Analyses statistiques avec R

Manuel destiné aux étudiants du cours LBIRA2110 Biométrie mais en général à d'autres utilisateurs d'outils statistiques en science expérimentale

Evrard Louis - l.evrard@student.uclouvain.be - LSBA - UCLouvain
Govaerts Bernadette - bernadette.govaerts@uclouvain.be - LSBA - ISBA - UCLouvain

Contents

1 Introduction **5**
- 1.1 Installation . . . 5
- 1.2 Bases du language R . . . 5
- 1.3 Packages . . . 6
- 1.4 Importer des données . . . 7
- 1.5 Dataframe . . . 8
- 1.6 Rmarkdown . . . 9

2 Statistique descriptive **11**
- 2.1 Numérique . . . 12
- 2.2 Graphique . . . 14

3 Inference statistique sur une et deux variables **23**
- 3.1 Données Chocolat . . . 23
- 3.2 Arguments utiles dans les fonctions d'inférence . . . 24
- 3.3 Tests sur une ou deux moyennes . . . 25
- 3.4 Tests sur une ou plusieurs variances . . . 26
- 3.5 Tests sur une ou plusieurs proportions . . . 27
- 3.6 Tests de corrélation / indépendance . . . 28
- 3.7 Intervalles de confiance . . . 29

4 Modeles lineaires **31**
- 4.1 Modélisation LM . . . 31
- 4.2 Visualiser les données avant la modélisation LM . . . 32
- 4.3 Fonctions utiles sur un objet de type `mod <- lm(..)` . . . 32
- 4.4 Données du chapitre LM . . . 33
- 4.5 Utiliser `summary()` sur lm . . . 34
- 4.6 Utiliser `drop1()` / `update()` sur lm . . . 35
- 4.7 Utiliser `anova() / Anova()` sur lm . . . 36
- 4.8 Utiliser `plot()`sur lm . . . 37
- 4.9 Utiliser `abline()` sur lm . . . 37
- 4.10 Utiliser `visreg()` sur lm . . . 38
- 4.11 Utiliser `emmip()` sur lm . . . 43
- 4.12 Utiliser `predict()` sur lm . . . 44

4.13 Utiliser `lsmeans()` et le package `emmeans` sur `lm` 45

5 Regression logistique **49**
5.1 Modélisation GLM 49
5.2 Visualiser les données avant modélisation GLM 50
5.3 Fonctions utiles sur un objet de type `mod <- glm()` 50
5.4 Données du chapitre GLM 50
5.5 Utiliser `summary()` sur glm 51
5.6 Utiliser `drop1()`/`update()` sur glm 52
5.7 Utiliser `residuals()` sur glm 52
5.8 Utiliser `visreg()` sur glm 53
5.9 Utiliser `predict()` sur glm 55

6 Analyse en composantes principales ACP **57**
6.1 Fonctions utiles pour l'ACP 57
6.2 Données du chapitre ACP 58
6.3 Utiliser `prcomp()` 59
6.4 Utiliser `fviz_pca_var()` 60
6.5 Utiliser `fviz_pca_ind()` 61
6.6 Utiliser `biplot()` 62
6.7 Utiliser `fviz_contrib()` 63

7 Clustering **67**
7.1 Fonctions utiles pour le clustering 67
7.2 Données du chapitre Clustering 68
7.3 Application des fonctions `hcut()` et `kmeans()` 69
7.4 Aide au choix du nombre de clusters 70
7.5 Utiliser `fviz_dend()` 71
7.6 Utiliser `fviz_cluster()` 71

8 Modeles mixtes **73**
8.1 Modélisation LMM 73
8.2 Fonctions utiles sur un objet de type `mod <- lmer()` 73

Chapter 1

Introduction

Durant votre cursus du cours LBIRA2110, nous vous offrons la chance de vous former au logiciel et langage de programmation R.

R c'est LE logiciel libre référence dans le domaine de l'analyse statistique de données. Programmer en R nécessite R-Studio qui est une interface utilisateur facilitant l'utilisation du langage R.

1.1 Installation

L'installation de R et R-studio n'est pas très compliquée mais requiert de suivre à la lettre plusieurs étapes. Le lien suivant vous mènera vers un support où l'installation est détaillée:

~> **Installation détaillée de R et R-Studio**

En suivant scrupuleusement ces étapes vous ne devriez avoir aucun soucis!

1.2 Bases du language R

R est un langage de programmation à part entière. Avant de vouloir réaliser des analyses statistiques il faut apprendre les bases. Pour vous familiariser avec le langage à l'UCLouvain, la société **DataCamp** vous donne l'occasion de suivre une série de cours en ligne gratuitement. Data Camp est une plateforme de cours de R et Python en ligne développée par une équipe Belge de la région de Leuven.

Dans le cadre du cours de biométrie, nous vous proposerons de suivre pour commencer le cours **Introduction to R** composé de 6 modules. Utilisez bien le lien fourni par votre titulaire du cours Biométrie pour vous inscrire afin de

bénéficier d'un accès gratuit aux cours Data Camp. Veillez à bien utiliser votre adresse UCLouvain pour cette inscription.

