" "  
## 8 ( 1 ) "*" " " "*" "*" "*" "*" "*" "*" "
```

```
> plot(reg.fit.for,scale="bic")  
> plot(reg.fit.back,scale="bic")
```



Remarque

Sur cet exemple, les deux algorithmes sélectionnent le même modèle (ce n'est pas forcément toujours le cas).

Cas de la discrimination binaire

- Les approches **exhaustive** et **pas à pas** ont été présentées dans un cadre de **régression** ($\mathcal{Y} = \mathbb{R}$) ;

Cas de la discrimination binaire

- Les approches **exhaustive** et **pas à pas** ont été présentées dans un cadre de **régression** ($\mathcal{Y} = \mathbb{R}$) ;
- Elles se **transposent** naturellement à la **discrimination binaire** ($\mathcal{Y} = \{-1, 1\}$).
- Sur R, on pourra utiliser :
 - la fonction **bestglm** du package **bestglm** pour l'approche **exhaustive**.
 - la fonction **step** pour les procédures **pas à pas**.

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

Cadre

1. on cherche toujours à expliquer $Y \in \mathbb{R}$ par $X = (X_1, \dots, X_p) \in \mathbb{R}^p$.
2. p grand et/ou fortes corrélations entre les $X_j, j = 1, \dots, p$.

Cadre

1. on cherche toujours à expliquer $Y \in \mathbb{R}$ par $X = (X_1, \dots, X_p) \in \mathbb{R}^p$.
2. p grand et/ou fortes corrélations entre les $X_j, j = 1, \dots, p$.

L'idée

- Réduire la dimension et atténuer l'influence de la corrélation en ne considérant plus les variables initiales...

Cadre

1. on cherche toujours à expliquer $Y \in \mathbb{R}$ par $X = (X_1, \dots, X_p) \in \mathbb{R}^p$.
2. p grand et/ou fortes corrélations entre les $X_j, j = 1, \dots, p$.

L'idée

- Réduire la dimension et atténuer l'influence de la corrélation en ne considérant plus les variables initiales...
- mais un nombre restreint de combinaisons (linéaires) de variables.

- Trouver un nombre (**petit**) de **combinaisons linéaires des variables initiales** :

$$Z_m = w_{1,m}X_1 + \dots + w_{p,m}X_p = w^t X.$$

- Effectuer **la régression linéaire** de Y sur les $Z_m, m = 1, \dots, M$:

$$Y = \alpha_0 + \alpha_1 Z_1 + \dots + \alpha_M Z_M + \varepsilon.$$

- Trouver un nombre (**petit**) de **combinaisons linéaires des variables initiales** :

$$Z_m = w_{1,m}X_1 + \dots + w_{p,m}X_p = w^t X.$$

- Effectuer **la régression linéaire** de Y sur les $Z_m, m = 1, \dots, M$:

$$Y = \alpha_0 + \alpha_1 Z_1 + \dots + \alpha_M Z_M + \varepsilon.$$

Questions

1. Comment choisir les **combinaisons linéaires** ?
2. Comment choisir **M** ?

Quelques notations

- Y la variable à expliquer, $X = (X_1, \dots, X_p)$ les variables explicatives ;
- $\mathbb{Y} = (y_1, \dots, y_n)$ le vecteur (colonne) des observations de Y ;
- $\mathbb{X}_j = (x_{j1}, \dots, x_{jp})$ le vecteur (colonne) des observations de X_j ;
- \mathbb{X} la matrice $n \times p$ des observations des variables explicatives.

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

- L'analyse en composantes principales (ACP) fait partie des méthodes standards pour construire des combinaisons linéaires de variables quantitatives.
- L'approche consiste à définir des CL des variables qui restituent "au mieux" l'information d'un tableau de données.
- Outil de statistique descriptive (visualiser des données de "grande dimension" dans des espaces de petite dimension) mais aussi de réduction de dimension.

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

Notations

- Tableau des données

$$\mathbb{X} = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{matrix} e_1 \\ \vdots \\ e_n \end{matrix} & \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ \vdots & & \vdots \\ x_{n,1} & \dots & x_{n,p} \end{pmatrix} \end{matrix}$$

Notations

- Tableau des données

$$\mathbb{X} = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{matrix} e_1 \\ \vdots \\ e_n \end{matrix} & \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ \vdots & & \vdots \\ x_{n,1} & \dots & x_{n,p} \end{pmatrix} \end{matrix}$$

- $e_i = (x_{i,1}, \dots, x_{i,p})$ l'individu i et $\mathbb{X}_j = (x_{1,j}, \dots, x_{n,j})$ la variable j .

Notations

- Tableau des données

$$\mathbb{X} = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{matrix} e_1 \\ \vdots \\ e_n \end{matrix} & \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ \vdots & & \vdots \\ x_{n,1} & \dots & x_{n,p} \end{pmatrix} \end{matrix}$$

- $e_i = (x_{i,1}, \dots, x_{i,p})$ l'individu i et $\mathbb{X}_j = (x_{1,j}, \dots, x_{n,j})$ la variable j .
- $e_i \in \mathbb{R}^p$, la représentation de l'ensemble des individus est un nuage de points dans \mathbb{R}^p , appelé **nuage des individus**, \mathcal{N} .

Notations

- Tableau des données

$$\mathbb{X} = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{matrix} e_1 \\ \vdots \\ e_n \end{matrix} & \begin{pmatrix} x_{1,1} & \dots & x_{1,p} \\ \vdots & & \vdots \\ x_{n,1} & \dots & x_{n,p} \end{pmatrix} \end{matrix}$$

- $e_i = (x_{i,1}, \dots, x_{i,p})$ l'individu i et $\mathbb{X}_j = (x_{1,j}, \dots, x_{n,j})$ la variable j .
- $e_i \in \mathbb{R}^p$, la représentation de l'ensemble des individus est un nuage de points dans \mathbb{R}^p , appelé **nuage des individus**, \mathcal{N} .
- $\mathbb{X}_j \in \mathbb{R}^n$, la représentation de l'ensemble des variables est un nuage de points dans \mathbb{R}^n , appelé **nuage des variables**, \mathcal{M} .

Remarque

Si l'œil était capable de **visualiser dans \mathbb{R}^n et \mathbb{R}^p** , il n'y aurait pas de problème...

Objectifs

Déterminer un sous-espace de dimension réduite qui soit "compréhensible" par l'œil sur lequel projeter le nuage.

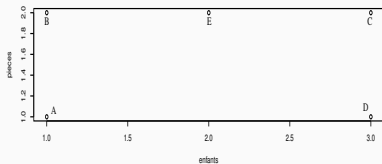
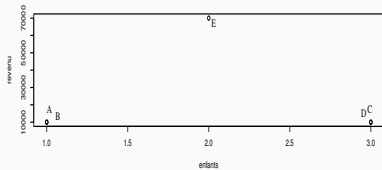
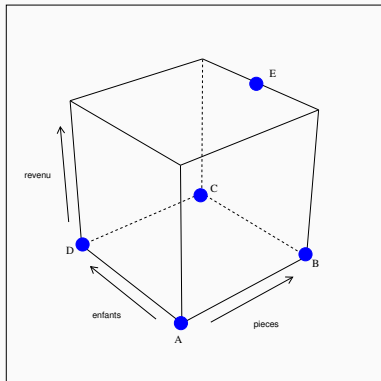
Objectifs

Déterminer un sous-espace de dimension réduite qui soit "compréhensible" par l'œil sur lequel projeter le nuage.

Un exemple "jouet"

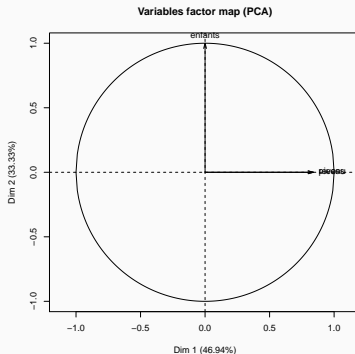
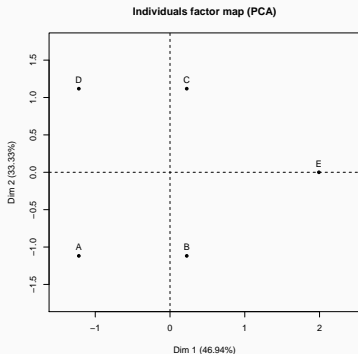
Ménage	Revenu	nb pièces	nb enfants
A	10 000	1	1
B	10 000	2	1
C	10 000	2	3
D	10 000	1	3
E	70 000	2	2

Diverses représentations



Fonction PCA

```
> library(FactoMineR)
> pes.pca <- PCA(df)
```



Projection ACP

Le plan de projection est ici défini par $\mathcal{P} = \text{vect}(X_1 + X_2, X_3)$.

Notations

On se place dans l'espace \mathbb{R}^p muni de la distance euclidienne :

- $\langle e_i, e_j \rangle = \sum_{k=1}^p x_{i,k} x_{j,k}$
- $\|e_i\|^2 = \sum_{k=1}^p e_{i,k}^2$
- $d(e_i, e_j)^2 = \sum_{k=1}^p (x_{i,k} - x_{j,k})^2 = \|e_i - e_j\|^2$

Notations

On se place dans l'espace \mathbb{R}^p muni de la distance euclidienne :

- $\langle e_i, e_j \rangle = \sum_{k=1}^p x_{i,k} x_{j,k}$
- $\|e_i\|^2 = \sum_{k=1}^p e_{i,k}^2$
- $d(e_i, e_j)^2 = \sum_{k=1}^p (x_{i,k} - x_{j,k})^2 = \|e_i - e_j\|^2$

Centrage des données :

- Soit $G = \frac{1}{n} \sum_{i=1}^n e_i = (\bar{X}_1, \dots, \bar{X}_p)$ le centre de gravité du nuage des individus.
- Pour simplifier l'écriture de la méthode, on centre le nuage :

$$e_i^c = \begin{pmatrix} x_{i,1} - \bar{X}_1 \\ \vdots \\ x_{i,p} - \bar{X}_p \end{pmatrix} \quad \text{et} \quad \mathcal{N}^c = \{e_1^c, \dots, e_n^c\}.$$

Idée

Chercher à projeter les observations dans un sous-espace \mathcal{F} visible à l'œil qui "restitue au mieux" l'information contenue dans le tableau.

Idée

Chercher à projeter les observations dans un sous-espace \mathcal{F} visible à l'œil qui "restitue au mieux" l'information contenue dans le tableau.

L'inertie

- On appelle **inertie totale** du nuage de points \mathcal{N}

$$I(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^n d(e_i, G)^2 = \frac{1}{n} \sum_{i=1}^n \|e_i - G\|^2 = \frac{1}{n} \sum_{i=1}^n \|e_i^c\|^2 = I(\mathcal{N}^c).$$

Idée

Chercher à **projeter** les observations dans un sous-espace \mathcal{F} visible à l'œil qui "restitue au mieux" l'**information** contenue dans le tableau.

L'inertie

- On appelle **inertie totale** du nuage de points \mathcal{N}

$$I(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^n d(e_i, G)^2 = \frac{1}{n} \sum_{i=1}^n \|e_i - G\|^2 = \frac{1}{n} \sum_{i=1}^n \|e_i^c\|^2 = I(\mathcal{N}^c).$$

- On appelle **inertie portée par un sous espace \mathcal{F}** du nuage de points \mathcal{N}

$$I_{\mathcal{F}}(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^n \|P_{\mathcal{F}}(e_i^c)\|^2,$$

où $P_{\mathcal{F}}(.)$ est la projection orthogonale sur \mathcal{F} .

Il est facile de voir que $I_{\mathcal{F}}(\mathcal{N}) \leq I(\mathcal{N})$: projeter fait perdre de l'inertie.

Il est facile de voir que $l_{\mathcal{F}}(\mathcal{N}) \leq l(\mathcal{N})$: projeter fait perdre de l'inertie.

Objectif

Trouver le sous espace \mathcal{F} qui minimise cette perte d'inertie, ou encore trouver le sous espace \mathcal{F} tel que

$l_{\mathcal{F}}(\mathcal{N})$ soit maximale.

Il est facile de voir que $I_{\mathcal{F}}(\mathcal{N}) \leq I(\mathcal{N})$: projeter fait perdre de l'inertie.

Objectif

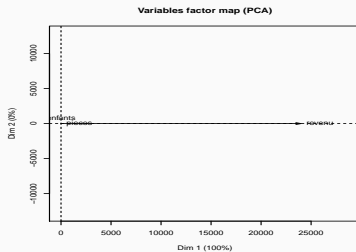
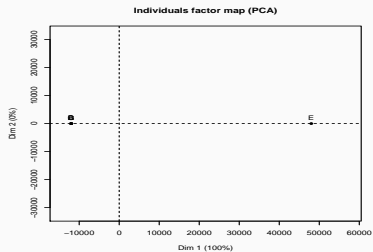
Trouver le sous espace \mathcal{F} qui minimise cette perte d'inertie, ou encore trouver le sous espace \mathcal{F} tel que

$$I_{\mathcal{F}}(\mathcal{N}) \text{ soit maximale.}$$

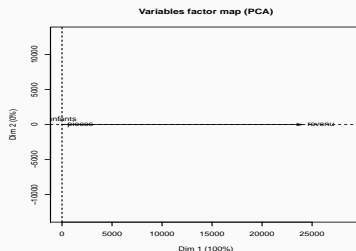
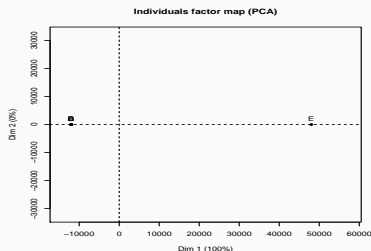
Un "léger" problème

1. Les variables ne sont généralement pas à la même échelle.
2. L'inertie est donc généralement "portée" par un sous groupe de variables.
3. Sur l'exemple, la variable revenu porte à elle seule la quasi totalité de l'inertie...

```
> pes.pca1 <- PCA(df, scale.unit = FALSE)
```



```
> pes.pca1 <- PCA(df, scale.unit = FALSE)
```



Centrage-réduction

Pour pallier à cette difficulté, on **réduit** les données initiales :

$$\mathbb{X} = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{matrix} e_1 \\ \vdots \\ e_n \end{matrix} & \begin{pmatrix} \tilde{x}_{11} & \dots & \tilde{x}_{1p} \\ \vdots & \vdots & \vdots \\ \tilde{x}_{n1} & \dots & \tilde{x}_{np} \end{pmatrix} \end{matrix} \quad \text{avec} \quad \tilde{x}_{ij} = \frac{x_{ij} - \bar{\mathbb{X}}_j}{\sigma_j} \quad \text{et} \quad \sigma_j = \sigma(\mathbb{X}_j).$$

Avec un léger abus, on note $x_{ij} = \tilde{x}_{ij}$.

"Meilleur" sous-espace de dimension 1

Il s'agit de chercher une droite vectorielle Δ_1 dirigée par un vecteur unitaire $u_1 \in \mathbb{R}^p$ telle que $I_{\Delta_1}(\mathcal{N})$ soit maximale.

"Meilleur" sous-espace de dimension 1

Il s'agit de chercher une droite vectorielle Δ_1 dirigée par un **vecteur unitaire** $u_1 \in \mathbb{R}^p$ telle que $I_{\Delta_1}(\mathcal{N})$ soit **maximale**.

Propriété

- $I_{\Delta_1}(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^n \langle e_i, u_1 \rangle^2 = \frac{1}{n} \mathbb{C}'_1 \mathbb{C}_1$ où

$$\mathbb{C}_1 = (\langle e_1, u_1 \rangle, \dots, \langle e_n, u_1 \rangle)' = \mathbb{X}u_1.$$

"Meilleur" sous-espace de dimension 1

Il s'agit de chercher une droite vectorielle Δ_1 dirigée par un **vecteur unitaire** $u_1 \in \mathbb{R}^p$ telle que $I_{\Delta_1}(\mathcal{N})$ soit **maximale**.

Propriété

- $I_{\Delta_1}(\mathcal{N}) = \frac{1}{n} \sum_{i=1}^n \langle e_i, u_1 \rangle^2 = \frac{1}{n} \mathbb{C}_1' \mathbb{C}_1$ où

$$\mathbb{C}_1 = (\langle e_1, u_1 \rangle, \dots, \langle e_n, u_1 \rangle)' = \mathbb{X}u_1.$$

Le problème mathématique

Chercher u_1 unitaire qui maximise $I_{\Delta_1}(\mathcal{N})$ revient à résoudre le **problème d'optimisation** suivant :

$$\text{maximiser } \frac{1}{n} u_1' \mathbb{X}' \mathbb{X} u_1 \text{ sous la contrainte } \|u_1\| = 1.$$

Propriété

Un vecteur propre unitaire u_1 rendant l'inertie $I_{\Delta_1}(\mathcal{N})$ maximale est un vecteur propre normé associé à la **plus grande valeur propre** λ_1 de la matrice $\Sigma = \frac{1}{n}\mathbb{X}'\mathbb{X}$.

Propriété

Un vecteur propre unitaire u_1 rendant l'inertie $I_{\Delta_1}(\mathcal{N})$ maximale est un vecteur propre normé associé à la **plus grande valeur propre** λ_1 de la matrice $\Sigma = \frac{1}{n}\mathbb{X}'\mathbb{X}$.

Remarques

- La matrice d'inertie $\Sigma = \frac{1}{n}\mathbb{X}'\mathbb{X}$ étant symétrique et définie positive, elle est diagonalisable et **toutes ses valeurs propres sont positives ou nulles**.
- u_1 est appelé **premier axe factoriel**.

Exemple

- Sur l'exemple "jouet" on a :

```
> df1 <- as.matrix(df)
> n <- nrow(df1)
> Xbar <- apply(df1,2,mean)
> stdX <- sqrt(apply(df1,2,var)*(n-1)/n)
> dfc <- sweep(df1,2,Xbar,FUN="-")
> dfcr <- sweep(dfc,2,stdX,FUN="/")
> 1/nrow(dfcr)*t(dfcr)%*%dfcr
```

##	revenu	pieces	enfants
## revenu	1.0000000	0.4082483	0
## pieces	0.4082483	1.0000000	0
## enfants	0.0000000	0.0000000	1

- Premier axe factoriel :

```
> XX <- 1/nrow(dfcr)*t(dfcr)%*%dfcr
> u1 <- eigen(XX)$vectors[,1]
> u1
## [1] 0.7071068 0.7071068 0.0000000
```

- Coordonnées sur le premier axe :

```
> dfcr %*%u1
##           [,1]
## A -1.2195788
## B  0.2237969
## C  0.2237969
## D -1.2195788
## E  1.9915638
```

- Que l'on retrouve dans les sorties de PCA :

