

TP Analyse de données - Apprentissage supervisé (Régression)

INTRODUCTION

Nous allons utiliser le logiciel R ([documentation](#) ; possibilité d'utiliser Rstudio). Dans un dossier *AnalyseDeDonnees* créer deux sous dossiers :

- Code - dans lequel vous placerez vos fichiers de code
- Data - dans lequel vous placerez les fichiers du répertoire "Data" (à télécharger via Moodle)

Les 3 TP *Analyse de données* présentent différentes méthodes d'analyse de données : le but n'est pas de finir les TP le plus vite possible mais d'analyser les résultats! Un rapport de 1 page **maximum** vous est demandé pour chacun des 3 TP : ne choisir que les résultats les plus intéressants et les **commenter**.

Ne pas hésiter à utiliser l'aide de R grâce à la commande :

```
1 help(...)
```

RÉGRESSION SIMPLE

0. Télécharger les cours sur la régression linéaire simple et la régression linéaire multiple.

1. Créer un fichier *regression-simple.R* et copier les lignes suivantes :

```
1 # Adresse du dossier où vous travaillez
2 setwd("/Users/.../TP/TP/Code")
3
4 # Packages utilisés dans la suite
5 library(MASS)
6 require(pls)
7 require(splines)
8
9 # Supprimer toutes les variables
10 rm(list=ls(all=TRUE))
```

2. Nous allons utiliser des données sur la ville de Boston. Commencer par afficher (et étudier les données).

```
1 # Utilisation de données sur data
2 # Affichage des informations
3 ?Boston
4
5 # Affichage des données
6 print(Boston)
```

3. Transformer les données (pour que le code fonctionne quel que soient les données) et définir les paramètres :

```
1 # Transformation des données
2 data <- Boston
3 data <- data.frame(y=Boston$medv,x1=Boston$lstat,x2=Boston$age,
4                   x3=Boston$crim,x4=Boston$zn,x5=Boston$indus,
5                   x6=Boston$chas,x7=Boston$nox,x8=Boston$rm,
6                   x9=Boston$dis,x10=Boston$rad,x11=Boston$tax,
7                   x12=Boston$ptratio,x13=Boston$black)
8
9 # Paramètres
10 n <- length(data$y)
11 alpha <- 0.05
```

4. Mettre en place la régression linéaire simple :

```
1 ## Mise en place de la régression linéaire [SIMPLE]
2 # Peut on utiliser x1 = pourcentage de la population pauvre
3 # pour prédire y = valeur médiane des maisons en milliers de dollars.
4 simpleLinearReg <- lm(y~x1, data=data)
```

5. Afficher le résultat de la régression linéaire :

```
1 # Affichage du résultat de la régression linéaire
2 # épaisseur de la ligne = 2 ; couleur de la ligne = rouge
3 plot(y~x1, data=data)
4 abline(simpleLinearReg,lwd=2,col="red")
```

6. Afficher les résidus en fonction de la prédiction :

```
1 # Affichage des résidus en fonction de la prédiction
2 plot(simpleLinearReg$fitted.values, simpleLinearReg$residuals)
3 abline(0,0)
```

7. Afficher les valeurs prédites en fonction des valeurs observées :

```
1 # Affichage des valeurs prédites en fonction des valeurs observées
2 plot(simpleLinearReg$fitted.values,data$y)
3 abline(0,1)
```

8. Calculer le risque à 5% :

```
1 # Affichage du résultat : calculer le risque à 5 % avec :
2 # la t-value / la p-value/ la statistique de Fisher
3 summary(simpleLinearReg)
4 # Risque à 5 % (pour la t-value / la statistique de Fisher)
5 qt(1-alpha/2, n-2)
6 qf(1-alpha/2, 1, n-2)
```

9. Étudier l'intervalle de confiance des paramètres estimés :

```
1 # Intervalle de confiance des paramètres estimés
2 # Risque à 5 % avec l'intervalle de confiance
3 confint(simpleLinearReg)
```

10. Étudier l'adéquation au modèle avec R^2 :

```
1 # Adéquation au modèle avec le R^2
2 summary(simpleLinearReg)
```