Voici le lien d'autoinscription si vous suivez le cours de biométrie : **DataCamp**

ATTENTION vous devez impérativement vous inscrire avec votre adresse UCLouvain

De nombreux autres modules existent et nous vous encourageons à les découvrir. Citons par exemple :

- **Intermediate R**
- **Introduction to writing functions in R**
- **Introduction to importing data in R**
- **Data Visualization in R**
- **Introduction to data visualization with ggplot2**
- **Introduction to tidyverse**

En complément à Data Camp, les liens suivant vous aideront dans votre apprentissage.

- Apprendre les bases ~> **Exploration de données avec R par A. El Ghouch**
- Cheatsheet R ~> **Cheatsheet**

1.3 Packages

R contient de base une multitude de fonctions. Mais étant un langage de programmation communautaire, il dispose également d'une multitude d'extensions créées par des utilisateurs. Ces extensions sont appelées packages.

Un package contient bon nombre de fonctions et/ou de jeux de données à utiliser pour réaliser les analyses souhaitées.

- Apprendre à gérer des packages ~> **Bookdown**

Dans le cadre de ce cours nous utiliserons essentiellement les packages ci-dessous:

PACKAGE	
readxl	Importer des données depuis des fichiers Excel
rmarkdown	Convertir directement vos codes et sorties en rapports depuis Rstudio
pander	Rendre les sorties R plus esthétiques dans les rapports Rmarkdown

PACKAGE	
dplyr	Manipuler des données
car - Hmisc - EnvStats - epitools	Diverses fonctions pour l'inférence, la modélisation et l'analyse de données
emmeans	Estimations et inférence sur des contrastes en modélisation
visreg	Visualiser une régression
sjPlot	Collection de fonctions pour visualiser des données
FactoMineR	Réaliser une analyse en composantes principales
factoextra	Analyser des données multivariées et visualiser les résultats

Le plus simple est d'installer tous les packages en une fois sur votre ordinateur en début d'année via le menu [**Tools -> Install**] Packages.

Un fois un package installé, vous devez toujours le charger quand vous l'utilisez dans un code/script donné via la fonction :

```
library(nom-du-package)
```

1.4 Importer des données

Lorsqu'on dispose de données, il faut tout d'abord les importer dans R pour ensuite les analyser. En fonction du type de fichier dans lequel sont stockées les données, la fonction à utiliser est différente.

Pour importer des données ..

```
# Depuis un fichier de type .txt
data <- read.table(file = "path/file.txt", header = TRUE / FALSE, sep = .. , dec = .. )

# Depuis un fichier de type .csv
data <- read.csv(file = "path/file.csv", header = TRUE / FALSE, sep = .. , dec = .. )

# Depuis un fichier de type .xlsx
data <- readxl::read_xlsx(path = "path/file.xlsx", sheet = .. , col_names = TRUE / FALSE)
```

Quelques remarques

- Si les observations sont séparées par TAB, utiliser `sep="\t"`
- Si il y a une observation par case dans un fichier .csv, utiliser `sep=";"`

L'objet `data` résultant de cette lecture est un objet de type dataframe.

1.5 Dataframe

Le dataframe est une structure centrale de R. C'est comme son nom l'indique un tableau de données pour lequel les colonnes correspondent à une variable et les lignes à un individu.

- Apprendre plus sur les dataframes ~> **Dataframe**

1.5.1 Préparer un dataframe

Supposons que nous disposions d'un dataframe `df`. Avant de commencer une quelconque analyse il faut le préparer.

Noms des variables S'ils ne sont pas valides (ex: un nom de variable en 2 mots ou contenant des caractères spéciaux) ou bien simplement s'ils ne sont pas très clairs il est possible de renommer soi-même les variables.

```
colnames(df) <- c("Var1", "Var2", .., "VarM") # M variables dans df
```

Encodage des variables qualitatives Celles-ci doivent être encodées commes des facteurs. Un niveau correspond à une modalité. Par défaut les variables qualitatives sont encodées comme des chaines de caractères (ou bien comme des nombres si les modalités sont représentées par des chiffres).

```
data$VarQuali <- as.factor(data$VarQuali) # Pour une variable qualitative nommée VarQu
```

Données manquantes La façon la plus simple de gérer cela est d'omettre chaque ligne de df où des données sont manquantes. La plus part des fonctions ne gèrent pas les données manquantes. C'est un moyen efficace pour éviter les erreurs mais qui peut bien-sûr vous faire perdre beaucoup d'information.

```
df <- na.omit(df)
```