```
> res.pca$ind$coord[,1]
##           A           B           C           D           E
## -1.2195788  0.2237969  0.2237969 -1.2195788  1.9915638
```

Problème

Trouver une droite vectorielle Δ_2 dirigée par un **vecteur normé** u_2 telle que

$$\begin{cases} I_{\Delta_2}(\mathcal{N}) = u_2' \Sigma u_2 \text{ maximale} \\ \|u_2\|^2 = u_2' u_2 = 1 \\ \langle u_2, u_1 \rangle = u_2' u_1 = 0 \end{cases}$$

Solution

Un vecteur unitaire u_2 solution du problème précédent est un vecteur propre normé associé à la **deuxième plus grande valeur propre** λ_2 de la matrice $\Sigma = \frac{1}{n} \mathbb{X}' \mathbb{X}$.

Question

Le plan $\text{vect}(u_1, u_2)$ est-il le meilleur sous-espace de dimension 2 en terme de maximisation d'inertie projetée ?

Question

Le plan $\text{vect}(u_1, u_2)$ est-il le meilleur sous-espace de dimension 2 en terme de maximisation d'inertie projetée ?

Réponse

La réponse est oui ! On déduit ainsi qu'un sous-espace de dimension $q < p$ qui maximise l'inertie projetée est donné par $\text{vect}(u_1, \dots, u_q)$ où u_j est un vecteur normé associé à la $j^{\text{ème}}$ plus grande valeur propre λ_j de $\Sigma = \frac{1}{n} \mathbb{X}' \mathbb{X}$.

Question

Le plan $\text{vect}(u_1, u_2)$ est-il le meilleur sous-espace de dimension 2 en terme de maximisation d'inertie projetée ?

Réponse

La réponse est oui ! On déduit ainsi qu'un sous-espace de dimension $q < p$ qui maximise l'inertie projetée est donné par $\text{vect}(u_1, \dots, u_q)$ où u_j est un vecteur normé associé à la $j^{\text{ème}}$ plus grande valeur propre λ_j de $\Sigma = \frac{1}{n} \mathbb{X}' \mathbb{X}$.

Conclusion : chercher les axes factoriels revient à diagonaliser $\Sigma = \frac{1}{n} \mathbb{X}' \mathbb{X}$.

ACP \approx changement de base

Base canonique

$$X = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix} \end{matrix}$$

Base $\{u_1, \dots, u_p\}$

$$X = \begin{matrix} & \mathbb{C}_1 & \dots & \mathbb{C}_p \\ \begin{pmatrix} c_{11} & \dots & c_{1p} \\ \vdots & \vdots & \vdots \\ c_{n1} & \dots & c_{np} \end{pmatrix} \end{matrix}$$

ACP \approx changement de base

Base canonique

$$X = \begin{matrix} & \mathbb{X}_1 & \dots & \mathbb{X}_p \\ \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix} \end{matrix}$$

Base $\{u_1, \dots, u_p\}$

$$X = \begin{matrix} & \mathbb{C}_1 & \dots & \mathbb{C}_p \\ \begin{pmatrix} c_{11} & \dots & c_{1p} \\ \vdots & \vdots & \vdots \\ c_{n1} & \dots & c_{np} \end{pmatrix} \end{matrix}$$

Propriété

1. $\mathbb{C}_j = X u_j = \sum_{k=1}^p u_{kj} \mathbb{X}_k$

ACP \approx changement de base

Base canonique

$$X = \begin{pmatrix} \mathbb{X}_1 & \dots & \mathbb{X}_p \\ x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix}$$

Base $\{u_1, \dots, u_p\}$

$$X = \begin{pmatrix} \mathbb{C}_1 & \dots & \mathbb{C}_p \\ c_{11} & \dots & c_{1p} \\ \vdots & \vdots & \vdots \\ c_{n1} & \dots & c_{np} \end{pmatrix}$$

Propriété

1. $\mathbb{C}_j = X u_j = \sum_{k=1}^p u_{kj} \mathbb{X}_k$
2. \mathbb{C}_j centrée, $\mathbf{V}(\mathbb{C}_j) = \frac{1}{n} \|\mathbb{C}_j\|^2 = \lambda_j = I_{\Delta_j}(\mathcal{N})$ et $\rho(\mathbb{C}_j, \mathbb{C}_k) = 0$, $k \neq j$.

ACP \approx changement de base

Base canonique

$$X = \begin{pmatrix} \mathbb{X}_1 & \dots & \mathbb{X}_p \\ x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix}$$

Base $\{u_1, \dots, u_p\}$

$$X = \begin{pmatrix} \mathbb{C}_1 & \dots & \mathbb{C}_p \\ c_{11} & \dots & c_{1p} \\ \vdots & \vdots & \vdots \\ c_{n1} & \dots & c_{np} \end{pmatrix}$$

Propriété

1. $\mathbb{C}_j = \mathbb{X}u_j = \sum_{k=1}^p u_{kj}\mathbb{X}_k$
2. \mathbb{C}_j centrée, $\mathbf{V}(\mathbb{C}_j) = \frac{1}{n}\|\mathbb{C}_j\|^2 = \lambda_j = l_{\Delta_j}(\mathcal{N})$ et $\rho(\mathbb{C}_j, \mathbb{C}_k) = 0$, $k \neq j$.

Conclusion

L'ACP normée remplace les variables d'origines \mathbb{X}_j par de nouvelles variables \mathbb{C}_j appelées **composantes principales**, de variance maximale, non corrélées deux à deux et qui s'expriment comme combinaison linéaire des variables d'origine.

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

1. Choisir un nombre de composantes M .

Etapes

1. Choisir un nombre de composantes M .
2. Calculer les composantes principales $Z_1 = u_1'X, \dots, Z_M = u_M'X$.

Etapas

1. Choisir un nombre de composantes M .
2. Calculer les composantes principales $Z_1 = u_1'X, \dots, Z_M = u_M'X$.
3. Calculer l'estimateur des MCO dans l'espace des composantes principales.

1. Choisir un nombre de composantes M .
2. Calculer les composantes principales $Z_1 = u_1'X, \dots, Z_M = u_M'X$.
3. Calculer l'estimateur des MCO dans l'espace des composantes principales.

Remarque importante

- Comme l'ACP, méthode non invariante par changement d'échelle.

1. Choisir un **nombre de composantes** M .
2. Calculer les **composantes principales** $Z_1 = u_1'X, \dots, Z_M = u_M'X$.
3. Calculer l'estimateur des **MCO** dans l'**espace des composantes principales**.

Remarque importante

- Comme l'ACP, méthode **non invariante par changement d'échelle**.
- Il est souvent préférable de **(centrer)-réduire** les données au préalable.

1. Choisir un **nombre de composantes** M .
2. Calculer les **composantes principales** $Z_1 = u_1'X, \dots, Z_M = u_M'X$.
3. Calculer l'estimateur des **MCO** dans l'**espace des composantes principales**.

Remarque importante

- Comme l'ACP, méthode **non invariante par changement d'échelle**.
- Il est souvent préférable de **(centrer)-réduire** les données au préalable.
- Souvent fait par défaut par les logiciels.

1. Faire l'ACP du tableau \tilde{X} (centré/réduit) $\Rightarrow u_1, \dots, u_M$ axes factoriels et $Z_k = \sum_{j=1}^p u_{k,j} \tilde{X}_j$, $k = 1, \dots, M$ composantes principales.

Algorithme PCR

1. Faire l'ACP du tableau $\tilde{\mathbf{X}}$ (centré/réduit) $\Rightarrow u_1, \dots, u_M$ axes factoriels et $Z_k = \sum_{j=1}^p u_{k,j} \tilde{X}_j, k = 1, \dots, M$ composantes principales.
2. Effectuer la régression de Y sur les Z_j :

$$Y = \theta_0 + \theta_1 Z_1 + \dots + \theta_M Z_M + \varepsilon$$

Estimateurs MCO

$$\hat{\theta}_0 = \bar{Y} \quad \text{et} \quad \hat{\theta}_k = \frac{\langle Z_k, Y \rangle}{\|Z_k\|^2}.$$

- Nouvelle observation $x \implies \tilde{x}$ sa version centrée réduite.

- Nouvelle observation $x \implies \tilde{x}$ sa version centrée réduite.
- Calcul des variables dans l'espace de l'ACP :

$$z_1 = u'_1 \tilde{x}, \dots, z_M = u'_M \tilde{x}.$$

- Nouvelle observation $x \implies \tilde{x}$ sa version centrée réduite.
- Calcul des variables dans l'espace de l'ACP :

$$z_1 = u'_1 \tilde{x}, \dots, z_M = u'_M \tilde{x}.$$

- Valeur prédite :

$$\hat{y} = \hat{\theta}_0 + \hat{\theta}_1 z_1 + \dots + \hat{\theta}_M z_M.$$

- Nouvelle observation $x \implies \tilde{x}$ sa version centrée réduite.
- Calcul des variables dans l'espace de l'ACP :

$$z_1 = u'_1 \tilde{x}, \dots, z_M = u'_M \tilde{x}.$$

- Valeur prédite :

$$\hat{y} = \hat{\theta}_0 + \hat{\theta}_1 z_1 + \dots + \hat{\theta}_M z_M.$$

Problème

- Modèle **difficilement interprétable** puisque les covariables Z sont des CL des variables initiales.
- Idée :

- Nouvelle observation $x \implies \tilde{x}$ sa version centrée réduite.
- Calcul des variables dans l'espace de l'ACP :

$$z_1 = u'_1 \tilde{x}, \dots, z_M = u'_M \tilde{x}.$$

- Valeur prédite :

$$\hat{y} = \hat{\theta}_0 + \hat{\theta}_1 z_1 + \dots + \hat{\theta}_M z_M.$$

Problème

- Modèle **difficilement interprétable** puisque les covariables Z sont des CL des variables initiales.
- Idée : revenir dans l'**espace initial**.

Le modèle dans l'espace initial

$$\begin{aligned}\hat{y} &= \hat{\theta}_0 + \hat{\theta}_1 z_1 + \dots + \hat{\theta}_M z_M \\ &= \hat{\theta}_0 + \hat{\theta}_1 u'_1 \tilde{x} + \dots + \hat{\theta}_M u'_M \tilde{x} \\ &= \hat{\theta}_0 + \hat{\theta}_1 (u_{1,1} \tilde{x}_1 + \dots + u_{1,p} \tilde{x}_p) \\ &\quad \vdots \\ &\quad + \hat{\theta}_M (u_{M,1} \tilde{x}_1 + \dots + u_{M,p} \tilde{x}_p) \\ &= \bar{Y} + \hat{\theta}' v_1 \tilde{x}_1 + \dots + \hat{\theta}' v_p \tilde{x}_p \\ &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p\end{aligned}$$

Le modèle dans l'espace initial

$$\begin{aligned}\hat{y} &= \hat{\theta}_0 + \hat{\theta}_1 z_1 + \dots + \hat{\theta}_M z_M \\ &= \hat{\theta}_0 + \hat{\theta}_1 u'_1 \tilde{x} + \dots + \hat{\theta}_M u'_M \tilde{x} \\ &= \hat{\theta}_0 + \hat{\theta}_1 (u_{1,1} \tilde{x}_1 + \dots + u_{1,p} \tilde{x}_p) \\ &\quad \vdots \\ &\quad + \hat{\theta}_M (u_{M,1} \tilde{x}_1 + \dots + u_{M,p} \tilde{x}_p) \\ &= \bar{Y} + \hat{\theta}' v_1 \tilde{x}_1 + \dots + \hat{\theta}' v_p \tilde{x}_p \\ &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \dots + \hat{\beta}_p x_p\end{aligned}$$

avec

$$\hat{\beta}_0 = \bar{Y} - \sum_{j=1}^p \hat{\theta}' v_j \frac{\bar{X}_j}{\sigma_{\mathbb{X}_j}} \quad \text{et} \quad \hat{\beta}_j = \frac{\hat{\theta}' v_j}{\sigma_{\mathbb{X}_j}}, j = 1, \dots, p.$$

Remarque importante

Ce changement d'espace est **important à connaître** car c'est généralement dans l'**espace initial** que les logiciels renvoient les **coefficients**.

Remarque importante

Ce changement d'espace est **important à connaître** car c'est généralement dans l'**espace initial** que les logiciels renvoient les **coefficients**.

- **Exemple** : jeu de données **Hitters**

```
> library(ISLR)
> Hitters <- na.omit(Hitters)
> dim(Hitters)
## [1] 263 20
> names(Hitters)
## [1] "AtBat"      "Hits"      "HmRun"     "Runs"     "RBI"
## [6] "Walks"     "Years"     "CAtBat"    "CHits"    "CHmRun"
## [11] "CRuns"     "CRBI"      "CWalks"    "League"   "Division"
## [16] "PutOuts"   "Assists"   "Errors"    "Salary"   "NewLeague"
```

Remarque importante

Ce changement d'espace est **important à connaître** car c'est généralement dans l'**espace initial** que les logiciels renvoient les **coefficients**.

- **Exemple** : jeu de données **Hitters**

```
> library(ISLR)
> Hitters <- na.omit(Hitters)
> dim(Hitters)
## [1] 263 20
> names(Hitters)
## [1] "AtBat"      "Hits"      "HmRun"     "Runs"     "RBI"
## [6] "Walks"     "Years"     "CAAtBat"   "CHits"    "CHmRun"
## [11] "CRuns"     "CRBI"      "CWalks"    "League"   "Division"
## [16] "PutOuts"   "Assists"   "Errors"    "Salary"   "NewLeague"
```

Le problème

Expliquer **Salary** par les autres variables.

La fonction `pcr`

- La fonction `pcr` du package `pls` permet d'effectuer la régression :

```
> library(pls)
> pcr.fit <- pcr(Salary~.,data=Hitters,scale=TRUE,ncomp=19)
```

La fonction `pcr`

- La fonction `pcr` du package `pls` permet d'effectuer la régression :

```
> library(pls)
> pcr.fit <- pcr(Salary~.,data=Hitters,scale=TRUE,ncomp=19)
```

- On peut obtenir les `coefficients` pour $M = 5$ (dans l'espace des variables initiales) avec :

```
> coefficients(pcr.fit,ncomp=5)
## , , 5 comps
##           Salary
## AtBat      28.766042
## Hits       30.447021
## HmRun       25.844498
## Runs       33.000876
```

Attention

- Ces coefficients sont calculés pour les données `réduites` ;
- Il faut diviser par les `écart-types` si on veut les valeurs sur les `variables initiales` ($\hat{\beta}_j$).

Prédiction

- Comme d'habitude, la fonction `predict` permet de calculer des prévisions.
- Pour obtenir les **valeurs prédites (ou ajustées)** des 5 premiers individus, on utilise

```
> predict(pcr.fit,newdata=Hitters[1:5,],ncomp=5)
## , , 5 comps
##
##           Salary
## -Alan Ashby    495.0068
## -Alvin Davis   547.8896
## -Andre Dawson  1010.2236
## -Andres Galarraga 409.8232
## -Alfredo Griffin 524.9053
```

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

- Idem à PCR : objectif Réduction de dimension.
- On cherche toujours des CL Z_1, \dots, Z_M des variables explicatives.

- Idem à PCR : objectif Réduction de dimension.
- On cherche toujours des CL Z_1, \dots, Z_M des variables explicatives.
- Différence avec PCR : les composantes PLS vont être construites en maximisant le lien avec la variable à expliquer Y .

Quelques notations

- Y variable à expliquer et $X = (X_1, \dots, X_p)$ variables explicatives.
- $\mathbb{Y} = (y_1, \dots, y_n)$ et \mathbb{X} la matrice $n \times p$ des variables explicatives.
- Comme pour PCR, les covariables **sont centrées réduites** en début d'analyse $\implies \tilde{\mathbb{X}}$ la matrice \mathbb{X} centrée réduite.

Première composante PLS

- On note $\mathbb{Y}^{(1)} = \mathbb{Y}$ et $\tilde{\mathbb{X}}^{(1)} = \tilde{\mathbb{X}}$
- Elle s'obtient en calculant les **produits scalaires** entre $\mathbb{Y}^{(1)}$ et les $\tilde{\mathbb{X}}_j^{(1)}$:

$$w_{1,j} = \langle \mathbb{Y}^{(1)}, \tilde{\mathbb{X}}_j^{(1)} \rangle \quad \text{ou encore} \quad w_1 = (w_{1,1}, \dots, w_{1,p}) = \tilde{\mathbb{X}}^{(1)'} \mathbb{Y}^{(1)}.$$

Première composante PLS

- On note $\mathbb{Y}^{(1)} = \mathbb{Y}$ et $\tilde{\mathbb{X}}^{(1)} = \tilde{\mathbb{X}}$
- Elle s'obtient en calculant les **produits scalaires** entre $\mathbb{Y}^{(1)}$ et les $\tilde{\mathbb{X}}_j^{(1)}$:

$$w_{1,j} = \langle \mathbb{Y}^{(1)}, \tilde{\mathbb{X}}_j^{(1)} \rangle \quad \text{ou encore} \quad w_1 = (w_{1,1}, \dots, w_{1,p}) = \tilde{\mathbb{X}}^{(1)'} \mathbb{Y}^{(1)}.$$

- **Première composante PLS** : $Z_1 = w_1' \tilde{\mathbb{X}}^{(1)}$ et $\mathbb{Z}_1 = \tilde{\mathbb{X}}^{(1)} w_1$.

Première composante PLS

- On note $\mathbb{Y}^{(1)} = \mathbb{Y}$ et $\tilde{\mathbb{X}}^{(1)} = \tilde{\mathbb{X}}$
- Elle s'obtient en calculant les **produits scalaires** entre $\mathbb{Y}^{(1)}$ et les $\tilde{\mathbb{X}}_j^{(1)}$:

$$w_{1,j} = \langle \mathbb{Y}^{(1)}, \tilde{\mathbb{X}}_j^{(1)} \rangle \quad \text{ou encore} \quad w_1 = (w_{1,1}, \dots, w_{1,p}) = \tilde{\mathbb{X}}^{(1)'} \mathbb{Y}^{(1)}.$$

- **Première composante PLS** : $Z_1 = w_1' \tilde{\mathbb{X}}^{(1)}$ et $\mathbb{Z}_1 = \tilde{\mathbb{X}}^{(1)} w_1$.
- **Régression MCO** de $Y^{(1)}$ sur Z_1 : $Y^{(1)} = \alpha_0 + \alpha_1 Z_1 + \varepsilon$. Estimateurs :

$$\hat{\alpha}_0 = \bar{Y}^{(1)} \quad \text{et} \quad \hat{\alpha}_1 = \frac{\langle \mathbb{Z}_1, \mathbb{Y}^{(1)} \rangle}{\langle \mathbb{Z}_1, \mathbb{Z}_1 \rangle}.$$

- Il s'obtient par

$$\hat{Y}^{(1)} = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1.$$

- Il s'obtient par

$$\hat{Y}^{(1)} = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1.$$

- On peut l'exprimer en fonction des **variables centrées réduites** :

$$\hat{Y}^{(1)} = \hat{\alpha}_0 + \hat{\alpha}_1 w_{1,1} \tilde{X}_1 + \dots + \hat{\alpha}_1 w_{1,p} \tilde{X}_p.$$

- La régression est une **projection** : $\hat{Y}^{(1)} = P_{Z_1}(Y^{(1)})$.
- **Interprétation** :

- La régression est une **projection** : $\hat{Y}^{(1)} = P_{Z_1}(Y^{(1)})$.
- **Interprétation** : $\hat{Y}^{(1)}$ s'interprète comme la part de $Y^{(1)}$ expliquée par Z_1 .
- La part non expliquée est le **résidu**

$$Y^{(2)} = P_{Z_1^\perp}(Y^{(1)}) = \hat{\varepsilon}_1 = Y^{(1)} - \hat{Y}^{(1)}$$

- La régression est une **projection** : $\hat{Y}^{(1)} = P_{Z_1}(Y^{(1)})$.
- **Interprétation** : $\hat{Y}^{(1)}$ s'interprète comme la part de $Y^{(1)}$ expliquée par Z_1 .
- La part non expliquée est le **résidu**

$$Y^{(2)} = P_{Z_1^\perp}(Y^{(1)}) = \hat{\varepsilon}_1 = Y^{(1)} - \hat{Y}^{(1)}$$

L'idée

Expliquer la **partie résiduelle** par la "meilleure" combinaison linéaire orthogonale à Z_1 .

Calcul de la deuxième composante

1. On **orthogonalise** chaque $\tilde{X}_j^{(1)}$ par rapport à Z_1 :

$$\tilde{X}_j^{(2)} = P_{Z_1^\perp}(\tilde{X}_j^{(1)}) = (\text{Id} - P_{Z_1})(\tilde{X}_j^{(1)}) = \tilde{X}_j^{(1)} - \frac{\langle Z_1, \tilde{X}_j^{(1)} \rangle}{\langle Z_1, Z_1 \rangle} Z_1.$$

Calcul de la deuxième composante

1. On **orthogonalise** chaque $\tilde{X}_j^{(1)}$ par rapport à Z_1 :

$$\tilde{X}_j^{(2)} = P_{Z_1^\perp}(\tilde{X}_j^{(1)}) = (\text{Id} - P_{Z_1})(\tilde{X}_j^{(1)}) = \tilde{X}_j^{(1)} - \frac{\langle Z_1, \tilde{X}_j^{(1)} \rangle}{\langle Z_1, Z_1 \rangle} Z_1.$$

2. **Produits scalaires** entre $Y^{(2)}$ et les $\tilde{X}_j^{(1)}$

$$w_{2,j} = \langle Y^{(2)}, \tilde{X}_j^{(2)} \rangle \quad \text{ou encore} \quad w_2 = (w_{2,1}, \dots, w_{2,p}) = \tilde{X}^{(2)'} Y^{(2)}.$$

3. **Deuxième composante PLS** : $Z_2 = w_2' \tilde{X}^{(2)}$ et $Z_2 = \tilde{X}^{(2)} w_2$.

Calcul de la deuxième composante

1. On **orthogonalise** chaque $\tilde{X}_j^{(1)}$ par rapport à Z_1 :

$$\tilde{X}_j^{(2)} = P_{Z_1^\perp}(\tilde{X}_j^{(1)}) = (\text{Id} - P_{Z_1})(\tilde{X}_j^{(1)}) = \tilde{X}_j^{(1)} - \frac{\langle Z_1, \tilde{X}_j^{(1)} \rangle}{\langle Z_1, Z_1 \rangle} Z_1.$$

2. **Produits scalaires** entre $Y^{(2)}$ et les $\tilde{X}_j^{(1)}$

$$w_{2,j} = \langle Y^{(2)}, \tilde{X}_j^{(2)} \rangle \quad \text{ou encore} \quad w_2 = (w_{2,1}, \dots, w_{2,p}) = \tilde{X}^{(2)'} Y^{(2)}.$$

3. **Deuxième composante PLS** : $Z_2 = w_2' \tilde{X}^{(2)}$ et $Z_2 = \tilde{X}^{(2)} w_2$.

4. **Régression MCO** de $Y^{(2)}$ sur Z_2 : $Y^{(2)} = \alpha_2 Z_2 + \varepsilon$. Estimateur :

$$\hat{\alpha}_2 = \frac{\langle Z_2, Y^{(2)} \rangle}{\langle Z_2, Z_2 \rangle} = \frac{\langle Z_2, Y \rangle}{\langle Z_2, Z_2 \rangle}$$

puisque $\hat{Y}^{(1)}$ est orthogonal à Z_2 .

- Il s'obtient par

$$\hat{Y}^{(2)} = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1 + \hat{\alpha}_2 Z_2.$$

- Il s'obtient par

$$\hat{Y}^{(2)} = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1 + \hat{\alpha}_2 Z_2.$$

- Ici encore, ce modèle peut s'exprimer comme un modèle **linéaire** en fonction des **variables initiales**

$$\hat{Y}^{(2)} = \hat{\alpha}_0 + \hat{\beta}_1^{(2)} \tilde{X}_1 + \dots + \hat{\beta}_p^{(2)} \tilde{X}_p.$$

Modèle PLS à 2 composantes

- Il s'obtient par

$$\hat{Y}^{(2)} = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1 + \hat{\alpha}_2 Z_2.$$

- Ici encore, ce modèle peut s'exprimer comme un modèle **linéaire** en fonction des **variables initiales**

$$\hat{Y}^{(2)} = \hat{\alpha}_0 + \hat{\beta}_1^{(2)} \tilde{X}_1 + \dots + \hat{\beta}_p^{(2)} \tilde{X}_p.$$

Généralisation

On peut **itérer ce procédé p fois** pour obtenir l'algorithme **PLS**.

L'algorithme PLS, voir [Hastie et al., 2009]

1. On pose $\hat{\mathbf{Y}}^{(0)} = \bar{\mathbf{Y}}$ et $\tilde{\mathbf{X}}_j^{(0)} = \tilde{\mathbf{X}}_j, j = 1, \dots, p$.
2. Pour $m = 1, \dots, p$:
 - 2.1 $\mathbf{Z}_m = \sum_{j=1}^p w_{m,j} \tilde{\mathbf{X}}_j^{(m-1)}$ avec $w_{m,j} = \langle \tilde{\mathbf{X}}_j^{(m-1)}, \mathbf{Y} \rangle$.
 - 2.2 $\hat{\alpha}_m = \frac{\langle \mathbf{Z}_m, \mathbf{Y} \rangle}{\langle \mathbf{Z}_m, \mathbf{Z}_m \rangle}$.
 - 2.3 $\hat{\mathbf{Y}}^{(m)} = \hat{\mathbf{Y}}^{(m-1)} + \hat{\alpha}_m \mathbf{Z}_m$.
 - 2.4 Orthogonaliser $\tilde{\mathbf{X}}_j^{(m-1)}$ par rapport à \mathbf{Z}_m :

$$\tilde{\mathbf{X}}_j^{(m)} = \tilde{\mathbf{X}}_j^{(m-1)} - \frac{\langle \mathbf{Z}_m, \tilde{\mathbf{X}}_j^{(m-1)} \rangle}{\langle \mathbf{Z}_m, \mathbf{Z}_m \rangle} \mathbf{Z}_m.$$

3. **Sortie** : la suite de valeurs ajustées $\{\hat{\mathbf{Y}}^{(m)}\}_1^p$.

- Le modèle à m composantes s'écrit en fonction des **composantes PLS** :

$$\hat{Y}^{(m)} = \bar{Y} + \hat{\alpha}_1 Z_1 + \dots + \hat{\alpha}_m Z_m$$

- Le modèle à m composantes s'écrit en fonction des **composantes PLS** :

$$\hat{Y}^{(m)} = \bar{Y} + \hat{\alpha}_1 Z_1 + \dots + \hat{\alpha}_m Z_m$$

- On peut le réécrire comme une **combinaison linéaire des variables initiales** :

$$\hat{Y}^{(m)} = \hat{\beta}_0^{(m)} + \hat{\beta}_1^{(m)} \tilde{X}_1 + \dots + \hat{\beta}_p^{(m)} \tilde{X}_p.$$

Retour dans l'espace initial

- Le modèle à m composantes s'écrit en fonction des **composantes PLS** :

$$\hat{Y}^{(m)} = \bar{Y} + \hat{\alpha}_1 Z_1 + \dots + \hat{\alpha}_m Z_m$$

- On peut le réécrire comme une **combinaison linéaire des variables initiales** :

$$\hat{Y}^{(m)} = \hat{\beta}_0^{(m)} + \hat{\beta}_1^{(m)} \tilde{X}_1 + \dots + \hat{\beta}_p^{(m)} \tilde{X}_p.$$

- Ce sont généralement ces coefficients $\hat{\beta}_j^{(m)}$ qui sont **renvoyés par les logiciels**.

- La régression PLS s'effectue à l'aide de la fonction `plsr` du package `pls`.

```
> pls.fit <- plsr(Salary~.,data=Hitters,scale=TRUE)
```

- La régression PLS s'effectue à l'aide de la fonction `plsr` du package `pls`.