11. Prédire une valeur ultérieure (int. de confiance de l'estim. paramétrique et de la prédiction) :

```
1 # Prédiction d'une valeur ultérieure (valeur de x1 testée = 10)
2 # Intervalle de confiance pour la prédiction de y pour une valeur donnée de x1
3 predict(simpleLinearReg,data.frame(x1=10), interval="confidence")
4 # Intervalle de prédiction pour la prédiction de y pour une valeur donnée de x1
5 predict(simpleLinearReg,data.frame(x1=10), interval="prediction")
```

12. Tracer les intervalles de confiance :

```
1 # Affichage de l'intervalle de confiance et de prédiction
2 seqx1 <- seq(min(data$x1),max(data$x1),length=50)
3 intpred <- predict(simpleLinearReg,data.frame(x1=seqx1),
4                   interval="prediction")[,c("lwr","upr")]
5 intconf <- predict(simpleLinearReg,data.frame(x1=seqx1),
6                   interval="confidence")[,c("lwr","upr")]
7 plot(data$y~data$x1,xlab="x1",ylab="y")
8 abline(simpleLinearReg)
9 matlines(seqx1,cbind(intconf,intpred),lty=c(2,2,3,3),
10         col=c("red","red","blue","blue"),lwd=c(2,2))
11 legend("bottomright",lty=c(2,3),lwd=c(2,1), c("conf","pred"),col=c("red","blue"))
```

13. Tester la normalité des résidus :

```
1 # Test de normalité des résidus
2 shapiro.test(resid(simpleLinearReg))
```

14. Valider le modèle par validation croisée :

```
1 # Validation du modèle par validation croisée
2 MSE <- 0
3 for (i in 1:n)
4 {
5   datatopredict <- data$y[i]
6   datatemp <- data[-c(i),]
7   reg <- lm(y~x1, data=datatemp)
8   predictedvalue <- predict(reg,data.frame(x1=data$x1[i]), interval="prediction")
9   MSE <- MSE+(datatopredict-predictedvalue[1])^2
10 }
11 MSE <- MSE/n
12 cat("Valeur du résidu avec la validation croisée", MSE)
```

15. Tester si ce n'est pas plutôt non linéaire ... Tester le cas polynomial (possible de jouer avec le degré du polynôme) ... Étudier les différences :

```
1 ## Et le non linéaire ?
2 ## Cas polynomial (simple = une variable)
3 degpoly <- 2
4 simplePolyReg <- lm(y~poly(x1,degpoly), data=data)
5
6 # Risque à 5 % (pour la t-value / la statistique de Fisher)
7 qt(1-alpha/2, n-2)
8 qf(1-alpha/2, 1, n-2)
9
10 # Intervalle de confiance des paramètres estimés
11 # Risque à 5 % avec l'intervalle de confiance
12 confint(simplePolyReg)
13
14 # Adéquation au modèle avec le R^2
15 summary(simplePolyReg)
16
17 # Affichage de l'intervalle de confiance et de prédiction
18 seqx1 <- seq(min(data$x1),max(data$x1),length=50)
19 intpred <- predict(simplePolyReg,data.frame(x1=seqx1),
20                   interval="prediction")[,c("lwr","upr")]
21 intconf <- predict(simplePolyReg,data.frame(x1=seqx1),
22                   interval="confidence")[,c("lwr","upr")]
23 plot(data$y~data$x1,xlab="x1",ylab="y")
24 pred <- predict(simplePolyReg,data.frame(x1=sort(data$x1)))
25 lines(sort(data$x1),pred,lwd = 2)
26 matlines(seqx1,cbind(intconf,intpred),lty=c(2,2,3,3),
27          col=c("red","red","blue","blue"),lwd=c(2,2))
28 legend("bottomright",lty=c(2,3),lwd=c(2,1), c("conf","pred"),col=c("red","blue"))
29
30 # Validation du modèle par validation croisée
31 MSE <- 0
32 for (i in 1:n)
33 {
34   datatopredict <- data$y[i]
35   datatemp <- data[-c(i),]
36   reg <- lm(y~poly(x1,degpoly), data=datatemp)
37   predictedvalue <- predict(reg,data.frame(x1=data$x1[i]), interval="prediction")
38   MSE <- MSE+(datatopredict-predictedvalue[1])^2
39 }
40 MSE <- MSE/n
41 # Valeur du résidu avec la validation croisée
42 print(MSE)
```