1.5.2 Manipulation

Pour manipuler les données contenues dans un dataframe, des fonctions de base existent ainsi que le package `dplyr` pour ceux qui veulent aller plus loin. Ce package contient quelques fonctions très intuitives pour la manipulation d'un dataframe.

```
# Filtrer les individus
subset(df, subset=condition)
dplyr::filter(df, ..)

# Sélectioner des variables
df$X
subset(df, select=...)
dplyr::select(df, ..)
```

```r
# Ajouter une nouvelle variable
cbind(df,...)
dplyr::mutate(df, ..)

# Transformer une variable existante
transform(df,...)
dplyr::transmute(df, ..)
```

1.5.3 Création

Une autre façon que l'importation pour créer un dataframe est tout simplement de le créer à la main.

```r
df <- data.frame("Var1" = ..,
                 "Var2" = ..,
                 "Var3" = ..)

# A gauche de l'égalité le nom de chaque variable et à droite un vecteur.
# /!\ Les vecteurs à droite doivent tous être de même taille.
```

1.6 Rmarkdown

Rmarkdown est une extension de R permettant de construire directement des rapports HTML, PDF ou Word depuis R incluant le code utilisé ainsi que les résultats obtenus.

- Apprendre Rmarkdown ~> **Rmarkdown**
- Cheatsheet Rmd ~> **CheatSheet**

Chapter 2

Statistique descriptive

Que faut-il faire pour décrire numériquement et / ou visualiser tel type de variable?

Types	Informations numériques	Graphes
1 variable quantitative	*Résumé statistique* *	*Histogramme si continue* *Bar plot si discrète* *Box plot* *QQ plot*
2 variables quantitatives	*Coefficient de corrélation*	*Scatter plot*
1 variable qualitative	*Médiane si ordinale* *Table de fréquence* *Table de proportions*	*Bar plot*
2 variables qualitatives	*Médiane par modalités si ordinale* *Table de contingence* *Table de proportions croisées*	*Bar plot par modalités* *Mosaic plot*
1 variable quantitative + 1 variable qualitative	*Résumé statistique* *par modalités*	*Box plot par modalités* *Bar plot par modalités si var num discrète*
Autres	*Matrice de corrélation* *Matrice de Variance-Covariance*	*Box plot par modalités croisées* *Scatter plot par modalités* *Matrice de scatter plots*

* Un résumé statistique en R est un tableau indiquant les informations suivantes
Minimum - 1er quartile - Médiane - Moyenne - 3eme quartile - Maximum

Afin d'illuster certaines fonctions nous allons utiliser le jeu de données suivant:

```
set.seed(123)
cat1 <- c(rep("H",25), rep("F",25))
```

```
cat2 <- c(rep("Grp1",17), rep("Grp2",17), rep("Grp3",16))
Quanti1 = rnorm(50,4,2)
df <- data.frame(varQuanti = runif(50,0,10),
                 varQuanti1 = Quanti1,
                 varQuanti2= Quanti1+rnorm(50,0,1),
                 varQuali = sample(cat1, replace=TRUE),
                 varQuali1 = sample(cat1, replace=TRUE),
                 varQuali2 = sample(cat2, replace=TRUE))
# On extrait uniquement les variables quantitatives de df
df_quanti <- subset(df, select=c(varQuanti, varQuanti1, varQuanti2))
```

Ce sont des données fictives, inventées de toute pièce. Il y a 3 variables quantitatives: `varQuanti` `varQuanti1` `varQuanti2` ainsi que 3 variables qualitatives `varQuali` `varQuali2` `varQuali3`. Ces données se trouvent dans un dataframe nommé `df` et les 3 variables quantitatives uniquement dans `df_quanti`.

2.1 Numérique

2.1.1 Tendance centrale

- Moyenne ~> `mean()`
- Médiane ~> `median()`

2.1.2 Variabilité

- Minimum / maximum ~> `min()` / `max()`
- Range ~> `range()`
- Variance ~> `var()`
- Ecart-type ~> `sd()`
- Covariance ~> `cov()`

2.1.3 Quantiles

- Quantile ~> `quantile()`

```
quantile(df$varQuanti, p=0.25) # Quantile d'ordre 0.25
```

- Ecarte interquartile ~> `IQR()`

2.1.4 Autres

- Résumé statistique ~> `summary()`

```
res <- summary(df$varQuanti)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.1047	2.596	5.049	5.154	7.358	9.842

- Table de fréquence ou contingence ~> `table()`
- Table de proportion ~> `prop.table()`

```
# Fréquence
res <- table(df$varQuali)
```

F	H
22	28

```
# Contingence
res <- table(df$varQuali1, df$varQuali2)
```

	Grp1	Grp2	Grp3
F	2	9	10
H	8	10	11

- Table de proportion ~> `prop.table()`

```
res <- prop.table( table(df$varQuali) )
```

F	H
0.44	0.56

- Coefficient / matrice de corrélation ~> `cor()`

```
# Matrice de corrélation des 3 variables quantitatives
res <- cor(df_quanti)
```

	varQuanti	varQuanti1	varQuanti2
varQuanti	1	0.06801	0.02392
varQuanti1	0.06801	1	0.9014
varQuanti2	0.02392	0.9014	1

- Matrice de Variance-Covariance ~> `cov()`