```
> pls.fit <- plsr(Salary~.,data=Hitters,scale=TRUE)
```

- On obtient les **coefficients PLS** pour la première composante avec

```
> coefficients(pls.fit,ncomp = 1)
## , , 1 comps
##
##           Salary
## AtBat      25.0420570
## Hits      27.8270677
## HmRun      21.7597795
## Runs      26.6334747
## RBI       28.5110396
```

PCR vs PLS

- Les deux approches permettent de réduire la dimension en considérant un nombre restreint de composantes Z_1, \dots, Z_m .
- Dans les deux cas, ces composantes
 1. sont des combinaisons linéaires des variables X_1, \dots, X_p .
 2. sont orthogonales.
- La seule différence entre les deux approches se situe dans le processus de construction des composantes.

PCR vs PLS

- Les deux approches permettent de **réduire la dimension** en considérant une **nombre restreint de composantes** Z_1, \dots, Z_m .
- Dans les deux cas, ces composantes
 1. sont des **combinaisons linéaires** des variables X_1, \dots, X_p .
 2. sont **orthogonales**.
- La seule **différence** entre les deux approches se situe dans le processus de **construction des composantes**.

Remarque

- **PCR** utilise **uniquement les X** pour construire les composantes.
- **PLS** utilise les **X et Y** .

PCR vs PLS

- Les deux approches permettent de **réduire la dimension** en considérant une **nombre restreint de composantes** Z_1, \dots, Z_m .
- Dans les deux cas, ces composantes
 1. sont des **combinaisons linéaires** des variables X_1, \dots, X_p .
 2. sont **orthogonales**.
- La seule **différence** entre les deux approches se situe dans le processus de **construction des composantes**.

Remarque

- **PCR** utilise **uniquement les X** pour construire les composantes.
- **PLS** utilise les **X et Y** .
- Il est possible d'écrire le **problème d'optimisation** résolu par ces composantes.

Propriété, voir [Hastie et al., 2009]

On note S la matrice de variance covariance de $\tilde{\mathbf{X}}$: $S = \tilde{\mathbf{X}}'\tilde{\mathbf{X}}$.

- Pour PCR, la m^{e} composante principale w_m est solution du problème

$$\max_w \mathbf{V}(\tilde{\mathbf{X}}w)$$

sous la contrainte $\|w\| = 1, w'Sw_\ell = 0, \ell = 1, \dots, m-1$.

Propriété, voir [Hastie et al., 2009]

On note S la matrice de variance covariance de $\tilde{\mathbf{X}}$: $S = \tilde{\mathbf{X}}'\tilde{\mathbf{X}}$.

- Pour PCR, la m^{e} composante principale w_m est solution du problème

$$\max_w \mathbf{V}(\tilde{\mathbf{X}}w)$$

sous la contrainte $\|w\| = 1, w'Sw_\ell = 0, \ell = 1, \dots, m-1$.

- Pour PLS, la m^{e} composante w_m est solution du problème

$$\max_w \text{Corr}^2(\mathbb{Y}, \tilde{\mathbf{X}}w) \mathbf{V}(\tilde{\mathbf{X}}w)$$

sous la contrainte $\|w\| = 1, w'Sw_\ell = 0, \ell = 1, \dots, m-1$.

- Les **composantes principales** se calculent en cherchant la direction de \mathbb{R}^p dans laquelle la **variabilité** (des X) est maximale.

- Les **composantes principales** se calculent en cherchant la direction de \mathbb{R}^p dans laquelle la **variabilité** (des X) est maximale.
- Pour **PLS**, on cherche à maximiser également la variabilité des X ,

- Les **composantes principales** se calculent en cherchant la direction de \mathbb{R}^p dans laquelle la **variabilité** (des X) est maximale.
- Pour **PLS**, on cherche à maximiser également la variabilité des X , mais aussi la **direction la plus corrélée à Y** .

- Les **composantes principales** se calculent en cherchant la direction de \mathbb{R}^p dans laquelle la **variabilité** (des X) est maximale.
- Pour **PLS**, on cherche à maximiser également la variabilité des X , mais aussi la **direction la plus corrélée à Y** .
- Selon [Hastie et al., 2009], la **variance** a souvent une place **plus importante que la corrélation** dans le critère

- Les **composantes principales** se calculent en cherchant la direction de \mathbb{R}^p dans laquelle la **variabilité** (des X) est maximale.
- Pour **PLS**, on cherche à maximiser également la variabilité des X , mais aussi la **direction la plus corrélée à Y** .
- Selon [Hastie et al., 2009], la **variance** a souvent une place **plus importante que la corrélation** dans le critère \implies les deux approches sont souvent **proches**.

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)

Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

- PCR et PLS construisent des **composantes** Z_1, \dots, Z_p .
- Chaque composante s'écrit omme une **combinaison linéaire** des $X_j, j = 1, \dots, p$.
- Pour une valeur de $m \leq p$, on peut prédire selon

$$\hat{Y}^{(m)} = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1 + \dots + \hat{\alpha}_m Z_m.$$

- PCR et PLS construisent des **composantes** Z_1, \dots, Z_p .
- Chaque composante s'écrit omme une **combinaison linéaire** des $X_j, j = 1, \dots, p$.
- Pour une valeur de $m \leq p$, on peut prédire selon

$$\hat{Y}^{(m)} = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1 + \dots + \hat{\alpha}_m Z_m.$$

Propriété

Si $m = p$ alors **PCR et PLS sont équivalents au modèle linaire classique** par MCO avec les p variables X_1, \dots, X_p .

- PCR et PLS construisent des **composantes** Z_1, \dots, Z_p .
- Chaque composante s'écrit omme une **combinaison linéaire** des $X_j, j = 1, \dots, p$.
- Pour une valeur de $m \leq p$, on peut prédire selon

$$\hat{Y}^{(m)} = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1 + \dots + \hat{\alpha}_m Z_m.$$

Propriété

Si $m = p$ alors **PCR et PLS sont équivalents au modèle linaire classique** par MCO avec les p variables X_1, \dots, X_p .

Conséquence

- Si $m = p$ alors PCR et PLS ne sont d'**aucune utilité**.
- Il est donc **crucial** de choisir le **"bon" nombre de composantes** $m \in \{1, \dots, p\}$.

- Pour $m \in \{1, \dots, p\}$ PCR et PLS fournissent un **algorithme de prévision**

$$\hat{f}_m(x) = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1(x) + \dots + \hat{\alpha}_m Z_m(x).$$

- Les méthodes permettant de choisir m sont identiques aux **procédures de calibration en machine learning** :

- Pour $m \in \{1, \dots, p\}$ PCR et PLS fournissent un **algorithme de prévision**

$$\hat{f}_m(x) = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1(x) + \dots + \hat{\alpha}_m Z_m(x).$$

- Les méthodes permettant de choisir m sont identiques aux **procédures de calibration en machine learning** :
 1. Choix d'un **risque** (erreur quadratique de prévision...)

- Pour $m \in \{1, \dots, p\}$ PCR et PLS fournissent un **algorithme de prévision**

$$\hat{f}_m(x) = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1(x) + \dots + \hat{\alpha}_m Z_m(x).$$

- Les méthodes permettant de choisir m sont identiques aux **procédures de calibration en machine learning** :
 1. Choix d'un **risque** (erreur quadratique de prévision...)
 2. Choix d'un **algorithme pour calculer ce risque** (validation hold out, validation croisée, LOO...)

- Pour $m \in \{1, \dots, p\}$ PCR et PLS fournissent un **algorithme de prévision**

$$\hat{f}_m(x) = \hat{\alpha}_0 + \hat{\alpha}_1 Z_1(x) + \dots + \hat{\alpha}_m Z_m(x).$$

- Les méthodes permettant de choisir m sont identiques aux **procédures de calibration en machine learning** :
 1. Choix d'un **risque** (erreur quadratique de prévision...)
 2. Choix d'un **algorithme pour calculer ce risque** (validation hold out, validation croisée, LOO...)
 3. Sélection du paramètre qui **minimise le risque** calculé.

Exemple : validation croisée

- **Risque RMSEP** : $\mathcal{R}(f) = \sqrt{\mathbf{E}[(Y - f(X))^2]}$.
- **Algorithme** : validation croisée K blocs.

Choix du nombre de composantes

Entrées :

- les observations $(x_1, y_1), \dots, (x_n, y_n)$;
- $\{\mathcal{I}_1, \dots, \mathcal{I}_K\}$ une partition de $\{1, \dots, n\}$ en K blocs ;

Pour $m = 1, \dots, p$

- Pour $k = 1, \dots, K$:
 1. Construire la régression sur m composantes en utilisant l'ensemble des données privé du k^{e} bloc, c'est-à-dire $\{(x_i, y_i) : i \in \{1, \dots, n\} \setminus \mathcal{I}_k\} \implies \hat{f}_k^{(m)}$.
 2. Calculer la valeur prédite par l'algorithme pour chaque observation du bloc k : $\hat{f}_k^{(m)}(x_i), i \in \mathcal{I}_k$.

Retourner : pour $m = 1, \dots, p$

$$\hat{\mathcal{R}}(\hat{f}^{(m)}) = \frac{1}{n} \sum_{k=1}^K \sum_{i \in \mathcal{I}_k} \sqrt{(y_i - \hat{f}_k^{(m)}(x_i))^2}.$$

Exemple : PCR

- Il suffit d'utiliser l'argument `validation="CV"`.

```
> set.seed(1234)
> pcr.val <- pcr(Salary~.,data=Hitters,scale=TRUE,validation="CV")
> RMSEP(pcr.val)
```

##	(Intercept)	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps
## CV	452	353.4	351.8	351.7	349.4	345.4	343.6
## adjCV	452	353.0	351.4	351.3	348.9	344.8	342.8
##	7 comps	8 comps	9 comps	10 comps	11 comps	12 comps	13 comps
## CV	343.6	345.3	347.0	349.3	349.4	351.5	355.2
## adjCV	342.9	344.4	346.1	348.1	348.0	350.1	353.8
##	14 comps	15 comps	16 comps	17 comps	18 comps	19 comps	
## CV	349.4	348.5	339.6	338.7	337.2	339.5	
## adjCV	347.6	346.8	337.9	336.9	335.4	337.5	

Exemple : PCR

- Il suffit d'utiliser l'argument `validation="CV"`.

```
> set.seed(1234)
> pcr.val <- pcr(Salary~.,data=Hitters,scale=TRUE,validation="CV")
> RMSEP(pcr.val)
```

##	(Intercept)	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps
## CV	452	353.4	351.8	351.7	349.4	345.4	343.6
## adjCV	452	353.0	351.4	351.3	348.9	344.8	342.8
##	7 comps	8 comps	9 comps	10 comps	11 comps	12 comps	13 comps
## CV	343.6	345.3	347.0	349.3	349.4	351.5	355.2
## adjCV	342.9	344.4	346.1	348.1	348.0	350.1	353.8
##	14 comps	15 comps	16 comps	17 comps	18 comps	19 comps	
## CV	349.4	348.5	339.6	338.7	337.2	339.5	
## adjCV	347.6	346.8	337.9	336.9	335.4	337.5	

Conclusion

On choisira 18 composantes pour PCR.

Exemple : PLS

```
> set.seed(1234)
> pls.val <- plsr(Salary~.,data=Hitters,scale=TRUE,validation="CV")
> RMSEP(pls.val)
```

##	(Intercept)	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps
## CV	452	348.5	345.6	345.7	345.1	348.4	349.0
## adjCV	452	348.1	344.8	344.7	344.1	346.7	346.9
##	7 comps	8 comps	9 comps	10 comps	11 comps	12 comps	13 comps
## CV	345.7	341.4	341.8	339.8	338.0	336.7	339.0
## adjCV	343.7	339.5	339.9	338.0	336.3	335.0	337.2
##	14 comps	15 comps	16 comps	17 comps	18 comps	19 comps	
## CV	338.9	338.2	338.2	338.2	338.1	339.5	
## adjCV	337.0	336.3	336.4	336.3	336.3	337.5	

Exemple : PLS

```
> set.seed(1234)
> pls.val <- plsr(Salary~.,data=Hitters,scale=TRUE,validation="CV")
> RMSEP(pls.val)
```

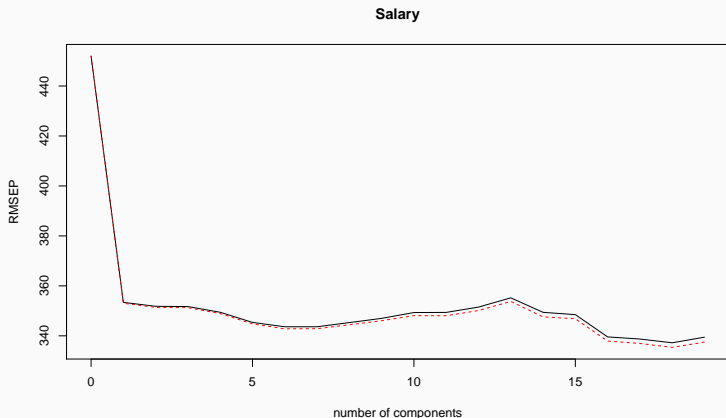
##	(Intercept)	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps
## CV	452	348.5	345.6	345.7	345.1	348.4	349.0
## adjCV	452	348.1	344.8	344.7	344.1	346.7	346.9
##	7 comps	8 comps	9 comps	10 comps	11 comps	12 comps	13 comps
## CV	345.7	341.4	341.8	339.8	338.0	336.7	339.0
## adjCV	343.7	339.5	339.9	338.0	336.3	335.0	337.2
##	14 comps	15 comps	16 comps	17 comps	18 comps	19 comps	
## CV	338.9	338.2	338.2	338.2	338.1	339.5	
## adjCV	337.0	336.3	336.4	336.3	336.3	337.5	

Conclusion

On choisira 12 composantes pour PLS.

- On peut également visualiser les **erreurs** en fonction du **nombre de composantes** avec **validationplot**.

```
> validationplot(pcr.val)
```



- Deux techniques pour réduire la dimension :
 1. sélection de variables.
 2. régression sur composantes.

- Deux techniques pour réduire la dimension :
 1. sélection de variables.
 2. régression sur composantes.
- A utiliser lorsque :
 1. p est grand.
 2. les $X_j, j = 1, \dots, p$ sont "corrélés".

Sélections exhaustive et pas à pas

Régression sur composantes

Régression sur composantes principales (PCR)


Rappels ACP

Retour à PCR

Régression PLS

Choix du nombre de composantes

Bibliographie

 Cornillon, P., Hengartner, N., Matzner-Løber, E., and Rouvière, L. (2019).

Régression avec R.

EDP Sciences.

 Hastie, T., Tibshirani, R., and Friedman, J. (2009).

The Elements of Statistical Learning : Data Mining, Inference, and Prediction.

Springer, second edition.

Troisième partie III

Approches régularisées

Régression ridge

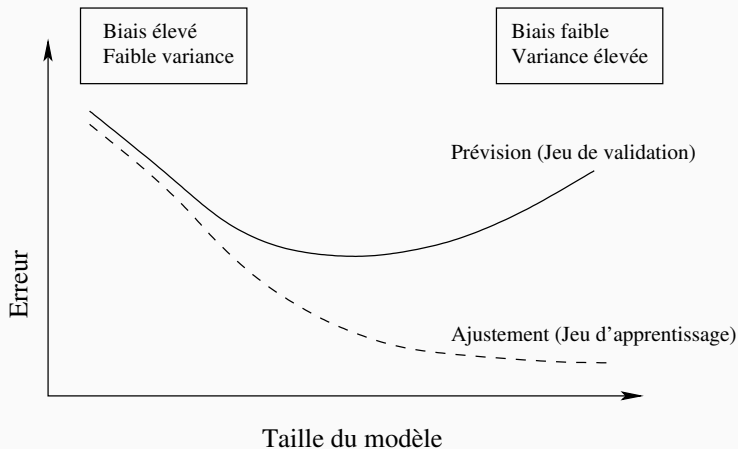
Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie

Rappels



Idem erreur d'estimation (variance) / erreur d'approximation (biais).

Illustration

- On génère des données $(x_i, y_i), i = 1, \dots, 500$ selon le modèle

$$Y = 1X_1 + 0X_2 + \dots + 0X_{q+1} + \varepsilon$$

où $X_2, X_{q+1}, \dots, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

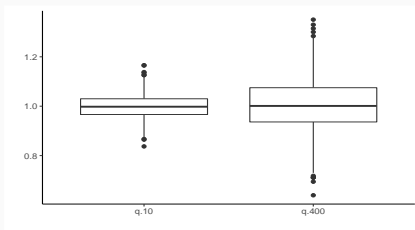
Illustration

- On génère des données $(x_i, y_i), i = 1, \dots, 500$ selon le modèle

$$Y = 1X_1 + 0X_2 + \dots + 0X_{q+1} + \varepsilon$$

où $X_2, X_{q+1}, \dots, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

- On calcule l'estimateur de MCO de β_1 sur 1000 répétitions. On trace les boxplot de ces estimateurs pour $q = 10$ et $q = 400$.



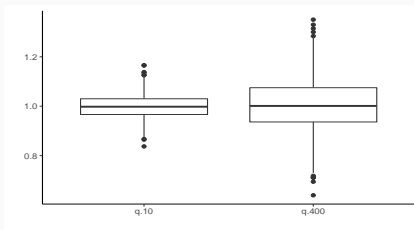
Illustration

- On génère des données $(x_i, y_i), i = 1, \dots, 500$ selon le modèle

$$Y = 1X_1 + 0X_2 + \dots + 0X_{q+1} + \varepsilon$$

où $X_2, X_{q+1}, \dots, \varepsilon$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

- On calcule l'estimateur de MCO de β_1 sur 1000 répétitions. On trace les boxplot de ces estimateurs pour $q = 10$ et $q = 400$.



Conclusion

Plus de variance (donc moins de précision) lorsque le nombre de variables inutiles augmente.

- Lorsque le nombre de variables d est grand, les estimateurs des moindres carrés du modèle linéaire

$$Y = \beta_1 X_1 + \dots + \beta_d X_d + \varepsilon$$

possèdent généralement une grande variance.

- Lorsque le nombre de variables d est grand, les estimateurs des moindres carrés du modèle linéaire

$$Y = \beta_1 X_1 + \dots + \beta_d X_d + \varepsilon$$

possèdent généralement une grande variance.

Idée des méthodes pénalisés

- Contraindre la valeur des estimateurs des moindres carrés de manière à réduire la variance (quitte à augmenter un peu le biais).

- Lorsque le nombre de variables d est grand, les estimateurs des moindres carrés du modèle linéaire

$$Y = \beta_1 X_1 + \dots + \beta_d X_d + \varepsilon$$

possèdent généralement une grande variance.

Idée des méthodes pénalisés

- Contraindre la valeur des estimateurs des moindres carrés de manière à réduire la variance (quitte à augmenter un peu le biais).
- Comment ? En imposant une contrainte sur la valeur des estimateurs des moindres carrés :

$$\hat{\beta}^{pen} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n \left(y_i - \sum_{j=1}^d x_{ij} \beta_j \right)^2$$

sous la contrainte $\|\beta\|_? \leq t$.

- Quelle **norme** utiliser pour la contrainte ?

- Quelle **norme** utiliser pour la contrainte ?
- **Existence/unicité** des estimateurs ? **Solutions explicites** du problème d'optimisation ?

- Quelle **norme** utiliser pour la contrainte ?
- **Existence/unicité** des estimateurs ? **Solutions explicites** du problème d'optimisation ?
- Comment **choisir t** ?
 - t petit \implies estimateurs **contraints** (proche de 0) ;
 - t grand \implies estimateurs des **moindres carrés** (non pénalisés).

- Cas similaire déjà vu pour LDA.
- **Modèle standard LDA** : $\mathcal{L}(X|Y = k) = \mathcal{N}(\mu_k, \Sigma)$.
- μ_k et Σ sont généralement estimés pour les **moyennes et matrice de covariance empiriques**.

- Cas similaire déjà vu pour LDA.
- **Modèle standard LDA** : $\mathcal{L}(X|Y = k) = \mathcal{N}(\mu_k, \Sigma)$.
- μ_k et Σ sont généralement estimés pour les **moyennes et matrice de covariance empiriques**.

LDA régularisée

On **régularise** la matrice de covariance en **augmentant les valeurs de la diagonale**

$$(1 - \gamma)\hat{\Sigma} + \gamma\hat{\sigma}^2 I_p.$$

Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie

- La **régression ridge** consiste à minimiser le critère des moindres carrés pénalisé par la norme 2 des coefficients.

Définition

1. Les **estimateurs ridge** $\hat{\beta}^R$ s'obtiennent en minimisant

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^d x_{ij} \beta_j \right)^2 \quad \text{sous la contrainte} \quad \sum_{j=1}^d \beta_j^2 \leq t \quad (1)$$

- La **régression ridge** consiste à minimiser le critère des moindres carrés pénalisé par la norme 2 des coefficients.

Définition

1. Les **estimateurs ridge** $\hat{\beta}^R$ s'obtiennent en minimisant

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^d x_{ij} \beta_j \right)^2 \quad \text{sous la contrainte} \quad \sum_{j=1}^d \beta_j^2 \leq t \quad (1)$$

2. ou de façon **équivalente**

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

Quelques remarques

- Les définitions (1) et (2) sont **équivalentes** dans le sens où pour tout t il existe un unique λ tels que les solutions aux deux problèmes d'optimisation **coïncident**.

Quelques remarques

- Les définitions (1) et (2) sont **équivalentes** dans le sens où pour tout t il existe un unique λ tels que les solutions aux deux problèmes d'optimisation **coïncident**.
- La **constante** β_0 n'entre généralement **pas** dans la **pénalité**.

Quelques remarques

- Les définitions (1) et (2) sont **équivalentes** dans le sens où pour tout t il existe un unique λ tels que les solutions aux deux problèmes d'optimisation **coïncident**.
- La **constante** β_0 n'entre généralement **pas** dans la **pénalité**.
- L'estimateur **dépend** bien entendu du paramètre t (ou λ) :
$$\hat{\beta}^R = \hat{\beta}^R(t) = \hat{\beta}^R(\lambda).$$

Quelques remarques

- Les définitions (1) et (2) sont **équivalentes** dans le sens où pour tout t il existe un unique λ tels que les solutions aux deux problèmes d'optimisation **coïncident**.
- La **constante** β_0 n'entre généralement **pas** dans la **pénalité**.
- L'estimateur **dépend** bien entendu du paramètre t (ou λ) :
$$\hat{\beta}^R = \hat{\beta}^R(t) = \hat{\beta}^R(\lambda).$$
- Les variables explicatives sont le plus souvent **réduites** pour **éviter les problèmes d'échelle** dans la pénalité.

Un exemple

- Chez des individus atteints du cancer de la prostate, on cherche à expliquer le **niveau d'un anticorps** par 8 variables cliniques.
- Les données ont été mesurées sur 100 individus et disponible à l'url <http://statweb.stanford.edu/~tibs/ElemStatLearn/>

Un exemple

- Chez des individus atteints du cancer de la prostate, on cherche à expliquer le **niveau d'un anticorps** par 8 variables cliniques.
- Les données ont été mesurées sur 100 individus et disponible à l'url <http://statweb.stanford.edu/~tibs/ElemStatLearn/>
- Il existe **plusieurs fonctions et packages** qui permettent de faire de la régression pénalisée sur R. Nous présentons ici **glmnet**.

- Importation

```
> prostate.data <- read.table("prostate.data.txt") %>%  
+   filter(train==TRUE) %>% select(-train)  
> names(prostate.data)  
## [1] "lcavol" "lweight" "age"      "lbph"      "svi"      "lcp"      "gleason"  
## [8] "pgg45"  "lpsa"
```

- Importation

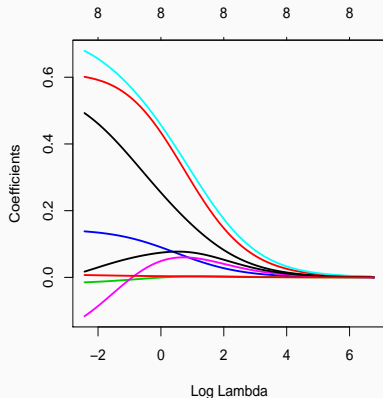
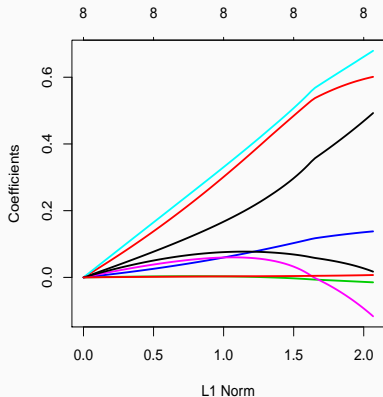
```
> prostate.data <- read.table("prostate.data.txt") %>%  
+   filter(train==TRUE) %>% select(-train)  
> names(prostate.data)  
## [1] "lcavol" "lweight" "age"      "lbph"      "svi"      "lcp"      "gleason"  
## [8] "pgg45"  "lpsa"
```

- `glmnet` n'accepte pas d'objet `formule`. Il faut spécifier la `matrice` des X et le `vecteur` des Y :

```
> prostate.X <- model.matrix(lpsa~.,data=prostate.data)[,-1]
```


Ridge avec glmnet

```
> library(glmnet)
> reg.ridge <- glmnet(prostate.X,prostate.data[,9],alpha=0)
> plot(reg.ridge,lwd=2)
> plot(reg.ridge,lwd=2,xvar="lambda")
```



Propriétés des estimateurs ridge

Propriétés

1. Lorsque les variables explicatives sont **centrée-réduites**, l'estimateur Ridge solution de (2) s'écrit

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

2. On déduit

$$\text{biais}(\hat{\beta}^R) = -\lambda(\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \beta$$

et

$$\mathbf{V}(\hat{\beta}^R) = \sigma^2 (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \mathbb{X}^t \mathbb{X} (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1}.$$

Propriétés des estimateurs ridge

Propriétés

1. Lorsque les variables explicatives sont **centrée-réduites**, l'estimateur Ridge solution de (2) s'écrit

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

2. On déduit

$$\text{biais}(\hat{\beta}^R) = -\lambda(\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \beta$$

et

$$\mathbf{V}(\hat{\beta}^R) = \sigma^2 (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \mathbb{X}^t \mathbb{X} (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1}.$$

Commentaires

- Si $\lambda = 0$, on retrouve le biais et la variance de l'estimateur des **MCO**.
- $\lambda \nearrow \implies \text{biais} \nearrow$ et variance \searrow et réciproquement lorsque $\lambda \searrow$.

- Il est **crucial** : si $\lambda \approx 0$ alors $\hat{\beta}^R \approx \hat{\beta}^{MCO}$, si λ "grand" alors $\hat{\beta}^R \approx 0$.

Choix de λ

- Il est **crucial** : si $\lambda \approx 0$ alors $\hat{\beta}^R \approx \hat{\beta}^{MCO}$, si λ "grand" alors $\hat{\beta}^R \approx 0$.
- Le choix de λ se fait le plus souvent de façon "classique" :
 1. **Estimation d'un critère** de choix de modèle pour toutes les valeurs de λ ;

Choix de λ

- Il est **crucial** : si $\lambda \approx 0$ alors $\hat{\beta}^R \approx \hat{\beta}^{MCO}$, si λ "grand" alors $\hat{\beta}^R \approx 0$.
- Le choix de λ se fait le plus souvent de façon "classique" :
 1. **Estimation d'un critère** de choix de modèle pour toutes les valeurs de λ ;
 2. Choix du λ qui **minimise** le critère estimé.

Choix de λ

- Il est **crucial** : si $\lambda \approx 0$ alors $\hat{\beta}^R \approx \hat{\beta}^{MCO}$, si λ "grand" alors $\hat{\beta}^R \approx 0$.
- Le choix de λ se fait le plus souvent de façon "classique" :
 1. **Estimation d'un critère** de choix de modèle pour toutes les valeurs de λ ;
 2. Choix du λ qui **minimise** le critère estimé.
- **Exemple** : la fonction `cv.glmnet` choisit la valeur de λ qui minimise l'erreur quadratique moyenne

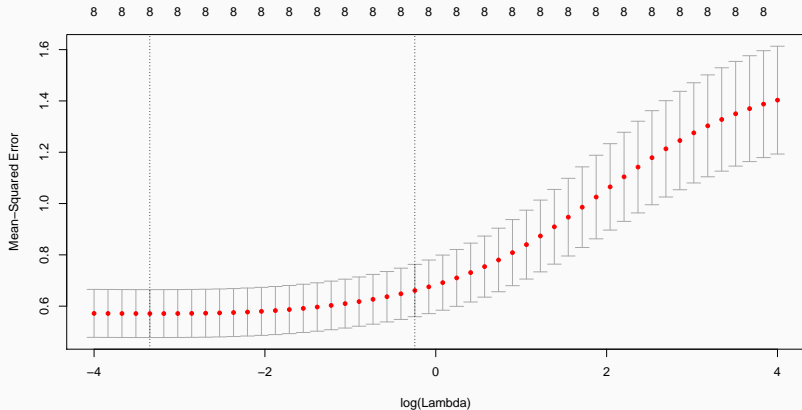
$$\mathbb{E}[(Y - X^t \hat{\beta}^R(\lambda))^2]$$

estimée par **validation croisée**.