16. Obtient-on de meilleurs résultats en utilisant des splines à la place des polynômes ? Tester (jouer aussi avec le degré) :

```
1 ## Et le non linéaire ?
2 ## Cas spline (simple = une variable)
3 degoffreedom <- 4
4 simpleSplineReg <- lm(y~ns(x1,degoffreedom), data=data)
5
6 # Risque à 5 % (pour la t-value / la statistique de Fisher)
7 qt(1-alpha/2, n-2)
8 qf(1-alpha/2, 1, n-2)
9
10 # Intervalle de confiance des paramètres estimés
11 # Risque à 5 % avec l'intervalle de confiance
12 confint(simpleSplineReg)
13
14 # Adéquation au modèle avec le R^2
15 summary(simpleSplineReg)
16
17 # Affichage de l'intervalle de confiance et de prédiction
18 seqx1 <- seq(min(data$x1),max(data$x1),length=50)
19 intpred <- predict(simpleSplineReg,data.frame(x1=seqx1),
20                   interval="prediction")[,c("lwr","upr")]
21 intconf <- predict(simpleSplineReg,data.frame(x1=seqx1),
22                   interval="confidence")[,c("lwr","upr")]
23 plot(data$y~data$x1,xlab="x1",ylab="y")
24 pred <- predict(simpleSplineReg,data.frame(x1=sort(data$x1)))
25 lines(sort(data$x1),pred,lwd = 2)
26 matlines(seqx1,cbind(intconf,intpred),lty=c(2,2,3,3),
27          col=c("red","red","blue","blue"),lwd=c(2,2))
28 legend("bottomright",lty=c(2,3),lwd=c(2,1), c("conf","pred"),col=c("red","blue"))
29
30 # Validation du modèle par validation croisée
31 MSE <- 0
32 for (i in 1:n)
33 {
34   datatopredict <- data$y[i]
35   datatemp <- data[-c(i),]
36   reg <- lm(y~ns(x1,degoffreedom), data=datatemp)
37   predictedvalue <- predict(reg,data.frame(x1=data$x1[i]), interval="prediction")
38   MSE <- MSE+(datatopredict-predictedvalue[1])^2
39 }
40 MSE <- MSE/n
41 # Valeur du résidu avec la validation croisée
42 print(MSE)
```

17. Tester enfin ce qui se passe en utilisant la technique des *smoothing splines* :

```
1 ## Cas smoothing spline (simple = une variable)
2 degoffreedom <- 6
3 simpleSmoothSplineReg <- smooth.spline(data$x1,data$y,df=degoffreedom)
4
5 # Affichage de l'intervalle de confiance et de prédiction
6 plot(data$y~data$x1,xlab="x1",ylab="y")
7 pred <- predict(simpleSmoothSplineReg,sort(data$x1))
8 lines(pred)
9
10 # Test de normalité des résidus
11 shapiro.test(resid(simpleSmoothSplineReg))
12
13 # Validation du modèle par validation croisée
14 MSE <- 0
15 for (i in 1:n)
16 {
17   datatopredict <- data$y[i]
18   datatemp <- data[-c(i),]
19   reg <- smooth.spline(datatemp$x1,datatemp$y,df=degoffreedom)
20   predictedvalue <- predict(reg,data$x1[i])
21   MSE <- MSE+(datatopredict-predictedvalue$y)^2
22 }
23 MSE <- MSE/n
24 # Valeur du résidu avec la validation croisée
25 print(MSE)
```

RÉGRESSION MULTIPLE

0. Continuer avec le cours sur la régression linéaire multiple.

1. Créer un fichier *regression-multiple.R* et copier les lignes suivantes :

```
1 # Adresse du dossier où vous travaillez
2 setwd("/Users/.../TP/TP/Code")
3
4 # Packages utilisés dans la suite
5 library(MASS)
6 require(pls)
7
8 # Supprimer toutes les variables
9 rm(list=ls(all=TRUE))
```