```
# Matrice de corrélation
res <- cov(df_quanti)
```

	varQuanti	varQuanti1	varQuanti2
varQuanti	8.645	0.3703	0.1364
varQuanti1	0.3703	3.429	3.237
varQuanti2	0.1364	3.237	3.761

- Informations par modalités ~> `tapply()`

```
res <- tapply(df$varQuanti, df$varQuali, FUN = summary) # summary de df$varQuanti par
```

- **F**:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.6072	2.226	3.606	4.564	7.358	9.353

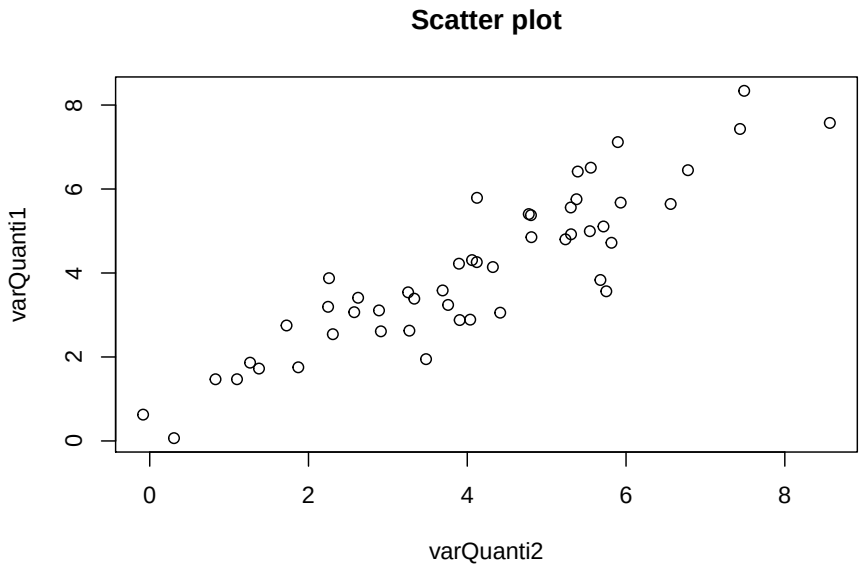
- **H**:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.1047	3.603	5.746	5.617	7.63	9.842

2.2 Graphique

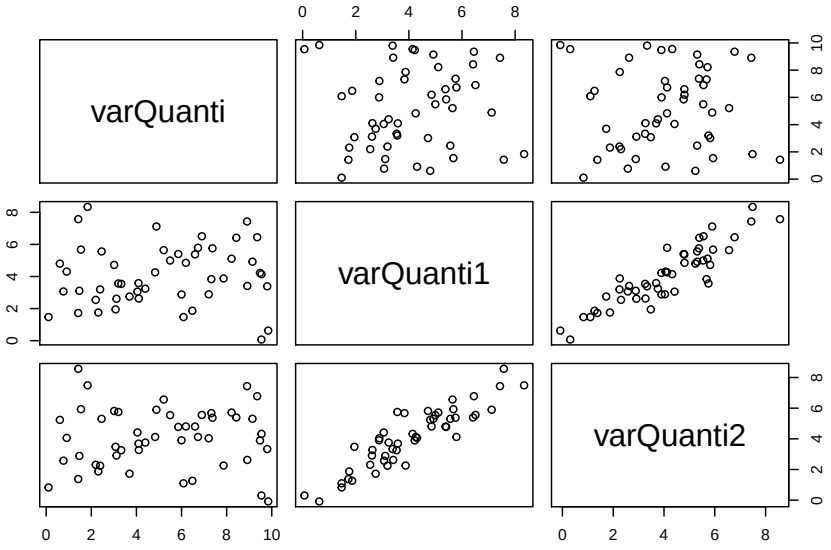
```
plot(varQuanti1 ~ varQuanti2, data = df, main = "Scatter plot")
```

2.2.1 Scatter plot ~> `plot()`

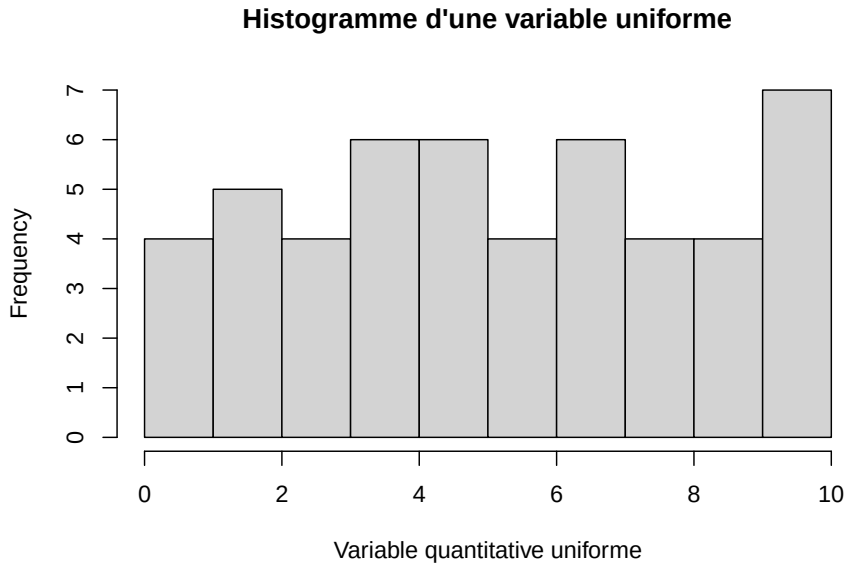


2.2.2 Matrice de scatter plots ~> `pairs()`