```

> set.seed(1234)
> reg.cvridge <- cv.glmnet(prostate.X,prostate.data[,9],alpha=0,
+                          lambda=exp(seq(-4,4,length=50)))
> bestlam <- reg.cvridge$lambda.min
> bestlam
## [1] 0.03519192
> plot(reg.cvridge)

```



Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie

- La **régression lasso** consiste à minimiser le critère des moindres carrés pénalisé par la norme 1 des coefficients.

Définition [**Tibshirani, 1996**]

1. Les **estimateurs lasso** $\hat{\beta}^L$ s'obtiennent en minimisant

$$\sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^d X_{ij} \beta_j \right)^2 \quad \text{sous la contrainte} \quad \sum_{j=1}^d |\beta_j| \leq t \quad (3)$$

2. ou de façon **équivalente**

$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^d X_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^d |\beta_j| \right\}. \quad (4)$$

Comparaison Ridge-Lasso

- Dans le cas où la matrice \mathbb{X} est **orthonormée**, on a une **écriture explicite** pour les estimateurs ridge et lasso.

Comparaison Ridge-Lasso

- Dans le cas où la matrice \mathbb{X} est **orthonormée**, on a une **écriture explicite** pour les estimateurs ridge et lasso.

Propriété

Si la matrice de design \mathbb{X} est orthonormée, alors

$$\hat{\beta}_j^R = \frac{\hat{\beta}_j}{1 + \lambda} \quad \text{et} \quad \hat{\beta}_j^L = \text{signe}(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$$

où $\hat{\beta}_j$ est l'estimateur MCO de β_j .

Comparaison Ridge-Lasso

- Dans le cas où la matrice \mathbb{X} est **orthonormée**, on a une **écriture explicite** pour les estimateurs ridge et lasso.

Propriété

Si la matrice de design \mathbb{X} est orthonormée, alors

$$\hat{\beta}_j^R = \frac{\hat{\beta}_j}{1 + \lambda} \quad \text{et} \quad \hat{\beta}_j^L = \text{signe}(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$$

où $\hat{\beta}_j$ est l'estimateur MCO de β_j .

Commentaires

- **Ridge** "diminue" l'estimateur MCO de façon **proportionnelle** ;

Comparaison Ridge-Lasso

- Dans le cas où la matrice \mathbb{X} est **orthonormée**, on a une **écriture explicite** pour les estimateurs ridge et lasso.

Propriété

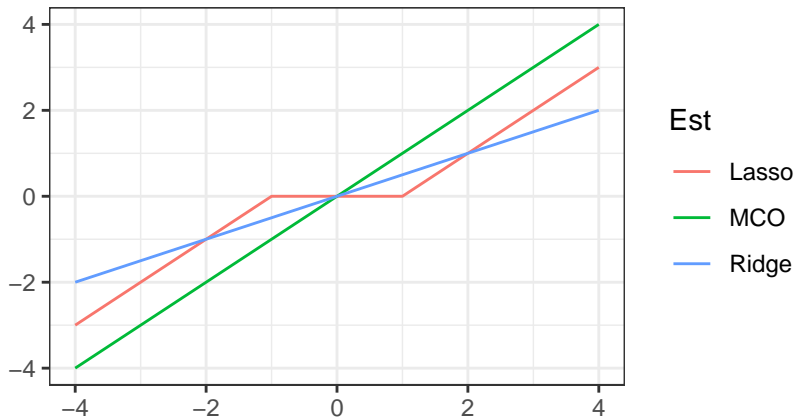
Si la matrice de design \mathbb{X} est orthonormée, alors

$$\hat{\beta}_j^R = \frac{\hat{\beta}_j}{1 + \lambda} \quad \text{et} \quad \hat{\beta}_j^L = \text{signe}(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$$

où $\hat{\beta}_j$ est l'estimateur MCO de β_j .

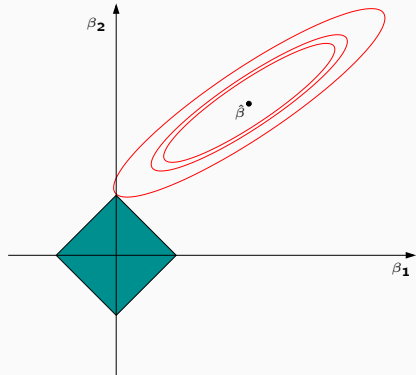
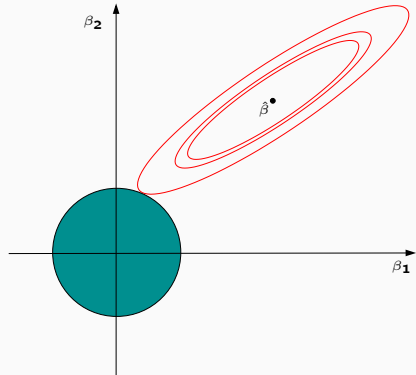
Commentaires

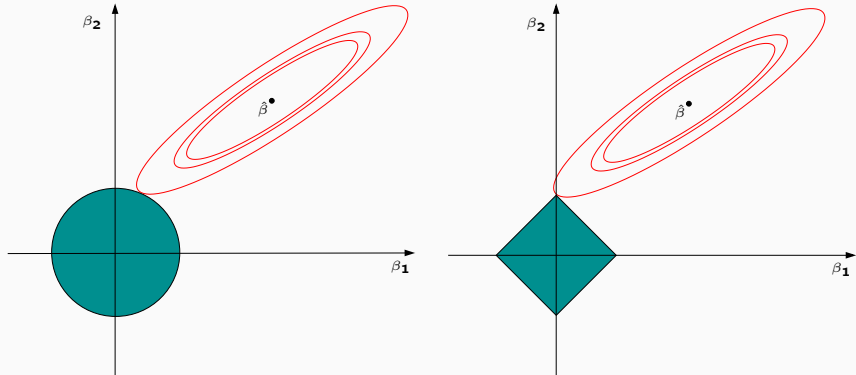
- **Ridge** "diminue" l'estimateur MCO de façon **proportionnelle** ;
- **Lasso** **translate et tronque** l'estimateur MCO (lorsque ce dernier est petit).



Conclusion

Le lasso va avoir tendance à "mettre" des coefficients à 0 et donc à faire de la sélection de variables.





Remarque

Ces approches reviennent (d'une certaine façon) à projeter l'estimateur des MCO sur les boules unités associées à

1. la norme 2 pour la régression ridge ;
2. la norme 1 pour le lasso.

Quelques remarques

- Comme pour la régression ridge :
 - on préfère souvent réduire la matrice de design avant d'effectuer la régression lasso ;

Quelques remarques

- Comme pour la régression ridge :
 - on préfère souvent réduire la matrice de design avant d'effectuer la régression lasso ;
 - Le choix de λ est crucial (il est le plus souvent sélectionné en minimisant un critère empirique).

Quelques remarques

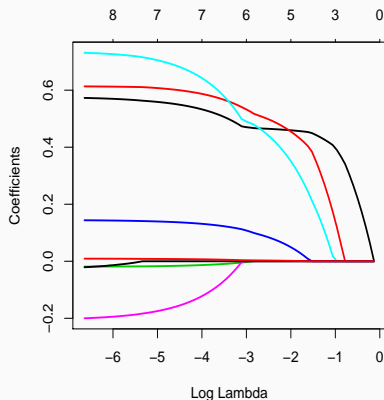
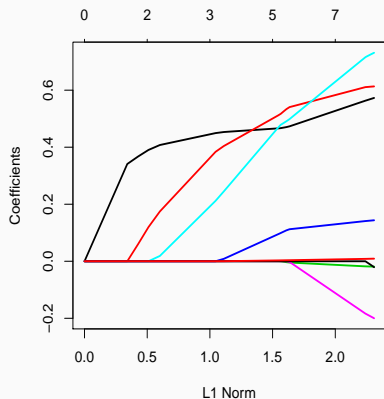
- Comme pour la régression ridge :
 - on préfère souvent **réduire la matrice de design** avant d'effectuer la régression lasso ;
 - Le choix de λ est **crucial** (il est le plus souvent sélectionné en minimisant un critère empirique).
 - $\lambda \nearrow \implies$ biais \nearrow et variance \searrow et réciproquement lorsque $\lambda \searrow$.

Quelques remarques

- Comme pour la régression ridge :
 - on préfère souvent **réduire la matrice de design** avant d'effectuer la régression lasso ;
 - Le choix de λ est **crucial** (il est le plus souvent sélectionné en minimisant un critère empirique).
 - $\lambda \nearrow \implies$ biais \nearrow et variance \searrow et réciproquement lorsque $\lambda \searrow$.
- **MAIS**, contrairement à ridge : $\lambda \nearrow \implies$ **le nombre de coefficients nuls augmente** ([Bühlmann and van de Geer, 2011]).

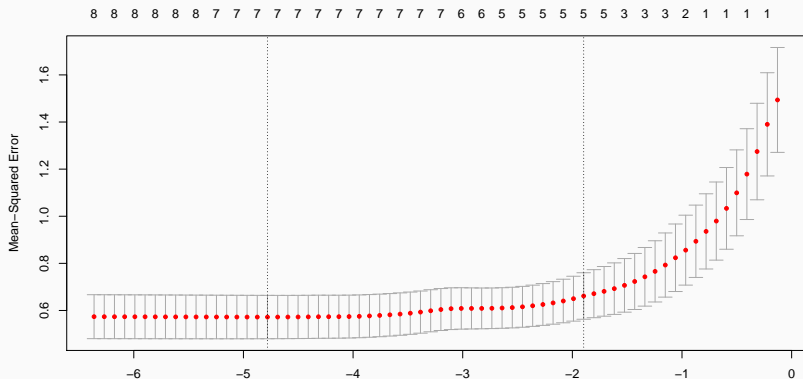
Le coin R

```
> reg.lasso <- glmnet(prostate.X,prostate.data[,9],alpha=1)
> plot(reg.lasso,lwd=2)
> plot(reg.lasso,lwd=2,xvar="lambda")
```



Sélection de λ

```
> set.seed(1234)
> reg.cvlasso <- cv.glmnet(prostate.X,prostate.data[,9],alpha=1)
> bestlam <- reg.cvlasso$lambda.min
> bestlam
## [1] 0.008389339
> plot(reg.cvlasso)
```



- Il existe plusieurs façons de résoudre le problème numérique d'optimisation lasso (ou ridge).
- Un des plus utilisés est l'algorithme de descente de coordonnées [Hastie et al., 2015].

- Il existe plusieurs façons de résoudre le problème numérique d'optimisation lasso (ou ridge).
- Un des plus utilisés est l'algorithme de descente de coordonnées [Hastie et al., 2015].
- On considère le problème lasso

$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^d X_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^d |\beta_j| \right\}$$

avec les variables explicatives centrées-réduites (pour simplifier).

1. **Initialisation** : $\hat{\beta}_0 = \bar{y}$, $\hat{\beta}_j = \dots, j = 1, \dots, d$.
2. Répéter jusqu'à convergence :
Pour $j = 1, \dots, d$:
 - 2.1 Calculer les **résidus partiels** $r_i^{(j)} = y_i - \sum_{k \neq j} x_{ik} \hat{\beta}_k$;
 - 2.2 Faire la **régression simple** des y_i contre $r_i^{(j)} \Rightarrow \tilde{\beta}_j$;
 - 2.3 **Mettre à jour** $\hat{\beta}_j = \text{signe}(\tilde{\beta}_j)(|\tilde{\beta}_j| - \lambda)_+$
3. **Retourner** : $\hat{\beta}_j, j = 1, \dots, d$.

Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie

Différentes pénalités

- Les approches **ridge** et **lasso** diffèrent uniquement au niveau de la **pénalité** ajoutée au critère des moindres carrés.
- **Norme 2** pour **ridge** et **norme 1** pour le **lasso**.

Différentes pénalités

- Les approches **ridge** et **lasso** diffèrent uniquement au niveau de la **pénalité** ajoutée au critère des moindres carrés.
- **Norme 2** pour **ridge** et **norme 1** pour le **lasso**.
- Il existe tout un tas d'**autres stratégies** de pénalisations.
- Nous en présentons quelques unes dans cette partie.
- On pourra consulter [[Hastie et al., 2015](#)] pour plus de détails.

- [Zou and Hastie, 2005] ont proposé de combiner les approches ridge et lasso en proposant une pénalité (appelée elastic net) de la forme

$$\lambda \sum_{j=1}^d ((1 - \alpha)\beta_j^2 + \alpha|\beta_j|)$$

où $\alpha \in [0, 1]$.

- [Zou and Hastie, 2005] ont proposé de combiner les approches ridge et lasso en proposant une pénalité (appelée elastic net) de la forme

$$\lambda \sum_{j=1}^d ((1 - \alpha)\beta_j^2 + \alpha|\beta_j|)$$

où $\alpha \in [0, 1]$.

- Le paramètre α définit le compromis ridge/lasso :
 - $\alpha = 1 \implies$ Lasso ;
 - $\alpha = 0 \implies$ Ridge ;

- [Zou and Hastie, 2005] ont proposé de combiner les approches ridge et lasso en proposant une pénalité (appelée elastic net) de la forme

$$\lambda \sum_{j=1}^d ((1 - \alpha)\beta_j^2 + \alpha|\beta_j|)$$

où $\alpha \in [0, 1]$.

- Le paramètre α définit le compromis ridge/lasso :
 - $\alpha = 1 \implies$ Lasso ;
 - $\alpha = 0 \implies$ Ridge ;
 - Ce paramètre correspond (évidemment) à l'argument `alpha` de la fonction `glmnet`.

- [Zou and Hastie, 2005] ont proposé de combiner les approches ridge et lasso en proposant une pénalité (appelée elastic net) de la forme

$$\lambda \sum_{j=1}^d ((1 - \alpha)\beta_j^2 + \alpha|\beta_j|)$$

où $\alpha \in [0, 1]$.

- Le paramètre α définit le compromis ridge/lasso :
 - $\alpha = 1 \implies$ Lasso ;
 - $\alpha = 0 \implies$ Ridge ;
 - Ce paramètre correspond (évidemment) à l'argument `alpha` de la fonction `glmnet`.
- **Avantage** : on a plus de flexibilité car la pénalité elastic net propose une gamme de modèles beaucoup plus large que lasso et ridge ;

- [Zou and Hastie, 2005] ont proposé de combiner les approches ridge et lasso en proposant une pénalité (appelée elastic net) de la forme

$$\lambda \sum_{j=1}^d ((1 - \alpha)\beta_j^2 + \alpha|\beta_j|)$$

où $\alpha \in [0, 1]$.

- Le paramètre α définit le compromis ridge/lasso :
 - $\alpha = 1 \implies$ Lasso ;
 - $\alpha = 0 \implies$ Ridge ;
 - Ce paramètre correspond (évidemment) à l'argument alpha de la fonction glmnet.
- **Avantage** : on a plus de flexibilité car la pénalité elastic net propose une gamme de modèles beaucoup plus large que lasso et ridge ;
- **Inconvénient** : en plus du λ il faut aussi sélectionner le α !

- Dans certaines applications, les variables explicatives appartiennent à des groupes de variables prédéfinis.

Group Lasso

- Dans certaines applications, les variables explicatives appartiennent à des groupes de variables prédéfinis.
- Nécessité de "shrinker" ou sélectionner les variables par groupe.

- Dans certaines applications, les variables **explicatives** appartiennent à des **groupes de variables** prédéfinis.
- Nécessité de "**shrinker**" ou **sélectionner** les variables **par groupe**.

Exemple : variables qualitatives

- 2 variables explicatives qualitatives X_1 et X_2 et une variable explicative continue X_3 .
- Le **modèle** s'écrit

$$Y = \beta_0 + \beta_1 \mathbf{1}_{X_1=A} + \beta_2 \mathbf{1}_{X_1=B} + \beta_3 \mathbf{1}_{X_1=C} \\ + \beta_4 \mathbf{1}_{X_2=D} + \beta_5 \mathbf{1}_{X_2=E} + \beta_6 \mathbf{1}_{X_2=F} + \beta_7 \mathbf{1}_{X_2=G} + \beta_8 X_3 + \varepsilon$$

muni des contraintes $\beta_1 = \beta_4 = 0$.

Group Lasso

- Dans certaines applications, les variables **explicatives** appartiennent à des **groupes de variables** prédéfinis.
- Nécessité de "**shrinker**" ou **sélectionner** les variables **par groupe**.

Exemple : variables qualitatives

- 2 variables explicatives qualitatives X_1 et X_2 et une variable explicative continue X_3 .
- Le **modèle** s'écrit

$$Y = \beta_0 + \beta_1 \mathbf{1}_{X_1=A} + \beta_2 \mathbf{1}_{X_1=B} + \beta_3 \mathbf{1}_{X_1=C} \\ + \beta_4 \mathbf{1}_{X_2=D} + \beta_5 \mathbf{1}_{X_2=E} + \beta_6 \mathbf{1}_{X_2=F} + \beta_7 \mathbf{1}_{X_2=G} + \beta_8 X_3 + \varepsilon$$

muni des contraintes $\beta_1 = \beta_4 = 0$.

- 3 groupes : $\mathbf{X}_1 = (\mathbf{1}_{X_1=B}, \mathbf{1}_{X_1=C})$, $\mathbf{X}_2 = (\mathbf{1}_{X_2=E}, \mathbf{1}_{X_2=F}, \mathbf{1}_{X_2=G})$ et $\mathbf{X}_3 = X_3$.

Définition

En présence de d variables réparties en L groupes $\mathbf{X}_1, \dots, \mathbf{X}_L$ de cardinal d_1, \dots, d_L . On note $\beta_\ell, \ell = 1, \dots, L$ le vecteur des coefficients associé au groupe \mathbf{X}_ℓ . Les **estimateurs group-lasso** s'obtiennent en minimisant le critère

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{\ell=1}^L \mathbf{x}_{i\ell} \beta_\ell \right)^2 + \lambda \sum_{\ell=1}^L \sqrt{d_\ell} \|\beta_\ell\|_2$$

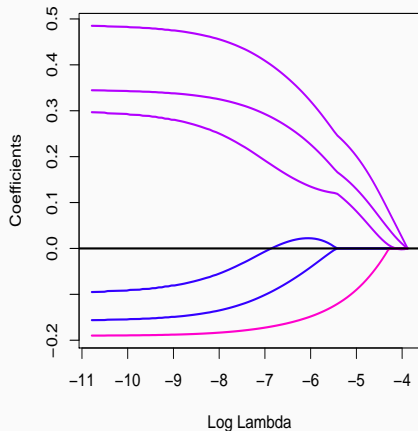
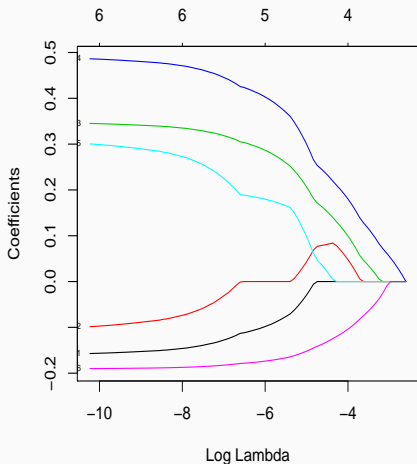
Remarque

Puisque $\|\beta_\ell\|_2 = 0$ ssi $\beta_{\ell 1} = \dots = \beta_{\ell d_\ell} = 0$, cette procédure encourage la **mise à zéro** des coefficients d'un **même groupe**.

Le coin R

- La fonction `gglasso` du package `gglasso` permet de faire du **groupe lasso** sur R.