2. Nous allons utiliser de nouveau les données sur la ville de Boston. Commencer par afficher (et étudier les données).

```
1 # Utilisation de données sur data
2 # Affichage des informations
3 ?Boston
```

```

4 |
5 | # Affichage des données
6 | print(Boston)
7 |
8 | # Transformation des données
9 | data <- Boston
10 | data <- data.frame(y=Boston$medv,x1=Boston$lstat,x2=Boston$age,
11 |                  x3=Boston$crim,x4=Boston$zn,x5=Boston$indus,
12 |                  x6=Boston$chas,x7=Boston$nox,x8=Boston$rm,
13 |                  x9=Boston$dis,x10=Boston$rad,x11=Boston$tax,
14 |                  x12=Boston$ptratio,x13=Boston$black)
15 |
16 | # Paramètres
17 | n <- length(data$y)
18 | alpha <- 0.05

```

3. Mettre en place la régression linéaire **simple** avec la variable x_1 (pourcentage de personnes pauvres) pour faire une comparaison dans la suite.

```

1 | ## Mise en place de la régression linéaire [SIMPLE]
2 | simpleLinearReg <- lm(y~x1, data=data)

```

4. Mettre en place une régression linéaire **multiple** avec les variables x_1 et x_2 (âge de la population).

```

1 | ## Mise en place de la régression linéaire [MULTIPLE]
2 | linearReg <- lm(y~x1+x2, data=data)

```

5. Afficher les résultats.

```

1 | # Affichage du résultat
2 | summary(linearReg)

```

6. Calculer le risque à 5% en testant la nullité des coefficients β_j du modèle de régression.

```

1 | # Risque à 5% : tester la nullité des coefficients du modèle de régression.
2 | numOfVariables <- 2
3 | qt(1-alpha/2, n-numOfVariables-1)
4 | qf(1-alpha/2, numOfVariables, n-numOfVariables-1)

```

7. Afficher l'intervalle de confiance.

```

1 | # Intervalle de confiance
2 | confint(linearReg)

```

8. Utiliser la méthode anova pour tester la contribution jointe des variables x_1 et x_2 .

```

1 | numOfVariablesToTest = 1
2 | qf(1-alpha/2, numOfVariablesToTest, n-numOfVariables-1)
3 | anova(simpleLinearReg,linearReg)
4 | simpleLinearRegx2 <- lm(y~x2, data=data)
5 | anova(simpleLinearRegx2,linearReg)

```

9. Prédire le cas où $(x_1 = 10, x_2 = 72)$ en utilisant la régression multiple (afficher les intervalles de confiance et de prédiction).

```
1 # Prédiction
2 predict(linearReg,data.frame(x1=10,x2=72), interval="confidence")
3 predict(linearReg,data.frame(x1=10,x2=72), interval="prediction")
```

10. Afficher les résidus en fonction de la prédiction.

```
1 # Affichage des résidus en fonction de la prédiction
2 plot(linearReg$fitted.values, linearReg$residuals)
3 abline(0,0)
```

11. Tester la normalité des résidus.

```
1 # Test de normalité des résidus
2 shapiro.test(resid(linearReg))
```

12. Sélectionner les variables (en utilisant TOUTES les variables du jeu de données).

```
1 ## Selection de variables (en utilisant TOUTES les variables)
2 fullLinearReg <- lm(y~., data=data)
3 back <- step(fullLinearReg, direction="backward", trace = 1)
4 formula(back)
5 null <- lm(y ~ 1, data=data)
6 forw <- step(null, scope=list(lower=null,upper=fullLinearReg),
7                       direction="forward", trace = 1)
8 formula(forw)
```

13. Utiliser l'ACP pour réduire la dimension du problème. Combien de composantes doit-on garder pour expliquer 80% de la variance ?

```
1 # Utilisation de l'ACP pour réduire la dimension du problème
2 # Test sur validation croisée
3 # Combien de composantes pour avoir 80 % de variance expliquée ?
4 redDim = pcr(y~.,data=data,scale=TRUE,validation="CV")
5 summary(redDim)
```