```
pairs(df_quanti)
```

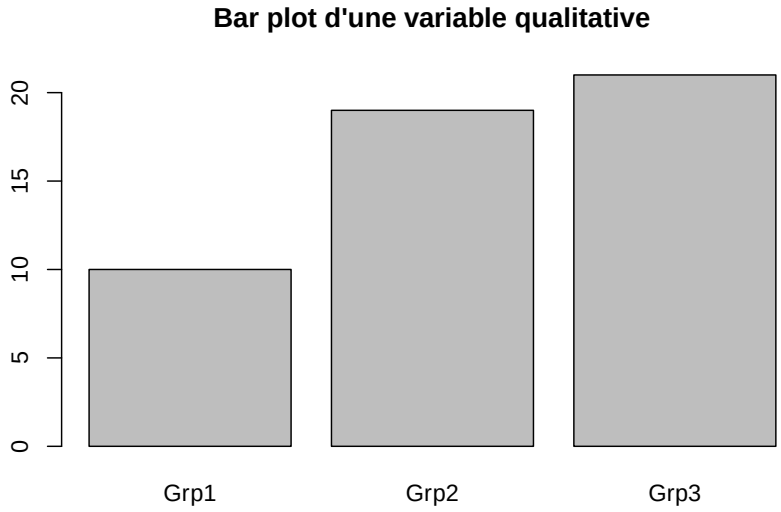


```r
hist(df$varQuanti, freq = TRUE ,
     main = "Histogramme d'une variable uniforme",
     xlab = "Variable quantitative uniforme")
```

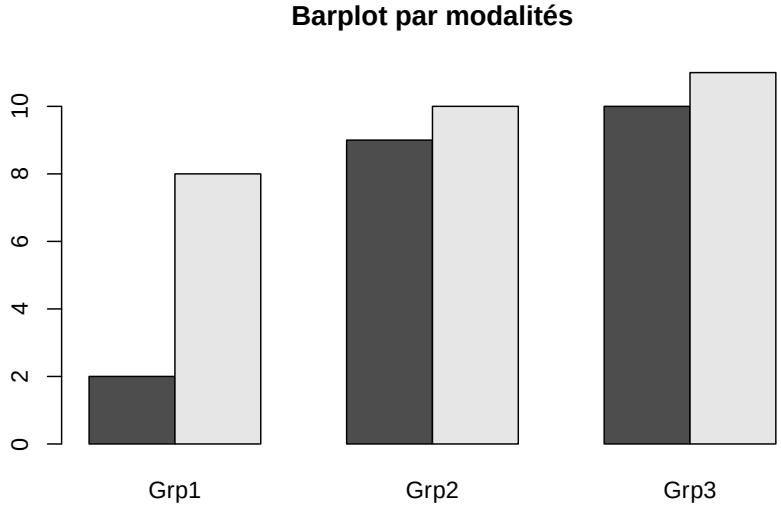