```
> summary(donnees)
##      X1      X2      X3      Y
##  A:60   E:40   Min.   :0.009496   Min.   : -3.23315
##  B:90   F:60   1st Qu.:0.237935   1st Qu.: -0.50404
##  C:50   G:55   Median :0.485563   Median :  0.16759
##                H:45   Mean   :0.483286   Mean    :  0.09792
##                3rd Qu.:0.734949   3rd Qu.:  0.66918
##                Max.    :0.998741   Max.     :  3.04377
> D <- model.matrix(Y~.,data=donnees)[,-1]
> model <- glmnet(D,Y,alpha=1)
> plot(model,label=TRUE,xvar="lambda")
> groupe <- c(1,1,2,2,2,3)
> library(gglasso)
> model1 <- gglasso(D,Y,group=groupe)
> plot(model1)
```

Remarque

Les coefficients **s'annulent par groupe** lorsque λ augmente (graphe de droite).

Sparse group lasso

- La **norme 2** de la pénalité group-lasso implique que, généralement, tous les coefficients d'un groupe sont **tous nuls** ou **tous non nuls**.
- Dans certains cas, il peut être intéressant de mettre de la **sparsité** dans les groupes aussi. Comment ?

Sparse group lasso

- La **norme 2** de la pénalité group-lasso implique que, généralement, tous les coefficients d'un groupe sont **tous nuls** ou **tous non nuls**.
- Dans certains cas, il peut être intéressant de mettre de la **sparsité** dans les groupes aussi. Comment ?
- En ajoutant **la norme 1** dans la pénalité.

Sparse group lasso

- La **norme 2** de la pénalité group-lasso implique que, généralement, tous les coefficients d'un groupe sont **tous nuls** ou **tous non nuls**.
- Dans certains cas, il peut être intéressant de mettre de la **sparsité** dans les groupes aussi. Comment ?
- En ajoutant **la norme 1** dans la pénalité.

Pénalité parse group lasso

$$\lambda \sum_{\ell=1}^L (1 - \alpha) \|\beta_{\ell}\|_2 + \alpha \|\beta_{\ell}\|_1.$$

- Sur **R** : package **SGL**.

Fused lasso

- Utile pour prendre en compte la **spatialité des données**.
- **Idée** : deux coefficients successifs doivent être proches/

Pénalité fused lasso

$$\lambda_1 \sum_{j=1}^d |\beta_j|$$

Fused lasso

- Utile pour prendre en compte la **spatialité des données**.
- **Idée** : deux coefficients successifs doivent être proches/

Pénalité fused lasso

$$\lambda_1 \sum_{j=1}^d |\beta_j| + \lambda_2 \sum_{j=2}^d |\beta_{j+1} - \beta_j|$$

qui peut se re-paramétriser en

$$\sum_{j=2}^d |\beta_{j+1} - \beta_j|.$$

- Sur **R** : package **genlasso**.

Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie

- Les méthodes **ridge et lasso** ont été présentées dans un cadre de régression linéaire.
- Ces techniques s'adaptent directement à la **régression logistique** $\mathcal{Y} = \{-1, 1\}$.

- Les méthodes **ridge et lasso** ont été présentées dans un cadre de régression linéaire.
- Ces techniques d'adaptent directement à la **régression logistique** $\mathcal{Y} = \{-1, 1\}$.
- Les **pénalités** sont **identiques**.
- **Seul changement** : le critère moindre carré est remplacé par la déviance \implies ce qui revient à **minimiser l'opposé de la vraisemblance plus la pénalité**.

Définition

On note $\tilde{y}_i = (y_i + 1)/2$.

- On appelle **estimateur ridge** en régression logistique l'estimateur

$$\hat{\beta}^R = \operatorname{argmin}_{\beta} \left\{ - \sum_{i=1}^n (\tilde{y}_i x_i^t \beta - \log(1 + \exp(x_i^t \beta))) + \lambda \sum_{j=1}^d \beta_j^2 \right\}.$$

- On appelle **estimateur lasso** en régression logistique l'estimateur

$$\hat{\beta}^L = \operatorname{argmin}_{\beta} \left\{ - \sum_{i=1}^n (\tilde{y}_i x_i^t \beta - \log(1 + \exp(x_i^t \beta))) + \lambda \sum_{j=1}^d |\beta_j| \right\}.$$

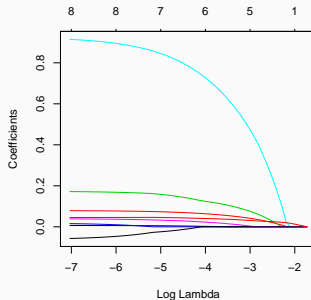
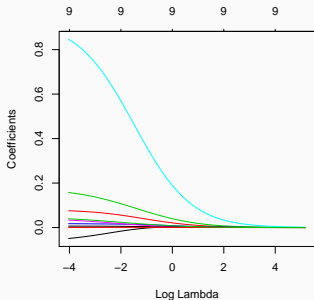
- Pour faire du ridge ou lasso en logistique, il suffit d'ajouter l'argument `family=binomial` dans `glmnet`.
- Tout reste identique pour le reste (tracé du chemin des coefficients, choix du λ ...).
- Exemple : données SAheart

```
> head(SAheart)
```

##	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
## 1	160	12.00	5.73	23.11	Present	49	25.30	97.20	52	1
## 2	144	0.01	4.41	28.61	Absent	55	28.87	2.06	63	1
## 3	118	0.08	3.48	32.28	Present	52	29.14	3.81	46	0
## 4	170	7.50	6.41	38.03	Present	51	31.99	24.26	58	1
## 5	134	13.60	3.50	27.78	Present	60	25.99	57.34	49	1
## 6	132	6.20	6.47	36.21	Present	62	30.77	14.14	45	0

- On obtient les chemins de régularisation ridge et lasso avec les commandes suivantes :

```
> SAheart.X <- model.matrix(chd~.,data=SAheart)
> log.ridge <- glmnet(SAheart.X,SAheart$chd,family="binomial",alpha=0)
> log.lasso <- glmnet(SAheart.X,SAheart$chd,family="binomial",alpha=1)
> plot(log.ridge,xvar="lambda")
> plot(log.lasso,xvar="lambda")
```



Régression ridge

Régression Lasso

Variantes de ridge/lasso

Discrimination binaire

Bibliographie



Bühlmann, P. and van de Geer, S. (2011).

Statistics for high-dimensional data.

Springer.



Hastie, T., Tibshirani, R., and Friedman, J. (2009).

The Elements of Statistical Learning : Data Mining, Inference, and Prediction.

Springer, second edition.



Hastie, T., Tibshirani, R., and Wainwright, M. (2015).
Statistical Learning with Sparsity : The Lasso and Generalizations.

CRC Press.

[https:](https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf)

[//web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf](https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf).



Tibshirani, R. (1996).

Regression shrinkage and selection via the lasso.

Journal of the Royal Statistical Society, Series B, 58 :267–288.



Zou, H. and Hastie, T. (2005).

Regularization and variable selection via the elastic net.

Journal of the Royal Statistical Society, Series B, 67 :301–320.

Quatrième partie IV

Modèle additif

Présentation du modèle

Estimation : l'algorithme du backfitting

Lisseurs

Bibliographie

- Toujours le même : modèle de régression $\implies (x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$ tels que

$$y_i = m(x_i) + \varepsilon.$$

- Rappels :
 - Modèle linéaire (paramétrique) : peu flexible mais convergence rapide ;
 - Modèle non paramétrique : flexible mais convergence lente.

- Toujours le même : modèle de **régression** $\implies (x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$ tels que

$$y_i = m(x_i) + \varepsilon.$$

- **Rappels** :
 - **Modèle linéaire (paramétrique)** : peu flexible mais convergence rapide ;
 - **Modèle non paramétrique** : flexible mais convergence lente.

Idée

Conserver des **vitesses de convergence raisonnables** en **allégeant** l'hypothèse de **linéarité** en les covariables.

Présentation du modèle

Estimation : l'algorithme du backfitting

Lisseurs

Bibliographie

Le modèle additif

Définition

Un modèle de régression est **additif** si la fonction de régression de y sur $x = (x_1, \dots, x_d)$ s'écrit

$$m(x) = \alpha + g_1(x_1) + \dots + g_d(x_d).$$

où $\alpha \in \mathbb{R}$ est inconnu et les fonction g_1, \dots, g_d sont inconnues (et donc **à estimer**).

Le modèle additif

Définition

Un modèle de régression est **additif** si la fonction de régression de y sur $x = (x_1, \dots, x_d)$ s'écrit

$$m(x) = \alpha + g_1(x_1) + \dots + g_d(x_d).$$

où $\alpha \in \mathbb{R}$ est inconnu et les fonction g_1, \dots, g_d sont inconnues (et donc **à estimer**).

Remarques

- Hypothèse de **structure additive** en les covariables comme dans le modèle linéaire mais...

Le modèle additif

Définition

Un modèle de régression est **additif** si la fonction de régression de y sur $x = (x_1, \dots, x_d)$ s'écrit

$$m(x) = \alpha + g_1(x_1) + \dots + g_d(x_d).$$

où $\alpha \in \mathbb{R}$ est inconnu et les fonction g_1, \dots, g_d sont inconnues (et donc **à estimer**).

Remarques

- Hypothèse de **structure additive** en les covariables comme dans le modèle linéaire mais...
- pas d'hypothèse de **linéarité**.

- Pour l'identifiabilité du modèle on suppose souvent $E[Y] = \alpha$ et $E[g_j(x_j)] = 0$.

- Pour l'identifiabilité du modèle on suppose souvent $E[Y] = \alpha$ et $E[g_j(x_j)] = 0$.
- On doit estimer d fonctions \implies cadre non paramétrique mais ...

- Pour l'identifiabilité du modèle on suppose souvent $E[Y] = \alpha$ et $E[g_j(x_j)] = 0$.
- On doit estimer d fonctions \implies cadre non paramétrique mais ...
- Ces fonctions sont univariées !

- Pour l'identifiabilité du modèle on suppose souvent $E[Y] = \alpha$ et $E[g_j(x_j)] = 0$.
- On doit estimer d fonctions \implies cadre non paramétrique mais ...
- Ces fonctions sont univariées !

Conclusion

Si ces fonctions sont convenablement estimées, on peut espérer obtenir des vitesses de convergence proches des vitesses non paramétrique en dimension 1 (qui ne sont pas si mauvaises !).

Présentation du modèle

Estimation : l'algorithme du backfitting

Lisseurs

Bibliographie

Retour au modèle linéaire

- $m(X) = \mathbf{E}[Y|X] = \beta_0 + \beta_1 X_1 + \dots + \beta_d X_d = \sum_{j=0}^d \beta_j X_j$.
- En conditionnant par X_k on obtient :

$$\begin{aligned}\mathbf{E}[Y|X_k = x_k] &= \mathbf{E}[\mathbf{E}[Y|X]|X_k = x_k] \\ &= \mathbf{E}\left[\sum_{j=0}^d \beta_j X_j \middle| X_k = x_k\right] \\ &= \beta_k x_k + \mathbf{E}\left[\sum_{j \neq k} \beta_j X_j \middle| X_k = x_k\right].\end{aligned}$$

Retour au modèle linéaire

- $m(X) = \mathbf{E}[Y|X] = \beta_0 + \beta_1 X_1 + \dots + \beta_d X_d = \sum_{j=0}^d \beta_j X_j$.
- En **conditionnant par X_k** on obtient :

$$\begin{aligned}\mathbf{E}[Y|X_k = x_k] &= \mathbf{E}[\mathbf{E}[Y|X]|X_k = x_k] \\ &= \mathbf{E}\left[\sum_{j=0}^d \beta_j X_j \middle| X_k = x_k\right] \\ &= \beta_k x_k + \mathbf{E}\left[\sum_{j \neq k} \beta_j X_j \middle| X_k = x_k\right].\end{aligned}$$

Propriété

On pose $Y^{(k)} = Y - \sum_{j \neq k} \beta_j X_j$. Alors

$$\mathbf{E}[Y^{(k)}|X_k = x_k] = \beta_k x_k.$$

- **Idée** : estimer β_k en faisant la régression (univariée) de $Y^{(k)}$ sur X_k .
- **Problème** :

- **Idée** : estimer β_k en faisant la régression (univariée) de $Y^{(k)}$ sur X_k .
- **Problème** : $Y^{(k)}$ dépend des $\beta_j, j \neq k$ qui sont inconnus !
- **Solution** :

- **Idée** : estimer β_k en faisant la régression (univariée) de $Y^{(k)}$ sur X_k .
- **Problème** : $Y^{(k)}$ dépend des $\beta_j, j \neq k$ qui sont inconnus !
- **Solution** : utiliser un algorithme itératif.

- **Idée** : estimer β_k en faisant la **régression (univariée)** de $Y^{(k)}$ sur X_k .
- **Problème** : $Y^{(k)}$ dépend des $\beta_j, j \neq k$ qui sont **inconnus** !
- **Solution** : utiliser un **algorithme itératif**.

Algorithme : backfitting pour le modèle linéaire

- Initialisation $\hat{\beta} = (0, \dots, 0), \gamma = (1, \dots, 1)$ (dim $d + 1$).
- Tant que $\min_{1 \leq j \leq d+1} (|\hat{\beta}_j - \gamma_j|) > \varepsilon$ (petit)
 1. $\hat{\beta} = \gamma$
 2. Pour $k = 0, \dots, d$:
 - $\mathbb{Y}^{(k)} = \mathbb{Y} - \sum_{j \neq k} \beta_j \mathbb{X}_j$ (résidus partiels)
 - γ_k : coefficient de la régression univariée sans constante de $\mathbb{Y}^{(k)}$ sur \mathbb{X}_k .
- **Retourner** $\hat{\beta}$.

- Cet algorithme converge vers les **estimateurs MCO**.
- **Aucune utilité pratique** puisque ces estimateurs s'obtiennent de façon matricielle !
- Mais...

- Cet algorithme converge vers les **estimateurs MCO**.
- **Aucune utilité pratique** puisque ces estimateurs s'obtiennent de façon matricielle !
- Mais... il ne repose pas sur l'hypothèse de linéarité du modèle, juste sur sa **structure additive**.
- Facilement généralisable au **modèle additif**.

- Retour au **modèle additif**

$$m(X) = \mathbf{E}[Y|X] = \alpha + g_1(X_1) + \dots + g_d(X_d)$$

ou

$$Y = \alpha + g_1(X_1) + \dots + g_d(X_d) + \varepsilon \quad \text{avec} \quad \mathbf{E}[\varepsilon|X] = 0.$$

- Retour au **modèle additif**

$$m(X) = \mathbf{E}[Y|X] = \alpha + g_1(X_1) + \dots + g_d(X_d)$$

ou

$$Y = \alpha + g_1(X_1) + \dots + g_d(X_d) + \varepsilon \quad \text{avec} \quad \mathbf{E}[\varepsilon|X] = 0.$$

Propriété

On pose $Y^{(k)} = Y - \alpha - \sum_{j \neq k} g_j(X_j)$. Alors

$$\mathbf{E}[Y^{(k)}|X_k = x_k] = g_k(x_k).$$

- Retour au **modèle additif**

$$m(X) = \mathbf{E}[Y|X] = \alpha + g_1(X_1) + \dots + g_d(X_d)$$

ou

$$Y = \alpha + g_1(X_1) + \dots + g_d(X_d) + \varepsilon \quad \text{avec} \quad \mathbf{E}[\varepsilon|X] = 0.$$

Propriété

On pose $Y^{(k)} = Y - \alpha - \sum_{j \neq k} g_j(X_j)$. Alors

$$\mathbf{E}[Y^{(k)}|X_k = x_k] = g_k(x_k).$$

Idée

Remplacer l'étape MCO par un **lissage non paramétrique** dans l'algorithme précédent.

L'algorithme du backfitting

1. Initialisation : $\hat{\alpha} = \bar{Y}$, $\hat{g}_k(x_k) = \bar{X}_k$.
2. Pour $k = 1, \dots, d$:
 - $Y^{(k)} = Y - \hat{\alpha} - \sum_{j \neq k} \hat{g}_j(X_j)$ (résidus partiels)
 - \hat{g}_k : lissage non paramétrique de $Y^{(k)}$ sur X_k .
3. Répéter l'étape précédente tant que les \hat{g}_k changent.

L'algorithme du backfitting

1. Initialisation : $\hat{\alpha} = \bar{Y}$, $\hat{g}_k(x_k) = \bar{X}_k$.
2. Pour $k = 1, \dots, d$:
 - $Y^{(k)} = Y - \hat{\alpha} - \sum_{j \neq k} \hat{g}_j(X_j)$ (résidus partiels)
 - \hat{g}_k : lissage non paramétrique de $Y^{(k)}$ sur X_k .
3. Répéter l'étape précédente tant que les \hat{g}_k changent.

Lisseurs non paramétrique

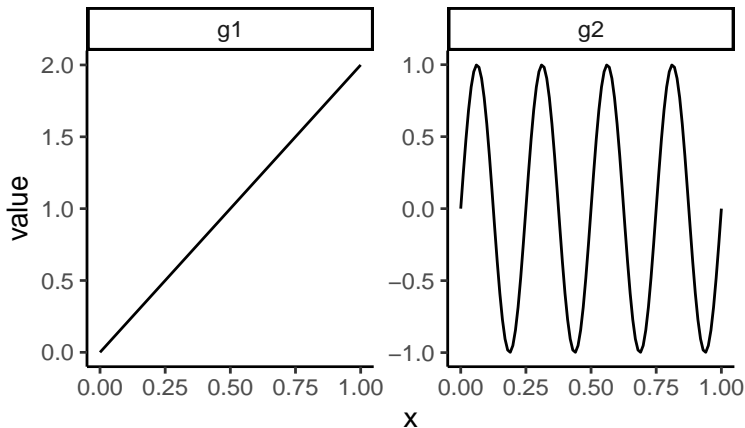
Loess, Nadaraya-Watson, spline de lissage...

Exemple

- On considère le **modèle GAM** :

$$m(X) = 2X_1 + \sin(8\pi X_2)$$

avec X_1 et X_2 indépendantes de lois uniformes sur $[0, 1]$.



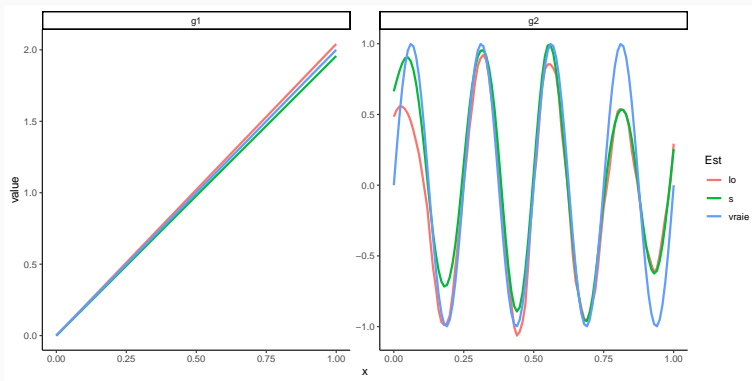
- Sur R, la fonction `gam` du package `gam` permet d'ajuster des modèles additifs :

```
> library(gam)
> model1 <- gam(Y~s(X1,df=1)+s(X2,df=24.579)-1,data=donnees)
> model2 <- gam(Y~lo(X1,span=3)+lo(X2,span=0.15,degree=2)-1,data=donnees)
```

Le coin R

- Sur R, la fonction `gam` du package `gam` permet d'ajuster des modèles additifs :

```
> library(gam)
> model1 <- gam(Y~s(X1,df=1)+s(X2,df=24.579)-1,data=donnees)
> model2 <- gam(Y~lo(X1,span=3)+lo(X2,span=0.15,degree=2)-1,data=donnees)
```



Vitesses de convergence

- Le **backfitting** fournit un estimateur de la fonction de régression

$$\hat{m}(x) = \hat{\alpha} + \hat{g}_1(x_1) + \hat{g}_d(x_d).$$

Théorème [Opsomer, 2000]

Sous certaines hypothèses techniques, notamment les $g_j, j = 1, \dots, d$ de classe C^2 on a

$$\text{Biais}(\hat{m}(x)|X_1, \dots, X_n) = C_1 h^2 + o_p(h^2)$$

et

$$\text{V}(\hat{m}(x)|X_1, \dots, X_n) = \frac{C_2}{nh} + o_p\left(\frac{1}{nh}\right).$$

Vitesses de convergence

- Le **backfitting** fournit un estimateur de la fonction de régression

$$\hat{m}(x) = \hat{\alpha} + \hat{g}_1(x_1) + \hat{g}_d(x_d).$$

Théorème [Opsomer, 2000]

Sous certaines hypothèses techniques, notamment les $g_j, j = 1, \dots, d$ de classe C^2 on a

$$\text{Biais}(\hat{m}(x)|X_1, \dots, X_n) = C_1 h^2 + o_p(h^2)$$

et

$$\text{V}(\hat{m}(x)|X_1, \dots, X_n) = \frac{C_2}{nh} + o_p\left(\frac{1}{nh}\right).$$

Commentaire

Ce sont les **vitesses de convergence classique** des estimateurs **non paramétrique en dimension 1**.

- Les lisseurs non paramétriques dépendent de **paramètres** dont le choix se révèle **crucial** pour la qualité de l'estimation.

- Les lisseurs non paramétriques dépendent de **paramètres** dont le choix se révèle **crucial** pour la qualité de l'estimation.
- **Exemples** : nombre de plus proches voisins, fenêtre de l'estimateur à noyau...

- Les lisseurs non paramétriques dépendent de **paramètres** dont le choix se révèle **crucial** pour la qualité de l'estimation.
- **Exemples** : nombre de plus proches voisins, fenêtre de l'estimateur à noyau...
- Ici encore, il va falloir trouver des **procédures** pour **sélectionner ces paramètres de lissage** dans l'algorithme du backfitting.

Présentation du modèle

Estimation : l'algorithme du backfitting

Lisseurs

Bibliographie

Définition

Soit \hat{m}_λ un estimateur de m qui dépend d'un paramètre λ . \hat{m}_λ est un **lisseur** si il existe une matrice $S_\lambda = S_\lambda(\mathbb{X})$ telle que

$$\hat{\mathbb{Y}} = S_\lambda \mathbb{Y}$$

où $\hat{\mathbb{Y}}$ est le vecteur des valeurs ajustées par \hat{m}_λ et \mathbb{Y} le vecteur des $y_i, i = 1, \dots, n$.

Définition

Soit \hat{m}_λ un estimateur de m qui dépend d'un paramètre λ . \hat{m}_λ est un **lisseur** si il existe une matrice $S_\lambda = S_\lambda(\mathbb{X})$ telle que

$$\hat{\mathbb{Y}} = S_\lambda \mathbb{Y}$$

où $\hat{\mathbb{Y}}$ est le vecteur des valeurs ajustées par \hat{m}_λ et \mathbb{Y} le vecteur des $y_i, i = 1, \dots, n$.

Exemple

- Estimateurs à noyau :

$$S_{ij,h} = \frac{K((x_i - x_j)/h)}{\sum_l K((x_i - x_l)/h)}.$$

Définition

Soit \hat{m}_λ un estimateur de m qui dépend d'un paramètre λ . \hat{m}_λ est un **lisseur** si il existe une matrice $S_\lambda = S_\lambda(\mathbb{X})$ telle que

$$\hat{\mathbb{Y}} = S_\lambda \mathbb{Y}$$

où $\hat{\mathbb{Y}}$ est le vecteur des valeurs ajustées par \hat{m}_λ et \mathbb{Y} le vecteur des $y_i, i = 1, \dots, n$.

Exemple

- Estimateurs à noyau :

$$S_{ij,h} = \frac{K((x_i - x_j)/h)}{\sum_l K((x_i - x_l)/h)}.$$

- Estimateurs des k -ppv :

$$S_{ij,k} = \begin{cases} 1/k & \text{si } x_j \text{ est parmi les } k\text{-ppv de } x_i \\ 0 & \text{sinon.} \end{cases}$$

- L'information sur le **niveau de lissage** est entièrement portée par la **matrice de lissage** S_λ .

- L'information sur le **niveau de lissage** est entièrement portée par la **matrice de lissage** S_λ .

Projecteur

- Lorsque S_λ est un **projecteur** (**modèle linéaire par exemple**), on a
$$\text{tr}(S_\lambda) = \text{rang}(S_\lambda) = \dim \text{esp. où on projette.}$$

- L'information sur le **niveau de lissage** est entièrement portée par la **matrice de lissage** S_λ .

Projecteur

- Lorsque S_λ est un **projecteur** (**modèle linéaire par exemple**), on a

$$\text{tr}(S_\lambda) = \text{rang}(S_\lambda) = \dim \text{esp. où on projette.}$$

- $\text{tr}(S_\lambda)$ est une mesure de la **complexité du lisseur** dans ce cas.

- Par analogie avec la remarque précédente, on pose donc la définition suivante :

Définition

Le nombre de **degrés de liberté** d'un lisseur S_λ est défini par

$$\text{df} = \text{tr}(S_\lambda).$$

- Exemple des **kppv** :

- Par analogie avec la remarque précédente, on pose donc la définition suivante :

Définition

Le nombre de **degrés de liberté** d'un lisseur S_λ est défini par

$$df = \text{tr}(S_\lambda).$$

- Exemple des **kppv** : $df = n/k$.
- **Interprétation** :

- Par analogie avec la remarque précédente, on pose donc la définition suivante :

Définition

Le nombre de **degrés de liberté** d'un lisseur S_λ est défini par

$$df = \text{tr}(S_\lambda).$$

- Exemple des **kppv** : $df = n/k$.
- **Interprétation** :
 1. $df \nearrow \implies$ flexibilité \nearrow , biais \searrow et variance \nearrow ;
 2. $df \searrow \implies$ flexibilité \searrow , biais \nearrow et variance \searrow ;

- Le choix du paramètre de lissage est **crucial** pour les estimateurs non-paramétriques.

- Le choix du paramètre de lissage est **crucial** pour les estimateurs non-paramétriques.
- Une approche : **validation croisée LOO**, on choisit λ qui minimise

$$\text{LOO}(\hat{m}_\lambda) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{m}_\lambda^i(x_i))^2$$

où \hat{m}_λ^i désigne le lisseur calculé sans la i ème observation.

- Le choix du paramètre de lissage est **crucial** pour les estimateurs non-paramétriques.
- Une approche : **validation croisée LOO**, on choisit λ qui minimise

$$\text{LOO}(\hat{m}_\lambda) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{m}_\lambda^i(x_i))^2$$

où \hat{m}_λ^i désigne le lisseur calculé sans la i ème observation.

Remarque

Cette approche peut se révéler **coûteuse en temps de calcul** car il faut calculer n fois l'estimateurs pour chaque valeur de λ .

- Pour la plupart des **lisseurs** non paramétriques, il n'est pas nécessaire de recalculer n fois l'estimateur.

- Pour la plupart des **lisseurs** non paramétriques, il n'est pas nécessaire de recalculer n fois l'estimateur.
- Le critère **LOO** se déduit directement de la **matrice de lissage** S_λ !

- Pour la plupart des **lisseurs** non paramétriques, il n'est pas nécessaire de recalculer n fois l'estimateur.
- Le critère **LOO** se déduit directement de la **matrice de lissage** S_λ !
- Par exemple, pour l'**estimateur à noyau** on a (voir [Cornillon et al., 2019])

$$\text{LOO}(\hat{m}_h) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{m}_h(x_i)}{1 - S_{ii,h}} \right)^2 ,$$

- Pour la plupart des **lisseurs** non paramétriques, il n'est pas nécessaire de recalculer n fois l'estimateur.
- Le critère **LOO** se déduit directement de la **matrice de lissage** S_λ !
- Par exemple, pour l'**estimateur à noyau** on a (voir [Cornillon et al., 2019])

$$\text{LOO}(\hat{m}_h) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{m}_h(x_i)}{1 - S_{ii,h}} \right)^2 ,$$

- et pour l'estimateur des **kppv**

$$\text{LOO}(\hat{m}_k) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{m}_{k+1}(x_i)}{1 - S_{ii,k+1}} \right)^2 .$$

- Pour la plupart des **lisseurs** non paramétriques, il n'est pas nécessaire de recalculer n fois l'estimateur.
- Le critère **LOO** se déduit directement de la **matrice de lissage** S_λ !
- Par exemple, pour l'**estimateur à noyau** on a (voir [Cornillon et al., 2019])

$$\text{LOO}(\hat{m}_h) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{m}_h(x_i)}{1 - S_{ii,h}} \right)^2,$$

- et pour l'estimateur des **kppv**

$$\text{LOO}(\hat{m}_k) = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{m}_{k+1}(x_i)}{1 - S_{ii,k+1}} \right)^2.$$

Remarque

Beaucoup plus **efficace** d'un point de vue **computationnel**.

- Il existe une variante, appelée **Validation Croisée Généralisée (GCV)** au critère LOO, qui peut se révéler plus stable (voir [[Woods, 2006](#)]).
- Elle consiste à remplacer au dénominateur $S_{ii,\lambda}$ par $\text{tr}(S_\lambda)/n$.

- Il existe une variante, appelée **Validation Croisée Généralisée (GCV)** au critère LOO, qui peut se révéler plus stable (voir [[Woods, 2006](#)]).
- Elle consiste à remplacer au dénominateur $S_{ii,\lambda}$ par $\text{tr}(S_\lambda)/n$.

Définition

Le critère de **Validation Croisée Généralisée (GCV)** est défini par

$$\text{GCV}(\hat{m}_\lambda) = \frac{n \sum_{i=1}^n (y_i - \hat{m}_\lambda(x_i))^2}{(n - \text{tr}(S_\lambda))^2}.$$

- Il existe une variante, appelée **Validation Croisée Généralisée (GCV)** au critère LOO, qui peut se révéler plus stable (voir [[Woods, 2006](#)]).
- Elle consiste à remplacer au dénominateur $S_{ii,\lambda}$ par $\text{tr}(S_\lambda)/n$.

Définition

Le critère de **Validation Croisée Généralisée (GCV)** est défini par

$$\text{GCV}(\hat{m}_\lambda) = \frac{n \sum_{i=1}^n (y_i - \hat{m}_\lambda(x_i))^2}{(n - \text{tr}(S_\lambda))^2}.$$

- On remarque, une fois de plus, que ce critère réalise un compromis entre **ajustement** (numérateur) et **complexité** (dénominateur).

- Les composantes du modèle additif ne nécessitent pas toutes le même degré de lissage.
- On utilise un lissage pour chaque composante dans l'algorithme du backfitting.

- Les composantes du modèle additif ne nécessitent pas toutes le même degré de lissage.
- On utilise un lissage pour chaque composante dans l'algorithme du backfitting.
- Il convient donc de sélectionner d paramètres de lissage $\lambda = (\lambda_1, \dots, \lambda_d)$.

- Les composantes du modèle additif ne nécessitent pas toutes le même degré de lissage.
- On utilise un lissage pour chaque composante dans l'algorithme du backfitting.
- Il convient donc de sélectionner d paramètres de lissage $\lambda = (\lambda_1, \dots, \lambda_d)$.
- Il est possible de montrer (voir [Woods, 2006]) que l'estimateur final du modèle additif est également un lisseur dont le degrés de liberté dépend des paramètres de lissage $\lambda_1, \dots, \lambda_d$.

- Les composantes du modèle additif ne nécessitent pas toutes le même degré de lissage.
- On utilise un lissage pour chaque composante dans l'algorithme du backfitting.
- Il convient donc de sélectionner d paramètres de lissage $\lambda = (\lambda_1, \dots, \lambda_d)$.
- Il est possible de montrer (voir [Woods, 2006]) que l'estimateur final du modèle additif est également un lisseur dont le degrés de liberté dépend des paramètres de lissage $\lambda_1, \dots, \lambda_d$.
- C'est pourquoi GCV est fréquemment utilisé pour sélectionner les paramètres de lissage dans les modèles additifs.
- Sur R, on pourra utiliser le package mgcv.