```r
# Une variable qualitative
barplot(table(df$varQuali2), main = "Bar plot d'une variable qualitative")
```

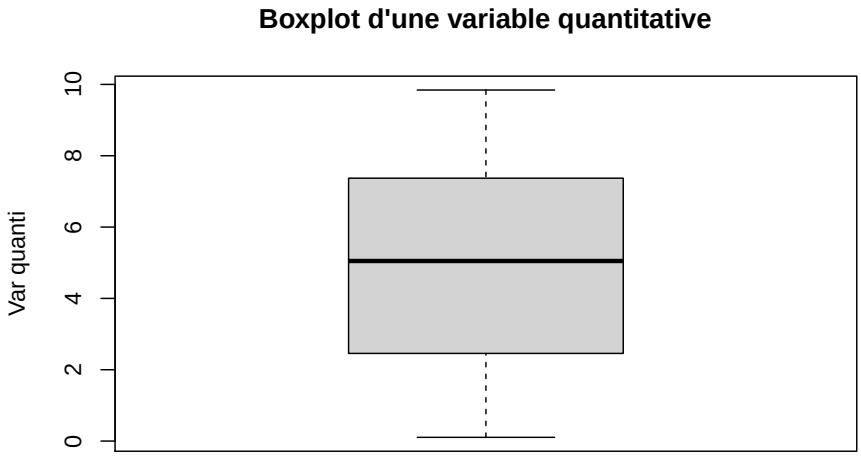
2.2.4 Bar plot ~> `barplot()`



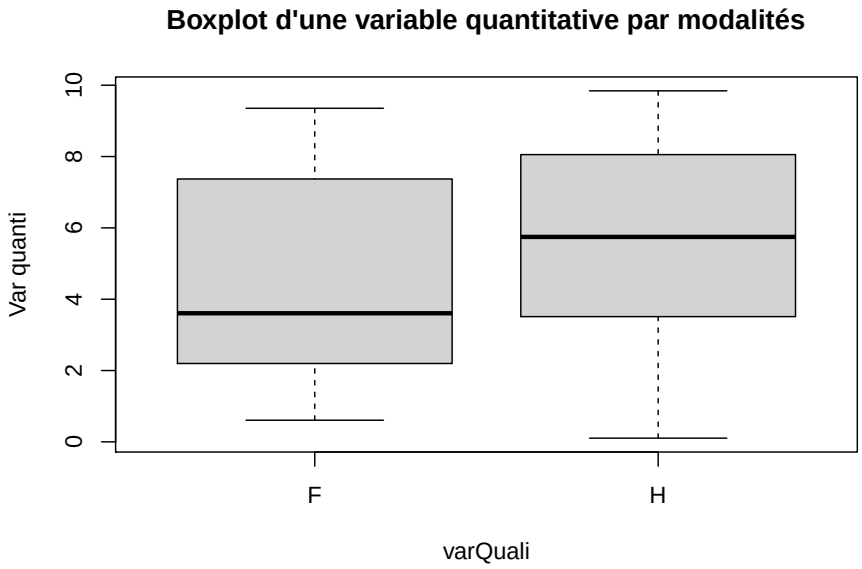

```r
# 2 variables qualitatives
barplot(table(df$varQuali1,df$varQuali2), beside = TRUE, legend = levels(df$varQuali1),
        main ="Barplot par modalités")
```



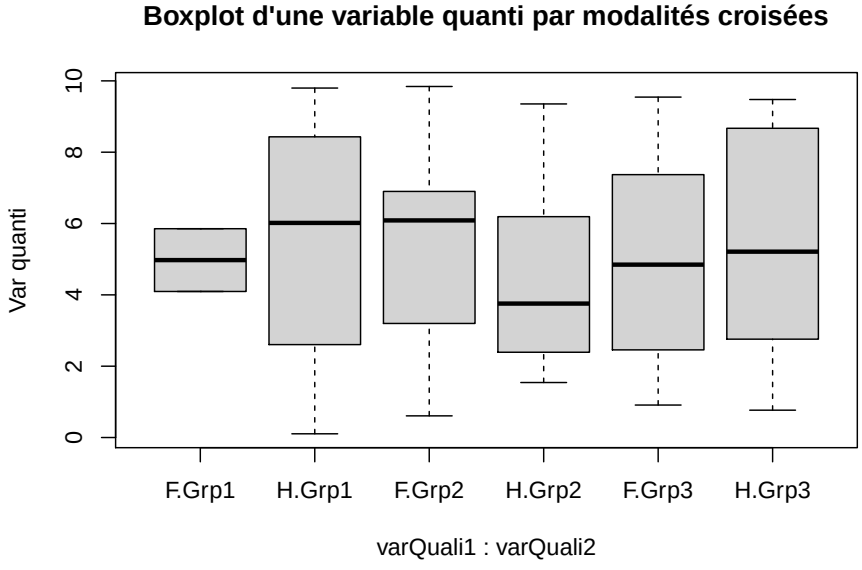
2.2.5 Box plot ~> `boxplot()`




```r
# Par modalités
boxplot(varQuanti ~ varQuali, data = df,
        main = "Boxplot d'une variable quantitative par modalités",  ylab = "Var quanti
```

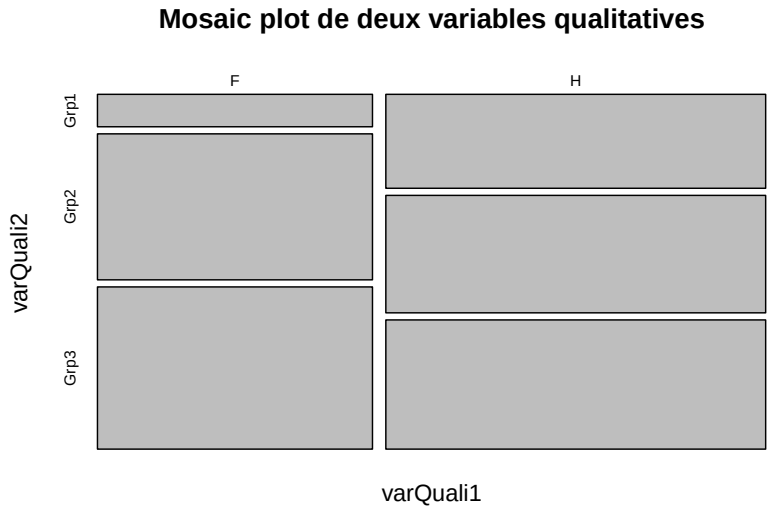



```r
# Par modalités croisées
boxplot(varQuanti ~ varQuali1*varQuali2, data = df,
        main = "Boxplot d'une variable quanti par modalités croisées",  ylab = "Var quanti")
```



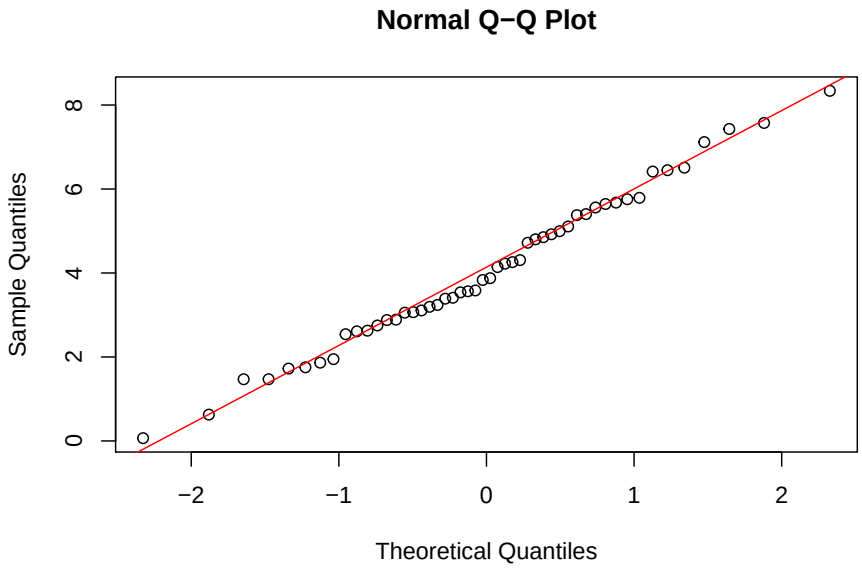
```r
mosaicplot(varQuali1 ~ varQuali2, data = df,
           main = "Mosaic plot de deux variables qualitatives")
```

2.2.6 Mosaic plot ~> `mosaicplot()`




```r
qqnorm(y = df$varQuanti1)
qqline(y = df$varQuanti1, col = "red")
```

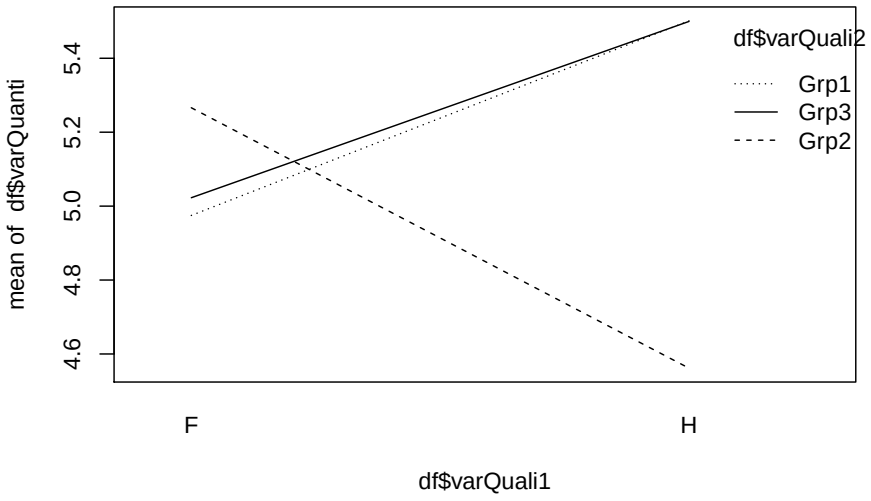
2.2.7 QQ-plot ~> `qqnorm()` + `qqlines()`



2.2.8 Graphes des interactions ~> `interaction.plot()`

Ce graphe présente les moyennes d'une variable quantitative pour toutes les combinaisons des modalités de 2 variables qualitatives.

```
interaction.plot(x.factor = df$varQuali1, trace.factor = df$varQuali2, response = df$varQuanti)
```


mean of df$varQuanti
4.6
4.8
5.0
5.2
5.4
F
H
df$varQuali1
df$varQuali2
Grp1
Grp3
Grp2

Chapter 3

Inference statistique sur une et deux variables

Cette section présente différentes fonctions R utiles pour réaliser de l'inférence statistique sur une ou deux variables.

3.1 Données Chocolat

Les fonctions de cette section sont illustrées sur base du jeu de données : `chocolat.xlsx` (disponible sur moodle). Ce data.frame de taille 20x2 reprend des mesures de la teneur en cacao de 20 plaquettes de chocolat de marque "Cabeau" et "Gallet".

Voici le code pour lire et préparer ces données