```
> mod.mgcv <- mgcv::gam(Y~s(X1)+s(X2),data=donnees)
> mod.mgcv
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Y ~ s(X1) + s(X2)
##
## Estimated degrees of freedom:
## 1.00 8.56 total = 10.56
##
## GCV score: 1.536021
```

Le coin R

```
> mod.mgcv <- mgcv::gam(Y~s(X1)+s(X2),data=donnees)
> mod.mgcv
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Y ~ s(X1) + s(X2)
##
## Estimated degrees of freedom:
## 1.00 8.56 total = 10.56
##
## GCV score: 1.536021
```

Remarque

Sans surprise, le **degré de liberté** sélectionné pour la deuxième composante est plus **grand** que celui de la première.

Régression logistique additive

- Nous nous sommes restreints à un modèle de régression (Y continue), qui est un modèle GLM particulier.

Régression logistique additive

- Nous nous sommes restreints à un **modèle de régression** (Y continue), qui est un modèle **GLM** particulier.
- Bien entendu, ce type de modélisation s'**étend à l'ensemble des GLM**, d'où le nom de modèle **GAM** (Generalized Additive Model).

Régression logistique additive

- Nous nous sommes restreints à un **modèle de régression** (Y continue), qui est un modèle **GLM** particulier.
- Bien entendu, ce type de modélisation s'étend à l'ensemble des GLM, d'où le nom de modèle **GAM** (Generalized Additive Model).

Régression logistique additive

Pour une variable Y binaire, le modèle de **régression logistique additive** s'écrit :

$$\text{logit } p(x) = \log \frac{\mathbf{P}(Y = 1|X = x)}{1 - \mathbf{P}(Y = 1|X = x)} = \alpha_0 + g_1(x_1) + \dots + g_d(x_d).$$

Régression logistique additive

- Nous nous sommes restreints à un **modèle de régression** (Y continue), qui est un modèle **GLM** particulier.
- Bien entendu, ce type de modélisation s'étend à l'ensemble des GLM, d'où le nom de modèle **GAM** (Generalized Additive Model).

Régression logistique additive

Pour une variable Y binaire, le modèle de **regression logistique additive** s'écrit :

$$\text{logit } p(x) = \log \frac{\mathbf{P}(Y = 1|X = x)}{1 - \mathbf{P}(Y = 1|X = x)} = \alpha_0 + g_1(x_1) + \dots + g_d(x_d).$$

- La **procédure d'estimation** combine l'algorithme du **backfitting** avec l'algorithme du **score de Fisher** (voir [Hastie et al., 2009]).

Présentation du modèle

Estimation : l'algorithme du backfitting

Lisseurs

Bibliographie



Cornillon, P., Hengartner, N., Matzner-Løber, E., and Rouvière, L. (2019).

Régression avec R.

EDP Sciences, 2 edition.



Hastie, T., Tibshirani, R., and Friedman, J. (2009).

The Elements of Statistical Learning : Data Mining, Inference, and Prediction.


Springer, second edition.



Opsomer, J. (2000).

Asymptotic properties of backfitting estimators.

Journal of Multivariate Analysis, 73 :166–179.

-  Woods, S. (2006).
Generalized additive models.
Chapman & Hall.

Cinquième partie V

Clustering spectral

Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie

Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie

Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie

Classification non supervisée

Definition (Classification)

Action de **répartir en classes**, en catégories, des choses, des objets, ayant des caractères communs afin notamment d'en faciliter l'étude.

Exemples

- Astronomie : classification d'étoiles
- Médecine : diagnostic de maladies à partir d'observation cliniques
- Géographie : délimitation de zones homogènes
- Marketing : détermination de segments de marchés (groupes de consommateurs ayant les mêmes habitudes)
- Réseaux sociaux : extraction de communautés
- ...

- Beaucoup de groupes avec peu d'individus à l'intérieur (réduire n)

- Beaucoup de groupes avec peu d'individus à l'intérieur (réduire n)
- Peu de groupes avec beaucoup d'individus à l'intérieur (extraire des profils que l'on interprète par la suite).

- Beaucoup de groupes avec peu d'individus à l'intérieur (réduire n)
- Peu de groupes avec beaucoup d'individus à l'intérieur (extraire des profils que l'on interprète par la suite).

Remarque

Les algorithmes seront proches mais pas calibrés de la même façon.

- n observations X_1, \dots, X_n à valeurs dans \mathbb{R}^d .

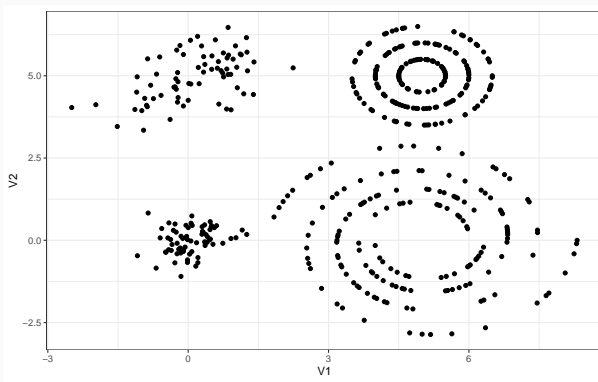
Le problème

Trouver une **partition** de $\{X_1, \dots, X_n\}$: on cherche donc $\mathcal{C}_1, \dots, \mathcal{C}_K$ tels que

$$\bigcup_{k=1}^K \mathcal{C}_k = \{X_1, \dots, X_n\} \quad \text{et} \quad \mathcal{C}_k \cap \mathcal{C}_{k'} = \emptyset \quad \text{si} \quad k \neq k'.$$

Chaque élément de la partition \mathcal{C}_k est appelé **cluster**.

Un exemple jouet



Différents problèmes

- Choix du nombre de groupes K .
- Choix de distances ou similarités entre objets mais aussi entre groupes.

Différents problèmes

- Choix du nombre de groupes K .
- Choix de distances ou similarités entre objets mais aussi entre groupes.
- Pour K fixé, on veut minimiser

$$\mathcal{I}_{\text{intra}} = \frac{1}{n} \sum_{k=1}^K \sum_{i \in \mathcal{C}_k} d^2(x_i, \bar{x}_{\mathcal{C}_k}).$$

Impossible de minimiser sur toutes les partitions. Par exemple, pour $n = 20$ et $K = 4$ on a $5.17e^{13}$ partitions possibles.

Différents problèmes

- Choix du nombre de groupes K .
- Choix de distances ou similarités entre objets mais aussi entre groupes.
- Pour K fixé, on veut minimiser

$$\mathcal{I}_{\text{intra}} = \frac{1}{n} \sum_{k=1}^K \sum_{i \in \mathcal{C}_k} d^2(x_i, \bar{x}_{\mathcal{C}_k}).$$

Impossible de minimiser sur toutes les partitions. Par exemple, pour $n = 20$ et $K = 4$ on a $5.17e^{13}$ partitions possibles.

Remarque importante

Souvent beaucoup plus difficile qu'un problème supervisé !

Différentes familles d'algorithmes

- Algorithmes par **partitionnement** : recherche de points qui optimisent un critère donné

Différentes familles d'algorithmes

- Algorithmes par **partitionnement** : recherche de points qui optimisent un critère donné \implies *k-means*, *k-médoids*...

Différentes familles d'algorithmes

- Algorithmes par **partitionnement** : recherche de points qui optimisent un critère donné \implies **k-means**, **k-médoids**...
- Algorithmes d'**agglomération** (ou division) : cluster construit de façon séquentielle en agglomérant les points les plus proches les uns des autres

Différentes familles d'algorithmes

- Algorithmes par **partitionnement** : recherche de points qui optimisent un critère donné \implies **k-means, k-médoids...**
- Algorithmes d'**agglomération** (ou division) : cluster construit de façon séquentielle en agglomérant les points les plus proches les uns des autres \implies **Classification Ascendante Hiérarchique.**

Différentes familles d'algorithmes

- Algorithmes par **partitionnement** : recherche de points qui optimisent un critère donné \implies **k-means, k-médoids...**
- Algorithmes d'**agglomération** (ou division) : cluster construit de façon séquentielle en agglomérant les points les plus proches les uns des autres \implies **Classification Ascendante Hiérarchique.**
- Algorithmes par **densité** (géométrie) : les clusters correspondent à des zones de forte densité séparées par des zones de faible densité

Différentes familles d'algorithmes

- Algorithmes par **partitionnement** : recherche de points qui optimisent un critère donné \implies **k-means, k-médoids...**
- Algorithmes d'**agglomération** (ou division) : cluster construit de façon séquentielle en agglomérant les points les plus proches les uns des autres \implies **Classification Ascendante Hiérarchique**.
- Algorithmes par **densité** (géométrie) : les clusters correspondent à des zones de forte densité séparées par des zones de faible densité \implies **modèle de mélange, DBSCAN**.

Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie

Les k -means

- Données : x_1, \dots, x_n à valeurs dans \mathbb{R}^d ;
- Nombre de groupes K **fixé** ;
- Soit $\mathcal{C} = (\mathcal{C}_1, \dots, \mathcal{C}_K)$ une partition de $\{1, \dots, n\}$.
- Soit $c = (c_1, \dots, c_K)$ K représentants de chaque classe, $c_k \in \mathbb{R}^d$.

Les k -means

- Données : x_1, \dots, x_n à valeurs dans \mathbb{R}^d ;
- Nombre de groupes K **fixé** ;
- Soit $\mathcal{C} = (\mathcal{C}_1, \dots, \mathcal{C}_K)$ une partition de $\{1, \dots, n\}$.
- Soit $c = (c_1, \dots, c_K)$ K représentants de chaque classe, $c_k \in \mathbb{R}^d$.

Le critère des k -means

On cherche la partition \mathcal{C} et les représentants c qui **minimise le critère**

$$g(\mathcal{C}, c) = \sum_{k=1}^K \sum_{i \in \mathcal{C}_k} \|x_i - c_k\|^2.$$

Les k -means

- Données : x_1, \dots, x_n à valeurs dans \mathbb{R}^d ;
- Nombre de groupes K **fixé** ;
- Soit $\mathcal{C} = (\mathcal{C}_1, \dots, \mathcal{C}_K)$ une partition de $\{1, \dots, n\}$.
- Soit $c = (c_1, \dots, c_K)$ K représentants de chaque classe, $c_k \in \mathbb{R}^d$.

Le critère des k -means

On cherche la partition \mathcal{C} et les représentants c qui **minimise le critère**

$$g(\mathcal{C}, c) = \sum_{k=1}^K \sum_{i \in \mathcal{C}_k} \|x_i - c_k\|^2.$$

- A l'heure actuelle, on ne sait **pas trouver la solution** de ce problème.

Les k -means

- Données : x_1, \dots, x_n à valeurs dans \mathbb{R}^d ;
- Nombre de groupes K **fixé** ;
- Soit $\mathcal{C} = (\mathcal{C}_1, \dots, \mathcal{C}_K)$ une partition de $\{1, \dots, n\}$.
- Soit $c = (c_1, \dots, c_K)$ K représentants de chaque classe, $c_k \in \mathbb{R}^d$.

Le critère des k -means

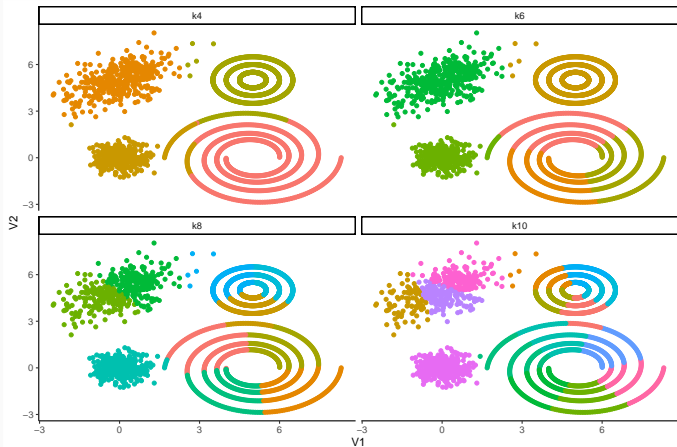
On cherche la partition \mathcal{C} et les représentants c qui **minimise le critère**

$$g(\mathcal{C}, c) = \sum_{k=1}^K \sum_{i \in \mathcal{C}_k} \|x_i - c_k\|^2.$$

- A l'heure actuelle, on ne sait **pas trouver la solution** de ce problème.
- Approche **récursive** pour trouver des **extrema locaux** : Lloyd ou Forgi, Mac Queen, Hartigan...

Le coin R

```
> km1 <- kmeans(don1,centers=4,nstart=100)
> km2 <- kmeans(don1,centers=6,nstart=100)
> km3 <- kmeans(don1,centers=10,nstart=100)
> km4 <- kmeans(don1,centers=15,nstart=100)
```



Objectif

Créer une **suite de partitions emboîtées** en partant de la partition la plus **fine** (n classes) jusqu'à obtenir **une seule classe**.

Objectif

Créer une **suite de partitions emboîtées** en partant de la partition la plus **fine** (n classes) jusqu'à obtenir **une seule classe**.

1. Calculer un tableau de **distances entre objets**, 1 partition n singletons.
2. **Agréger** les deux objets les plus proches.
3. Mettre à jour **la matrice de distances**.
4. Itérer jusqu'à avoir **un seul groupe**.

- Choix de la **distance** ou de la **similarité** entre objets ;

Questions

- Choix de la **distance** ou de la **similarité** entre objets ;
- Choix de la méthode d'agrégation \implies single linkage, complete linkage, Ward...

Questions

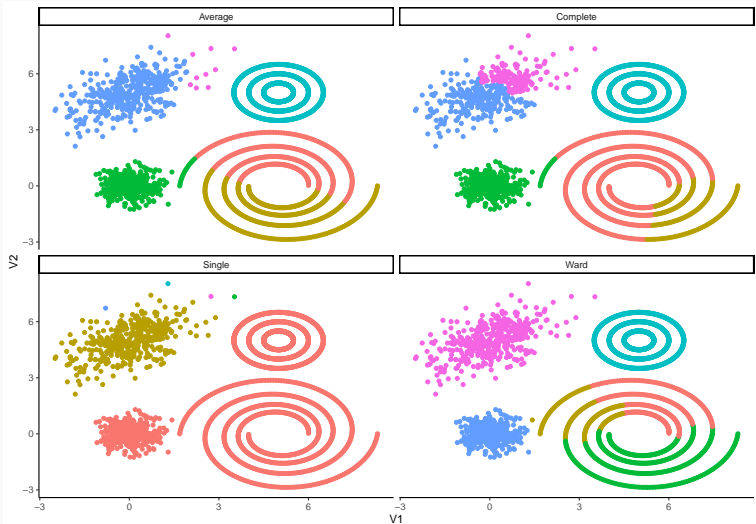
- Choix de la **distance** ou de la **similarité** entre objets ;
- Choix de la méthode d'agrégation \implies single linkage, complete linkage, Ward...
- Choix du nombre de classes...

Questions

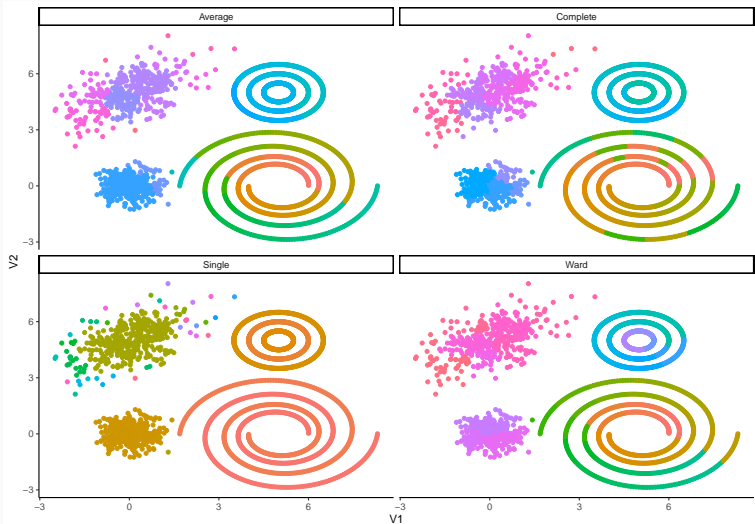
- Choix de la **distance** ou de la **similarité** entre objets ;
- Choix de la méthode d'agrégation \implies single linkage, complete linkage, Ward...
- Choix du nombre de classes...
- Sur **R**, on peut utiliser **hclust** :

```
> Dist <- dist(don2)
> hc1 <- hclust(Dist,method="ward.D2");hc11 <- cutree(hc1,k=6)
> hc2 <- hclust(Dist,method="single");hc22 <- cutree(hc2,k=6)
> hc3 <- hclust(Dist,method="complete");hc33 <- cutree(hc3,k=6)
> hc4 <- hclust(Dist,method="average");hc44 <- cutree(hc4,k=6)
```

CAH avec 6 groupes



CAH avec 40 groupes



- Complexité : $O(n)$ pour le k -means, $O(n^3)$ pour la CAH.

Un bilan

- Complexité : $O(n)$ pour le k -means, $O(n^3)$ pour la CAH.
- Structure sphérique pour le k -means.

Un bilan

- Complexité : $O(n)$ pour le k -means, $O(n^3)$ pour la CAH.
- Structure sphérique pour le k -means.
- Structure hiérarchique pour la CAH.

Un bilan

- Complexité : $O(n)$ pour le k -means, $O(n^3)$ pour la CAH.
- Structure sphérique pour le k -means.
- Structure hiérarchique pour la CAH.

CAH avec n grand

- faire un k -means avec beaucoup de classes (par exemple 1000) ;
- faire la CAH sur les centres des classes calculés à l'étape précédente.

Un bilan

- **Complexité** : $O(n)$ pour le k -means, $O(n^3)$ pour la CAH.
- Structure **sphérique** pour le k -means.
- Structure **hiérarchique** pour la CAH.

CAH avec n grand

- faire un k -means avec beaucoup de classes (par exemple 1000) ;
- faire la CAH sur les centres des classes calculés à l'étape précédente.
- Sur **R** on pourra utiliser la fonction **HCPC** du package **FactoMineR**.

Un bilan

- Complexité : $O(n)$ pour le k -means, $O(n^3)$ pour la CAH.
- Structure sphérique pour le k -means.
- Structure hiérarchique pour la CAH.

CAH avec n grand

- faire un k -means avec beaucoup de classes (par exemple 1000) ;
- faire la CAH sur les centres des classes calculés à l'étape précédente.
- Sur R on pourra utiliser la fonction HCPC du package FactoMineR.

Le clustering spectral

- Approche basée sur la décomposition spectrale du Laplacien d'un graphe.
- Peut être utilisée pour détecter des communautés sur des graphes mais aussi pour faire de clustering sur des données "classiques".

Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie

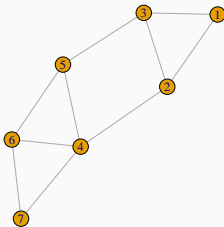
- Le **graph mining** ou **fouille de graphes** correspond à la fouille de données spécifiques aux graphes.

- Le **graph mining** ou **fouille de graphes** correspond à la fouille de données spécifiques aux graphes.

Graphe

Objet mathématique utilisé pour modéliser des **connexions** ou **interactions** entre **individus** ou **entités** :

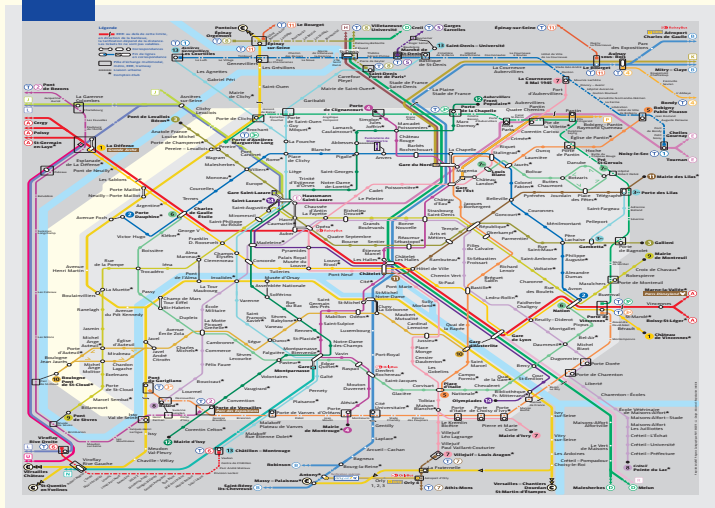
- les entités sont appelées **nœuds** ou **sommets** ;
- une relation entre deux entités est modélisée par une **arête**.



Nombreuses applications

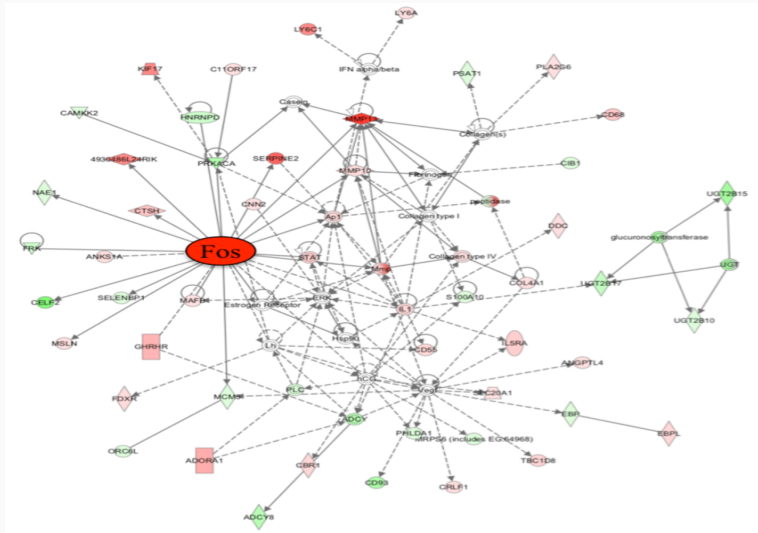
- Réseaux routiers entre villes, réseaux aériens entre aéroports...
- Réseaux électriques (cables reliant des prises)
- Internet (routeurs et ordinateurs connectés par ethernet ou wifi)
- Réseaux d'amis Facebook
- Communication : personnes avec qui on communique (téléphone par exemple)
- World wide web (les nœuds sont les pages internet et les arêtes sont les hyperliens)
- Réseaux de régulation entre gènes
- Systèmes de recommandation...

Métro parisien



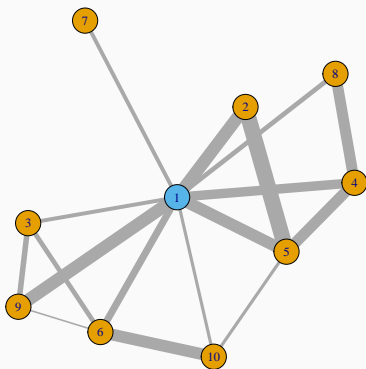


Réseaux moléculaires



Communications

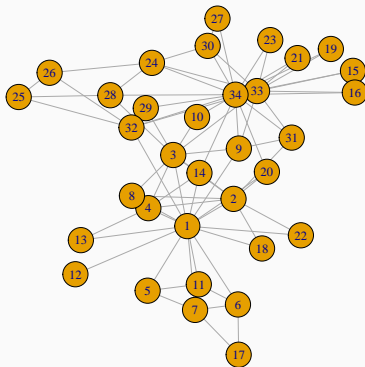
- **Objectif** : visualiser les communications d'un individu sur une période donnée.



- **Nœuds** : membres d'un club de karaté universitaire ;
- **Arêtes** : lien d'amitié calculé en fonction du nombre d'activités communes.

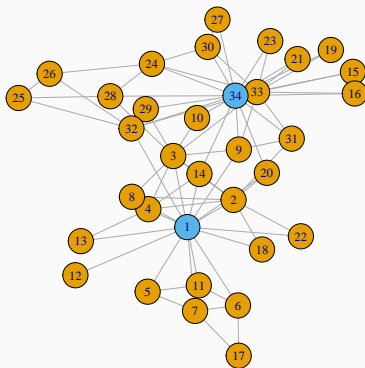
Karate

- **Nœuds** : membres d'un club de karaté universitaire ;
- **Arêtes** : lien d'amitié calculé en fonction du nombre d'activités communes.



Karate

- **Nœuds** : membres d'un club de karaté universitaire ;
- **Arêtes** : lien d'amitié calculé en fonction du nombre d'activités communes.



Plusieurs problématiques

Analyse exploratoire

Comprendre la **structure d'interaction** entre entités en analysant la topologie du graphe.

1. **Construction** d'un graphe : définition des **nœuds** et des **arêtes**.
2. **Visualisation** : comment représenter et dessiner un graphe ?
3. **Détection de communautés** : identifier des sous-groupes de nœuds très connectés.

Plusieurs problématiques

Analyse exploratoire

Comprendre la **structure d'interaction** entre entités en analysant la topologie du graphe.

1. **Construction** d'un graphe : définition des **nœuds** et des **arêtes**.
2. **Visualisation** : comment représenter et dessiner un graphe ?
3. **Détection de communautés** : identifier des sous-groupes de nœuds très connectés.

Inférence sur les graphes

1. Modèles de **graphes aléatoires**.
2. **Prédiction de connexions** pour des **nouveaux nœuds**.

Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie

Dans cette partie

- On ne présente que les caractéristiques des graphes utiles pour présenter le clustering spectral.
- Pour plus de précisions, on pourra consulter :

<https://lrouviere.github.io/INP-HB/>

Dans cette partie

- On ne présente que les **caractéristiques des graphes** utiles pour présenter le **clustering spectral**.
- Pour plus de **précisions**, on pourra consulter :

<https://lrouviere.github.io/INP-HB/>

Graphe

Un **graphe** $G = (V, E)$ est composé :

- d'un ensemble V de **noeuds ou sommets** (vertices) qui représentent les individus ou entités qui interagissent entre eux,

Dans cette partie

- On ne présente que les **caractéristiques des graphes** utiles pour présenter le **clustering spectral**.
- Pour plus de **précisions**, on pourra consulter :

<https://lrouviere.github.io/INP-HB/>

Graphe

Un **graphe** $G = (V, E)$ est composé :

- d'un ensemble V de **noeuds ou sommets** (vertices) qui représentent les individus ou entités qui interagissent entre eux,
- et d'un ensemble E d'**arêtes** (edges) qui indiquent la présence d'une **interaction** ou **connexion entre deux noeuds** :

$\{i, j\} \in E$ si il y a une arête entre i et j dans G .

Définitions

- Le nombre de nœuds $|V|$ est l'**ordre** du graphe. Le nombre d'arêtes $|E|$ est la **taille** du graphe.
- Un graphe est **dirigé** (ou **orienté**) lorsque ses arêtes le sont. Il est **non dirigé** sinon.

Définitions

- Le nombre de nœuds $|V|$ est l'**ordre** du graphe. Le nombre d'arêtes $|E|$ est la **taille** du graphe.
- Un graphe est **dirigé** (ou **orienté**) lorsque ses arêtes le sont. Il est **non dirigé** sinon.
- Les graphes peuvent être **binaires** (arête présente ou absente), ou **valués** (arêtes munies d'un poids positif).

Définitions

- Le nombre de nœuds $|V|$ est l'**ordre** du graphe. Le nombre d'arêtes $|E|$ est la **taille** du graphe.
- Un graphe est **dirigé** (ou **orienté**) lorsque ses arêtes le sont. Il est **non dirigé** sinon.
- Les graphes peuvent être **binaires** (arête présente ou absente), ou **valués** (arêtes munies d'un poids positif).
- Un graphe est dit **simple** s'il n'y a pas de boucles : (i, i) n'est jamais une arête.

Définitions

- Le nombre de nœuds $|V|$ est l'**ordre** du graphe. Le nombre d'arêtes $|E|$ est la **taille** du graphe.
- Un graphe est **dirigé** (ou **orienté**) lorsque ses arêtes le sont. Il est **non dirigé** sinon.
- Les graphes peuvent être **binaires** (arête présente ou absente), ou **valués** (arêtes munies d'un poids positif).
- Un graphe est dit **simple** s'il n'y a pas de boucles : (i, i) n'est jamais une arête.
- Le graphe **complet** ou **clique** est le graphe (non dirigé) qui contient toutes les arêtes possibles entre les sommets ($C_{|V|}^2$ arêtes).

Définitions

- Le nombre de nœuds $|V|$ est l'**ordre** du graphe. Le nombre d'arêtes $|E|$ est la **taille** du graphe.
- Un graphe est **dirigé** (ou **orienté**) lorsque ses arêtes le sont. Il est **non dirigé** sinon.
- Les graphes peuvent être **binaires** (arête présente ou absente), ou **valués** (arêtes munies d'un poids positif).
- Un graphe est dit **simple** s'il n'y a pas de boucles : (i, i) n'est jamais une arête.
- Le graphe **complet** ou **clique** est le graphe (non dirigé) qui contient toutes les arêtes possibles entre les sommets ($C_{|V|}^2$ arêtes).

Question ?

Comment **stocker** un graphe ?

Matrice d'adjacence

Définition

La **matrice d'adjacence** d'un graphe $G = (V, E)$ binaire est la matrice $|V| \times |V|$ de terme général

$$A_{ij} = \begin{cases} 1 & \text{si } \{i, j\} \in E \\ 0 & \text{sinon.} \end{cases}$$

Matrice d'adjacence

Définition

La **matrice d'adjacence** d'un graphe $G = (V, E)$ binaire est la matrice $|V| \times |V|$ de terme général

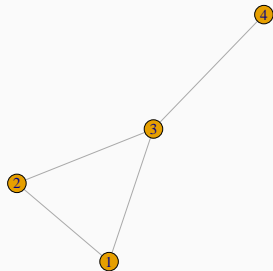
$$A_{ij} = \begin{cases} 1 & \text{si } \{i, j\} \in E \\ 0 & \text{sinon.} \end{cases}$$

Remarques

- Graphe **non dirigé** $\implies A$ symétrique.
- Graphe **simple** $\implies A_{ii} = 0 \forall i$.
- Graphe **valué** $\implies A_{ij} = w_{ij} \in \mathbb{R}^+$.

Exemple : graphe non dirigé

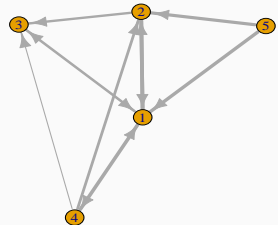
```
> A
##      [,1] [,2] [,3] [,4]
## [1,]    0    1    1    0
## [2,]    1    0    1    0
## [3,]    1    1    0    1
## [4,]    0    0    1    0
> set.seed(1234)
> G <- graph_from_adjacency_matrix(A,
+                                     mode='undirected')
> plot(G)
```



Exemple : graphe dirigé

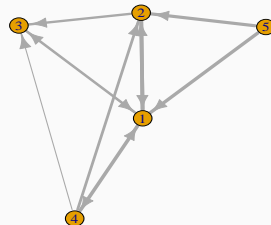
```
> A
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    4    3    2    0
## [2,]    4    0    3    0    0
## [3,]    1    0    0    0    0
## [4,]    4    3    1    0    0
## [5,]    4    4    0    0    0

> G <- graph_from_adjacency_matrix(A,
+   mode='directed',weighted = TRUE)
> E(G)$width <- E(G)$weight
> plot(G)
```



Exemple : graphe dirigé

```
> A
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    4    3    2    0
## [2,]    4    0    3    0    0
## [3,]    1    0    0    0    0
## [4,]    4    3    1    0    0
## [5,]    4    4    0    0    0
> G <- graph_from_adjacency_matrix(A,
+   mode='directed',weighted = TRUE)
> E(G)$width <- E(G)$weight
> plot(G)
```



Remarque

- Pas toujours efficace en terme de stockage : $O(|V|^2)$.
- Utiliser des matrices sparses si le graphe est très creux.

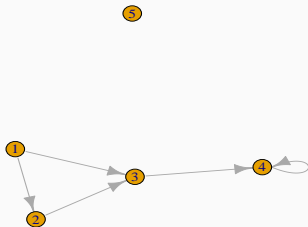
Liste d'arêtes

- Il est souvent plus efficace de définir le graphe en donnant une liste d'arêtes.
- **Attention** : penser à donner le **nombre total de nœuds du graphe** pour éviter d'oublier les **nœuds isolés**.

Liste d'arêtes

- Il est souvent plus efficace de définir le graphe en donnant une liste d'arêtes.
- **Attention** : penser à donner le **nombre total de nœuds du graphe** pour éviter d'oublier les **nœuds isolés**.

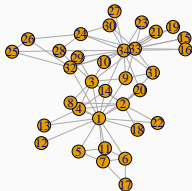
```
> G <- graph(edges=c(1,2,1,3,3,4,4,4,2,3),n=5)  
> plot(G)
```



Visualisation d'un graphe

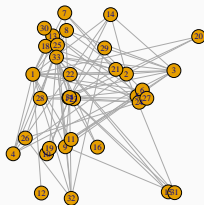
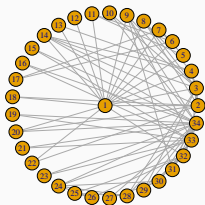
- **Etape importante** : elle permet de comprendre la **structure du graphe** :
identification de nœuds importants, très connectés...
- **Différentes représentations** :
 1. en cercle, en étoile...
 2. selon différents algorithmes (voir [[Bahoken et al., 2013](#)])mais **attention** : certaines peuvent être **trompeuses** :

```
> plot(kar)
```



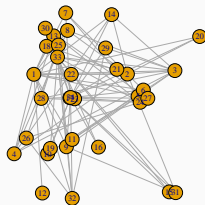
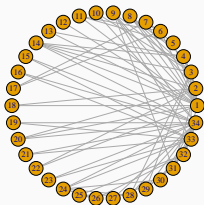
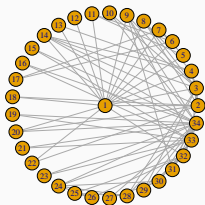
Autre représentations

```
> plot(kar,layout=layout_as_star(kar))  
> plot(kar,layout=layout_circle(kar))  
> plot(kar,layout=layout_randomly(kar))
```



Autre représentations

```
> plot(kar,layout=layout_as_star(kar))  
> plot(kar,layout=layout.circle(kar))  
> plot(kar,layout=layout_randomly(kar))
```



Packages

- **R** : igraph, visNetwork, GGally.
- **Python** : NetworkX.

Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie

Caractéristiques d'un graphe

- De nombreux indicateurs permettent de décrire un graphe.

Objectifs

Caractéristiques d'un graphe

- De nombreux indicateurs permettent de décrire un graphe.

Objectifs

- Donner une version résumée du graphe.

Caractéristiques d'un graphe

- De nombreux indicateurs permettent de décrire un graphe.

Objectifs

- Donner une version résumée du graphe.
- Décrire et comprendre les interactions entre entités :
 - transfert d'information entre deux sommets.
 - importance de certains sommets.
 - sous-structure particulière dans le graphe.

Caractéristiques d'un graphe

- De nombreux indicateurs permettent de décrire un graphe.

Objectifs

- Donner une version résumée du graphe.
- Décrire et comprendre les interactions entre entités :
 - transfert d'information entre deux sommets.
 - importance de certains sommets.
 - sous-structure particulière dans le graphe.
- Comparer deux graphes.
- Comparer un graphe avec un modèle de graphe aléatoire.

Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie

Distance - diamètre

- Un **chemin** entre $i \in V$ et $j \in V$ est une suite de nœuds et d'arêtes permettant de relier i et j .
- **Longueur d'un chemin** entre i et j : nombre d'arêtes qui composent ce chemin.
- **Distance ℓ_{ij} entre i et j** : longueur du plus court chemin qui les relie. Si les deux nœuds ne sont pas connectés : $\ell_{ij} = +\infty$.

Distance - diamètre

- Un **chemin** entre $i \in V$ et $j \in V$ est une suite de nœuds et d'arêtes permettant de relier i et j .
- **Longueur d'un chemin** entre i et j : nombre d'arêtes qui composent ce chemin.
- **Distance ℓ_{ij} entre i et j** : longueur du plus court chemin qui les relie. Si les deux nœuds ne sont pas connectés : $\ell_{ij} = +\infty$.
- **Diamètre d'un graphe** : plus grande distance entre deux nœuds (quantité définie uniquement pour les **graphes connexes**).

Distance - diamètre

- Un **chemin** entre $i \in V$ et $j \in V$ est une suite de nœuds et d'arêtes permettant de relier i et j .
- **Longueur d'un chemin** entre i et j : nombre d'arêtes qui composent ce chemin.
- **Distance ℓ_{ij} entre i et j** : longueur du plus court chemin qui les relie. Si les deux nœuds ne sont pas connectés : $\ell_{ij} = +\infty$.
- **Diamètre d'un graphe** : plus grande distance entre deux nœuds (quantité définie uniquement pour les **graphes connexes**).

Transfert d'information

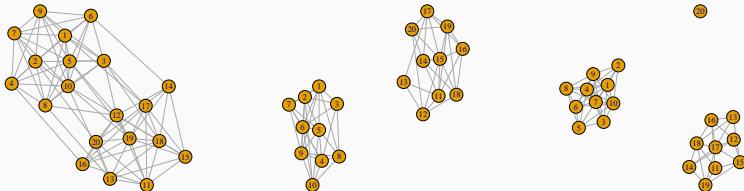
Un **petit diamètre** indique que l'**information circule rapidement** dans le graphe entier.

- Une **composante connexe** est un sous-ensemble $C = \{v_1, \dots, v_k\} \subset V$ tel que, pour tout $v_i, v_j \in C$, il existe un chemin dans G de v_i à v_j .

- Une **composante connexe** est un sous-ensemble $C = \{v_1, \dots, v_k\} \subset V$ tel que, pour tout $v_i, v_j \in C$, il existe un chemin dans G de v_i à v_j .
- Un nœud qui n'est connecté à aucun autre est dit **isolé** \implies il forme une composante connexe à lui tout seul.

- Une **composante connexe** est un sous-ensemble $C = \{v_1, \dots, v_k\} \subset V$ tel que, pour tout $v_i, v_j \in C$, il existe un chemin dans G de v_i à v_j .
- Un nœud qui n'est connecté à aucun autre est dit **isolé** \implies il forme une composante connexe à lui tout seul.
- Un graphe est dit **connexe** s'il possède une **unique composante connexe**.

Exemple



Commentaires

- **Gauche** : graphe connexe.
- **Centre** : 2 composantes connexes.
- **Droite** : 3 composantes connexes, 1 nœud isolé.

1. Plusieurs composantes connexes dans le graphe ?
 - présence de nœuds isolés (participant inactif) ?
 - 1 composante connexe = 1 groupe d'individus (ou 1 cluster) ?

Questions liées à la connexité

1. Plusieurs composantes connexes dans le graphe ?
 - présence de nœuds isolés (participant inactif) ?
 - 1 composante connexe = 1 groupe d'individus (ou 1 cluster) ?
2. Si non, est-ce "presque" le cas ? \implies Création de nouvelles composantes connexes en supprimant quelques nœuds ou arêtes.

Questions liées à la connexité

1. Plusieurs composantes connexes dans le graphe ?
 - présence de nœuds isolés (participant inactif) ?
 - 1 composante connexe = 1 groupe d'individus (ou 1 cluster) ?
2. Si non, est-ce "presque" le cas ? \implies Création de nouvelles composantes connexes en supprimant quelques nœuds ou arêtes.
3. Recherche de communautés dans les graphes connexes : groupes de sommets très connectés entre eux et peu connectés avec les autres groupes.

La plupart des indicateurs s'obtiennent directement avec R

- Nombre de nœuds, d'arêtes, composantes connexes, diamètre, densité :

```
> vcount(kar)
## [1] 34
> ecoun(kar)
## [1] 78
> count_components(kar)
## [1] 1
> diameter(kar)
## [1] 5
> edge_density(kar)
## [1] 0.1390374
```

- Nombre de **triangles** :

```
> head(count_triangles(kar))  
## [1] 18 12 11 10 2 3  
> length(triangles(kar))/3  
## [1] 45
```

- Nombre de triangles :

```
> head(count_triangles(kar))  
## [1] 18 12 11 10 2 3  
> length(triangles(kar))/3  
## [1] 45
```

- Nombre de cliques de taille 3 à 5 :

```
> count_max_cliques(kar,min=3,max=5)  
## [1] 25
```

Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie

Objectif

- Identifier les **nœuds centraux** d'un réseau.
- Rechercher les **nœuds influents**, clés d'un réseau.

Objectif

- Identifier les **nœuds centraux** d'un réseau.
- Rechercher les **nœuds influents**, clés d'un réseau.

Comment ?

En définissant des **indicateurs de centralité** pour les nœuds d'un graphe.

Voisins, degré

- Les **voisins** de $i \in V$ sont les nœuds $j \in V$ tels que $i, j \in E$. On note

$$\mathcal{V}(i) = \{j \in V : \{i, j\} \in E\}.$$

Voisins, degré

- Les **voisins** de $i \in V$ sont les nœuds $j \in V$ tels que $i, j \in E$. On note

$$\mathcal{V}(i) = \{j \in V : \{i, j\} \in E\}.$$

Degré

- Le **degré** d_i d'un nœud i est le nombre de voisins de i : $d = |\mathcal{V}(i)|$.
- Calcul** à partir de la matrice d'adjacence A :
 - Graphe non dirigé** : $d_i = \sum_{j, j \neq i} A_{ij}$.
 - Graphe dirigé** : degrés **sortant** et **entrant**

$$d_i^{\text{out}} = \sum_{j, j \neq i} A_{ij} \quad \text{et} \quad d_i^{\text{in}} = \sum_{j, j \neq i} A_{ji}.$$

Voisins, degré

- Les **voisins** de $i \in V$ sont les nœuds $j \in V$ tels que $i, j \in E$. On note

$$\mathcal{V}(i) = \{j \in V : \{i, j\} \in E\}.$$

Degré

- Le **degré** d_i d'un nœud i est le nombre de voisins de i : $d = |\mathcal{V}(i)|$.
- Calcul** à partir de la matrice d'adjacence A :
 - Graphe non dirigé** : $d_i = \sum_{j, j \neq i} A_{ij}$.
 - Graphe dirigé** : degrés **sortant** et **entrant**

$$d_i^{\text{out}} = \sum_{j, j \neq i} A_{ij} \quad \text{et} \quad d_i^{\text{in}} = \sum_{j, j \neq i} A_{ji}.$$

Remarque

Notion la plus simple mais qui ne prend **pas nécessairement en compte la structure du graphe**.

Il existe d'autres indicateurs qui permettent de mesurer l'importance des nœuds, Notamment :

Il existe d'autres indicateurs qui permettent de mesurer l'importance des nœuds, Notamment :

- Degré de centralité de proximité : identifier les nœuds qui se trouvent à une faible distance des autres ;
- Degré de centralité d'intermédiarité : identifier les nœuds important en terme de transfert d'information.

Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie

But

- Partitionner les nœuds du graphe en un nombre fini de groupes.
- Trouver des groupes de nœuds homogènes, des individus au comportement similaire.

But

- Partitionner les nœuds du graphe en un nombre fini de groupes.
- Trouver des groupes de nœuds homogènes, des individus au comportement similaire.

Idéal

- Beaucoup de connexions entre les nœuds d'une même communauté.
- Peu de connexions entre les nœuds de communautés différentes.

But

- Partitionner les nœuds du graphe en un nombre fini de groupes.
- Trouver des groupes de nœuds homogènes, des individus au comportement similaire.

Idéal

- Beaucoup de connexions entre les nœuds d'une même communauté.
- Peu de connexions entre les nœuds de communautés différentes.

Remarque

Thème très proche du clustering.

Différentes méthodes pour détecter des communautés :

- techniques basées sur la modularité.
- clustering spectral.
- méthodes probabilistes (modèles SBM).
- ...

Comment ?

Différentes méthodes pour détecter des communautés :

- techniques basées sur la modularité.
- clustering spectral.
- méthodes probabilistes (modèles SBM).
- ...

Dans cette partie

nous nous focaliserons sur le clustering spectral.

- **Cadre** : $G = (V, E)$ un graphe et on veut trouver une partition de V en **clusters** ou **communautés**.

- **Cadre** : $G = (V, E)$ un graphe et on veut trouver une partition de V en **clusters** ou **communautés**.
- Approche basée sur la **décomposition spectrale du Laplacien du graphe**.

- **Cadre** : $G = (V, E)$ un graphe et on veut trouver une partition de V en **clusters** ou **communautés**.
- Approche basée sur la **décomposition spectrale du Laplacien du graphe**.
- Approche utilisée dans un **cadre plus large** :
 - **Problème** : clustering sur un jeu de données standards $n \times p$;
 - L'approche peut être appliquée à une **matrice de similarité**.

- **Cadre** : $G = (V, E)$ un graphe et on veut trouver une partition de V en **clusters** ou **communautés**.
- Approche basée sur la **décomposition spectrale du Laplacien du graphe**.
- Approche utilisée dans un **cadre plus large** :
 - **Problème** : clustering sur un jeu de données standards $n \times p$;
 - L'approche peut être appliquée à une **matrice de similarité**.
- On pourra consulter [von Luxburg, 2017] dont cette partie est fortement inspirée.

- $G = (V, E)$ un graphe non dirigé valué avec $n = |V|$.
- $w_{ij} \geq 0$ poids de l'arête entre i et j et $W = (w_{ij})_{1 \leq i, j \leq n}$ la matrice d'adjacence.
- $d_i = \sum_{j \neq i} w_{ij}$ degré du nœud i et $D = \text{diag}(d_i)_{1 \leq i \leq n}$ la matrice des degrés.

- $G = (V, E)$ un graphe **non dirigé valué** avec $n = |V|$.
- $w_{ij} \geq 0$ **poids de l'arête** entre i et j et $W = (w_{ij})_{1 \leq i, j \leq n}$ la **matrice d'adjacence**.
- $d_i = \sum_{j \neq i} w_{ij}$ **degré** du nœud i et $D = \text{diag}(d_i)_{1 \leq i \leq n}$ la matrice des degrés.

Laplacien non normalisé

Le **Laplacien non normalisé** de G est la matrice $n \times n$ définie par :

$$L = D - W.$$

Quelques propriétés

Les deux propositions suivantes sont **fondamentales** pour l'algorithme de **clustering spectral**.

Proposition 1

1. Pour tout vecteur $f \in \mathbb{R}^n$ on a

$$f'Lf = \frac{1}{2} \sum_{1 \leq i, j \leq n} w_{ij} (f_i - f_j)^2.$$

2. L est symétrique et semi définie positive.

Quelques propriétés

Les deux propositions suivantes sont **fondamentales** pour l'algorithme de **clustering spectral**.

Proposition 1

1. Pour tout vecteur $f \in \mathbb{R}^n$ on a

$$f'Lf = \frac{1}{2} \sum_{1 \leq i, j \leq n} w_{ij} (f_i - f_j)^2.$$

2. L est symétrique et semi définie positive.
3. La **plus petite valeur propre** de L est 0, le vecteur propre correspondant est $\mathbf{1}_n$.
4. L a n valeurs propres non nulle $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$.

Proposition 2

Soit G un graphe non dirigé. Alors

1. le degrés de multiplicité k de la valeur propre 0 de L est égal au nombre de composantes connexes A_1, \dots, A_k dans G .

Proposition 2

Soit G un graphe non dirigé. Alors

1. le degrés de multiplicité k de la valeur propre 0 de L est égal au nombre de composantes connexes A_1, \dots, A_k dans G .
2. l'espace propre associé à la valeur propre 0 est engendré par les vecteurs d'indicatrices $\mathbf{1}_{A_1}, \dots, \mathbf{1}_{A_k}$.

Proposition 2

Soit G un graphe non dirigé. Alors

1. le degrés de multiplicité k de la valeur propre 0 de L est égal au nombre de composantes connexes A_1, \dots, A_k dans G .
2. l'espace propre associé à la valeur propre 0 est engendré par les vecteurs d'indicatrices $\mathbf{1}_{A_1}, \dots, \mathbf{1}_{A_k}$.

Conséquence importante

Le spectre de L permet d'identifier ses composantes connexes.

- En pratique : 1 communauté n'est pas forcément égale à une composante connexe.
- On peut par exemple vouloir extraire des communautés dans un graphe à une composante connexe.

- En pratique : 1 communauté n'est pas forcément égale à une composante connexe.
- On peut par exemple vouloir extraire des communautés dans un graphe à une composante connexe.

Idée

Considérer les k plus petites valeurs propres du Laplacien.

Algorithme

Entrées : un graphe non dirigé G , k le nombre de clusters.

1. Calculer le Laplacien non normalisé L de G .

Spectral clustering non normalisé

Algorithme

Entrées : un graphe non dirigé G , k le nombre de clusters.

1. Calculer le Laplacien non normalisé L de G .
2. Calculer les k premiers vecteurs propres u_1, \dots, u_k de G .
3. On note U la matrice $n \times k$ qui contient les u_k et y_i la i^{e} ligne de U .

Spectral clustering non normalisé

Algorithme

Entrées : un graphe non dirigé G , k le nombre de clusters.

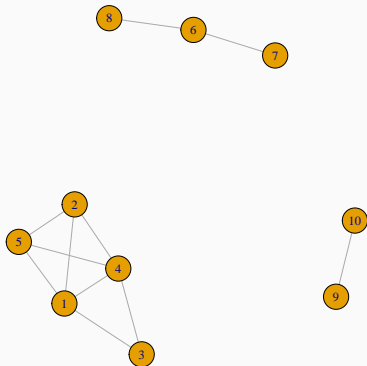
1. Calculer le Laplacien non normalisé L de G .
2. Calculer les k premiers vecteurs propres u_1, \dots, u_k de G .
3. On note U la matrice $n \times k$ qui contient les u_k et y_i la i^{e} ligne de U .
4. Faire un k -means avec les points $y_i, i = 1, \dots, n \implies A_1, \dots, A_k$.

Sortie : clusters C_1, \dots, C_k avec

$$C_j = \{i | y_i \in A_j\}.$$

Exemple

- On considère le graphe suivant à 3 composantes connexes.



- Calcul des valeurs propres du Laplacien :

```
> L <- laplacian_matrix(G,sparse=F)
> spec <- eigen(L)
> spec$values %>% round(3)
## [1] 5 5 4 3 2 2 1 0 0 0
```

- Extraction des vecteurs propres associés à la valeur propre 0 :

```
> U <- spec$vectors[,8:10]
> round(U,3)
##      [,1] [,2] [,3]
## [1,] -0.447 0.000 0.000
## [2,] -0.447 0.000 0.000
## [3,] -0.447 0.000 0.000
## [4,] -0.447 0.000 0.000
## [5,] -0.447 0.000 0.000
## [6,] 0.000 -0.577 0.000
## [7,] 0.000 -0.577 0.000
## [8,] 0.000 -0.577 0.000
## [9,] 0.000 0.000 0.707
## [10,] 0.000 0.000 0.707
```

Remarque

- L'étape k -means n'est pas nécessaire lorsque le graphe a exactement k composantes connexes mais...
- ce n'est que très rarement le cas en pratique.

Remarque

- L'étape k -means n'est pas nécessaire lorsque le graphe a exactement k composantes connexes mais...
- ce n'est que très rarement le cas en pratique.
- Si G ne possède pas k composantes connexes alors U n'est pas composé que de 1 et de 0.
- On ne peut donc pas extraire directement les composantes à cette étape.

- L'étape k -means n'est pas nécessaire lorsque le graphe a exactement k composantes connexes mais...
- ce n'est que très rarement le cas en pratique.
- Si G ne possède pas k composantes connexes alors U n'est pas composé que de 1 et de 0.
- On ne peut donc pas extraire directement les composantes à cette étape.
- Mais

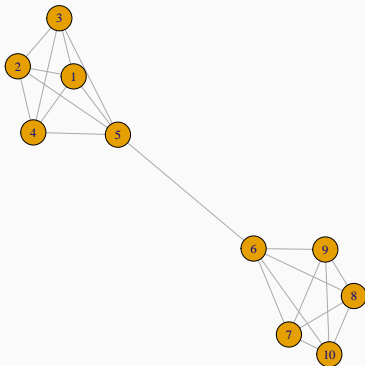
Remarque

- L'étape k -means n'est pas nécessaire lorsque le graphe a exactement k composantes connexes mais...
- ce n'est que très rarement le cas en pratique.
- Si G ne possède pas k composantes connexes alors U n'est pas composé que de 1 et de 0.
- On ne peut donc pas extraire directement les composantes à cette étape.
- Mais si il existe (presque) k composantes, alors les $y_i \in \mathbb{R}^k$ risquent de se rapprocher de cette configuration 0-1.

- L'étape k -means n'est pas nécessaire lorsque le graphe a exactement k composantes connexes mais...
- ce n'est que très rarement le cas en pratique.
- Si G ne possède pas k composantes connexes alors U n'est pas composé que de 1 et de 0.
- On ne peut donc pas extraire directement les composantes à cette étape.
- Mais si il existe (presque) k composantes, alors les $y_i \in \mathbb{R}^k$ risquent de se rapprocher de cette configuration 0-1.
- C'est pourquoi on fait un k -means en 4.

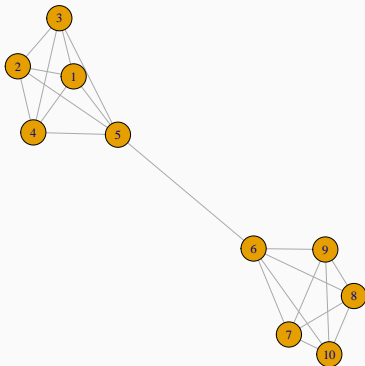
Exemple

- On considère le graphe suivant à 1 composante connexe.



Exemple

- On considère le graphe suivant à 1 composante connexe.



- mais dans lequel on a envie de faire 2 groupes !

- Calcul des valeurs propres du Laplacien :

```
> L <- laplacian_matrix(G1,sparse=F)
> spec <- eigen(L)
> spec$values %>% round(3)
## [1] 6.702 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 0.298 0.000
```

- Extraction des vecteurs propres associés aux deux plus petites valeurs propres :

```
> U <- spec$vectors[,9:10]
> round(U,3)
##           [,1]    [,2]
## [1,] -0.334 -0.316
## [2,] -0.334 -0.316
## [3,] -0.334 -0.316
## [4,] -0.334 -0.316
## [5,] -0.234 -0.316
## [6,]  0.234 -0.316
## [7,]  0.334 -0.316
## [8,]  0.334 -0.316
## [9,]  0.334 -0.316
## [10,] 0.334 -0.316
```

- Calcul des valeurs propres du Laplacien :

```
> L <- laplacian_matrix(G1,sparse=F)
> spec <- eigen(L)
> spec$values %>% round(3)
## [1] 6.702 5.000 5.000 5.000 5.000 5.000 5.000 5.000 5.000 0.298 0.000
```

- Extraction des vecteurs propres associés aux deux plus petites valeurs propres :

```
> U <- spec$vectors[,9:10]
> round(U,3)
##           [,1]    [,2]
## [1,] -0.334 -0.316
## [2,] -0.334 -0.316
## [3,] -0.334 -0.316
## [4,] -0.334 -0.316
## [5,] -0.234 -0.316
## [6,]  0.234 -0.316
## [7,]  0.334 -0.316
## [8,]  0.334 -0.316
## [9,]  0.334 -0.316
## [10,] 0.334 -0.316
```

- Etape k -means pour obtenir les groupes :

```
> kmeans(U,2,nstart=100)$cluster
```

- Il existe **plusieurs versions** d'algorithmes de **clustering spectral**.
- Les plus utilisées s'appliquent à une **version normalisée du Laplacien**, par exemple :

$$L_{\text{norm}} = I - D^{-1/2} W D^{-1/2}.$$

- Les propriétés de L_{norm} sont **proches** de celles de L . On a par exemple la propriété suivante.

- Il existe **plusieurs versions** d'algorithmes de **clustering spectral**.
- Les plus utilisées s'appliquent à une **version normalisée du Laplacien**, par exemple :

$$L_{\text{norm}} = I - D^{-1/2} W D^{-1/2}.$$

- Les propriétés de L_{norm} sont **proches** de celles de L . On a par exemple la propriété suivante.

Proposition 3

Soit G un graphe **non dirigé**. Alors

1. le **degrés de multiplicité** k de la valeur propre 0 de L_{norm} est égal au **nombre de composantes connexes** A_1, \dots, A_k dans G .

- Il existe **plusieurs versions** d'algorithmes de **clustering spectral**.
- Les plus utilisées s'appliquent à une **version normalisée du Laplacien**, par exemple :

$$L_{\text{norm}} = I - D^{-1/2} W D^{-1/2}.$$

- Les propriétés de L_{norm} sont **proches** de celles de L . On a par exemple la propriété suivante.

Proposition 3

Soit G un graphe **non dirigé**. Alors

1. le **degrés de multiplicité** k de la valeur propre 0 de L_{norm} est égal au **nombre de composantes connexes** A_1, \dots, A_k dans G .
2. l'**espace propre** associé à la valeur propre 0 est engendré par les vecteurs d'indicatrices $D^{1/2} \mathbf{1}_{A_1}, \dots, D^{1/2} \mathbf{1}_{A_k}$.

Clustering spectral normalisé

- On déduit de cette propriété la version la plus courante de clustering spectral du à [Ng et al., 2002].

Algorithme

Entrées : un graphe non dirigé G , k le nombre de clusters.

1. Calculer le Laplacien normalisé L_{norm} de G .

Clustering spectral normalisé

- On déduit de cette propriété la version la plus courante de clustering spectral du à [Ng et al., 2002].

Algorithme

Entrées : un graphe non dirigé G , k le nombre de clusters.

1. Calculer le Laplacien normalisé L_{norm} de G .
2. Calculer les k premiers vecteurs propres u_1, \dots, u_k de G . On note U la matrice $n \times k$ qui les contient.
3. Calculer T en normalisant les lignes de U : $t_{ij} = u_{ij} / (\sum_{\ell} u_{i\ell}^2)^{1/2}$.

Clustering spectral normalisé

- On déduit de cette propriété la version la plus courante de clustering spectral du à [Ng et al., 2002].

Algorithme

Entrées : un graphe non dirigé G , k le nombre de clusters.

1. Calculer le Laplacien normalisé L_{norm} de G .
2. Calculer les k premiers vecteurs propres u_1, \dots, u_k de G . On note U la matrice $n \times k$ qui les contient.
3. Calculer T en normalisant les lignes de U : $t_{ij} = u_{ij} / (\sum_{\ell} u_{i\ell}^2)^{1/2}$.
4. Faire un k -means avec les points $y_i, i = 1, \dots, n$ (i^e ligne de T) $\implies A_1, \dots, A_k$.

Sortie : clusters C_1, \dots, C_k avec

$$C_j = \{i | y_i \in A_j\}.$$

- Algorithme **quasi similaire** au clustering spectral **non normalisé**.
- Une étape de **normalisation** en plus.
- Cette étape se justifie par la **théorie de la perturbation** du spectre d'une matrice.
- On pourra consulter [[von Luxburg, 2017](#)] pour des justifications.

- Comme souvent en **clustering**, cette algorithmne nécessite de connaître le **nombre de groupes**.

- Comme souvent en **clustering**, cette algorithmne nécessite de connaître le **nombre de groupes**.
- Utilisation de **connaissances métier** pour ce choix
- ou

- Comme souvent en **clustering**, cette algorithmne nécessite de connaître le **nombre de groupes**.
- Utilisation de **connaissances métier** pour ce choix
- ou étude des **valeurs propres** du Laplacien.

Remarque importante

- L'algorithme n'utilise pas nécessairement la structure du graphe.

Remarque importante

- L'algorithme n'utilise pas nécessairement la structure du graphe.
- Il est entièrement basé sur la matrice (d'adjacence) W des poids qui contient des arêtes.

Remarque importante

- L'algorithme n'utilise pas nécessairement la structure du graphe.
- Il est entièrement basé sur la matrice (d'adjacence) W des poids qui contient des arêtes.
- Cette matrice peut également être vue comme une matrice de similarité.

Généralisation

Remarque importante

- L'algorithme n'utilise pas nécessairement la structure du graphe.
- Il est entièrement basé sur la matrice (d'adjacence) W des poids qui contient des arêtes.
- Cette matrice peut également être vue comme une matrice de similarité.

Conséquence

- On peut donc généraliser cet algorithme à n'importe quel problème où on possède une matrice de similarité.
- Exemple : problème de clustering standard sur des données $n \times p$ (il "suffit" de construire une matrice de similarité).

Clustering spectral sur un tableau de données

- **Données** : tableau $n \times p$ n individus, p variables.
- **Problème** : classification non supervisée des n individus.

Clustering spectral sur un tableau de données

- **Données** : tableau $n \times p$ n individus, p variables.
- **Problème** : classification non supervisée des n individus.
- **Méthodes classiques** : k -means, CAH...

Clustering spectral sur un tableau de données

- **Données** : tableau $n \times p$ n individus, p variables.
- **Problème** : classification non supervisée des n individus.
- **Méthodes classiques** : k -means, CAH...

Alternative : clustering spectral

1. construire un **graphe de similarité** ;
2. lancer l'algorithme de **clustering spectral** sur **ce graphe** (ou plutôt sur sa **matrice de similarité**).

Construction du graphe de similarités

- On peut utiliser les techniques standards : ε -neighborhood graph ou plus proches voisins (mutuels ou non).

Construction du graphe de similarités

- On peut utiliser les techniques standards : ε -neighborhood graph ou plus proches voisins (mutuels ou non).
- De façon plus générale, la matrice de similarités s'obtient souvent à partir d'un noyau K :

$$K : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}$$
$$(x, y) \mapsto \langle \Phi(x), \Phi(y) \rangle_{\mathcal{H}}$$

où $\Phi : \mathbb{R}^p \rightarrow \mathcal{H}$ est une fonction qui plonge les observations dans un espace de Hilbert \mathcal{H} appelé feature space.

- Linéaire (vanilladot) : $K(x, y) = \langle x, y \rangle$.
- Gaussien (rfbdot) : $K(x, y) = \exp(-\sigma \|x - y\|^2)$.
- Polynomial (polydot) : $K(x, y) = (\text{scale} \langle x, y \rangle + \text{offset})^{\text{degree}}$.
- ...

Exemples de noyau

- Linéaire (vanilladot) : $K(x, y) = \langle x, y \rangle$.
- Gaussien (rfbndot) : $K(x, y) = \exp(-\sigma \|x - y\|^2)$.
- Polynomial (polydot) : $K(x, y) = (\text{scale} \langle x, y \rangle + \text{offset})^{\text{degree}}$.
- ...

Références

On pourra trouver dans exemples de noyau dans
[Karatzoglou et al., 2004].

Matrice de similarités avec un noyau

- Etant données n observations $x_i \in \mathbb{R}^p$
- La **matrice de similarités** W associée à un **noyau** K est la matrice $n \times n$ dont le terme général vaut

$$w_{ij} = \begin{cases} K(x_i, x_j) & \text{si } i \neq j \\ 0 & \text{sinon} \end{cases}$$

Matrice de similarités avec un noyau

- Etant données n observations $x_i \in \mathbb{R}^p$
- La **matrice de similarités** W associée à un **noyau** K est la matrice $n \times n$ dont le terme général vaut

$$w_{ij} = \begin{cases} K(x_i, x_j) & \text{si } i \neq j \\ 0 & \text{sinon} \end{cases}$$

Clustering spectral

Le **clustering spectral** consiste à appliquer l'algorithme vu précédemment en calculant le **Laplacien normalisé** à partir de cette **matrice de similarités** (voir [Ng et al., 2002, Arias-Castro, 2011]).

Clustering spectral sur des données $n \times p$

Algorithme

Entrées : tableau de données $x \times x$, K un noyau, k le nombre de clusters.

1. Calculer la matrice de similarités W sur les données avec le noyau K .
2. Calculer le Laplacien normalisé L_{norm} à partir de W .

Clustering spectral sur des données $n \times p$

Algorithme

Entrées : tableau de données $x \times x$, K un noyau, k le nombre de clusters.

1. Calculer la matrice de similarités W sur les données avec le noyau K .
2. Calculer le Laplacien normalisé L_{norm} à partir de W .
3. Calculer les k premiers vecteurs propres u_1, \dots, u_k de G . On note U la matrice $n \times k$ qui les contient.
4. Calculer T en normalisant les lignes de U : $t_{ij} = u_{ij} / (\sum_{\ell} u_{i\ell}^2)^{1/2}$.
5. Faire un k -means avec les points $y_i, i = 1, \dots, n$ (i^e ligne de T) $\implies A_1, \dots, A_k$.

Sortie : clusters C_1, \dots, C_k avec

$$C_j = \{i | y_i \in A_j\}.$$

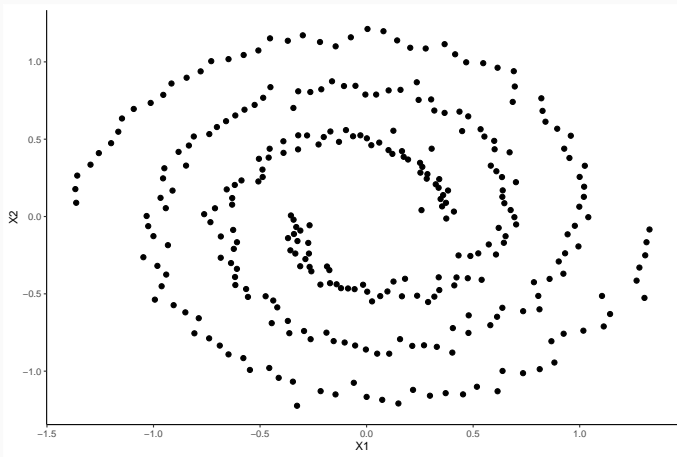
- La fonction `specc` du package `kernlab` permet de faire le clustering spectral.

- La fonction `specc` du package `kernlab` permet de faire le clustering spectral.
- Exemple : données `spirals`

```
> library(kernlab)
> data(spirals)
> spirals1 <- data.frame(spirals)
> head(spirals1)
##           X1           X2
## 1  0.8123568 -0.98712687
## 2 -0.2675890 -0.32552004
## 3  0.3739746 -0.01293652
## 4  0.2576481  0.04130805
## 5 -0.8472613  0.32939461
## 6  0.4097649  0.03205686
```

Visualisation du nuage de points

```
> ggplot(spirals1)+aes(x=X1,y=X2)+geom_point()+theme_classic()
```

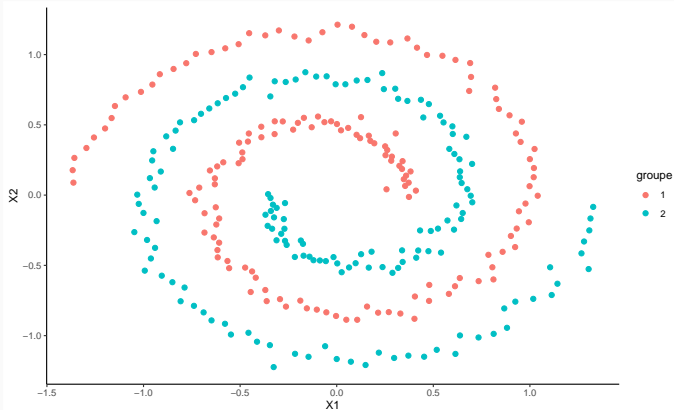


Le clustering spectral

```
> groupe <- specc(spirals,centers=2,kernel="rbfdot")
> head(groupe)
## [1] 2 2 1 1 2 1
> spirals1 <- spirals1 %>% mutate(groupe=as.factor(groupe))
> ggplot(spirals1)+aes(x=X1,y=X2,color=groupe)+geom_point(size=2)+theme_classic()
```

Le clustering spectral

```
> groupe <- specc(spirals,centers=2,kernel="rbfdot")  
> head(groupe)  
## [1] 2 2 1 1 2 1  
> spirals1 <- spirals1 %>% mutate(groupe=as.factor(groupe))  
> ggplot(spirals1)+aes(x=X1,y=X2,color=groupe)+geom_point(size=2)+theme_classic()
```



Apprentissage non supervisé - rappels

Contexte - cadre mathématique

Les classiques

Quelques notions sur les graphes

Définitions - vocabulaire sur les graphes

Statistiques descriptives sur les graphes

Caractéristiques générales

Importance des nœuds

Détection de communautés

Bibliographie



Arias-Castro, E. (2011).

Clustering based on pairwise distances when the data is of mixed dimensions.

IEEE Transaction on Information Theory, 57(3) :1692–1706.



Bahoken, F., Beauguitte, L., and Lhomme, S. (2013).

La visualisation des réseaux. Principes, enjeux et perspectives.





halshs-00839905.



Blondel, V. D., Guillaume, J., Lambiotte, R., and Lefebvre, E. (2008).

Fast unfolding of communities in large networks.

Journal of Statistical Mechanics : Theory and Experiment.

-  Daudin, J.-J., Picard, F., and Robin, S. (2008).
A mixture model for random graphs.
Statistics and computing, 18 :173–183.
-  Fortunato, S. (2010).
Community detection in graphs.
Physics report, 486 :75–174.
-  Girvan, M. and Newman, M. E. J. (2002).
Community structure in social and biological networks.
Proc. Natl. Acad. Sci., pages 7821–7826.
-  Karatzoglou, A., Hornik, K., Smola, A., and Zeileis, A. (2004).
kernlab – an s4 package for kernel methods in r.
Journal of Statitstical Software, 11(9).



Ng, A., Jordan, M., and Weiss, Y. (2002).

On spectral clustering analysis.

In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 849–856.



von Luxburg, U. (2017).

A tutorial on spectral clustering.

Statistics and computing, 17 :395–416.