```
choco <- readxl::read_xlsx("data/chocolat.xlsx", col_names = TRUE)
choco$Marque <- as.factor(choco$Marque)
cabeau <- subset(choco, Marque =="Cabeau")
gallet <- subset(choco, Marque =="Gallet")
```

Voici quelques résumés descriptifs de ces données

Visualiser via un box-plot

```
boxplot(Cacao ~ Marque, data = choco, main = "Box plot des données Chocolat", ylab = "C
```

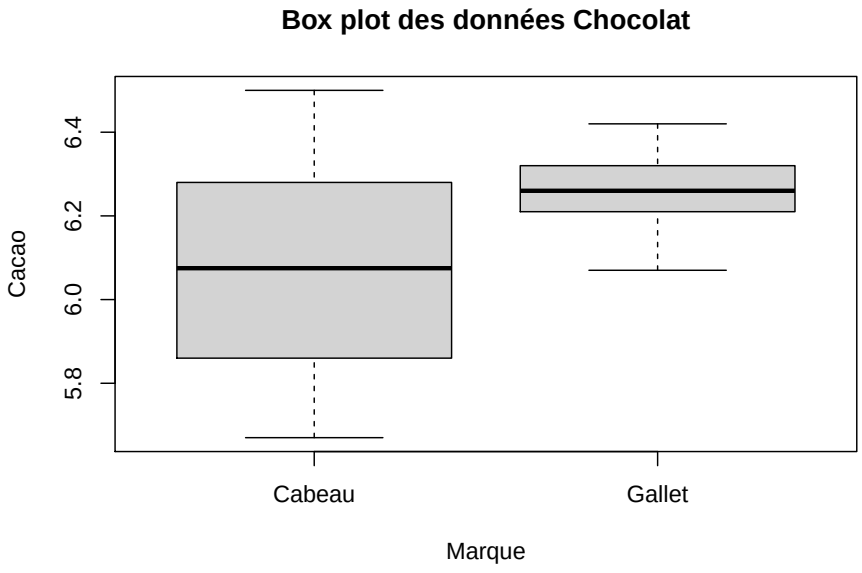


Tableau résumé par marque

```
N <- table(choco$Marque)
Moyenne<-tapply(choco$Cacao, choco$Marque, mean)
EcartType<-tapply(choco$Cacao, choco$Marque, sd)
cbind(N,Moyenne,EcartType)
```

	N	Moyenne	EcartType
Cabeau	10	6.063	0.2568
Gallet	10	6.258	0.108

3.2 Arguments utiles dans les fonctions d'inférence

Argument `alternative =` dans une fonction de test

- $H_0 : \theta = \theta_0$ contre $H_1 : \theta \neq \theta_0$ ~> `alternative = "two.sided"`
- $H_0 : \theta \geq \theta_0$ contre $H_1 : \theta < \theta_0$ ~> `alternative = "less"`
- $H_0 : \theta \leq \theta_0$ contre $H_1 : \theta > \theta_0$ ~> `alternative = "greater"`

Argument `conf.level =` dans une fonction de test

Par défaut, le niveau de confiance des intervalles de confiance est 95%. On peut le modifier via l'argument `conf.level = 0.9` pour, par exemple, un niveau de

confiance de 90%.

3.3 Tests sur une ou deux moyennes

3.3.1 Test t sur une moyenne

$$H_0 : \mu = 6 \text{ contre } H_1 : \mu \neq 6$$

```
t.test(choco$Cacao, mu = 6, alternative = "two.sided")
```

```
One Sample t-test

data:  choco$Cacao
t = 3.3192, df = 19, p-value = 0.003606
alternative hypothesis: true mean is not equal to 6
95 percent confidence interval:
 6.059293 6.261707
sample estimates:
mean of x
   6.1605
```

la fonction pander dans RMarkdown permet de mieux présenter les résultats mais ne présente pas tout. Ceci est aussi vrai pour les autres fonctions de test ou de modélisation.

```
res= t.test(choco$Cacao, mu = 6, alternative = "two.sided")
pander::pander(res)
```

Table 3.2: One Sample t-test: `choco$Cacao`

Test statistic	df	P value	Alternative hypothesis	mean of x
3.319	19	0.003606 * *	two.sided	6.16

3.3.2 Test t de comparaison de deux moyennes

$$H_0 : \mu_1 = \mu_2 \text{ contre } H_1 : \mu_1 \neq \mu_2$$

Echantillons indépendants et variances égales

```
t.test(choco$Cacao~choco$Marque, alternative = "two.sided", var.equal = TRUE)
```

Echantillons indépendants et variances différentes

```
t.test(choco$Cacao~choco$Marque, alternative = "two.sided")
```

```
Welch Two Sample t-test

data:  choco$Cacao by choco$Marque
t = -2.2137, df = 12.087, p-value = 0.04682
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.386775949 -0.003224051
sample estimates:
mean in group Cabeau mean in group Gallet
               6.063                6.258
```

Données pairées (non applicable aux données chocolat)

```
# Observations pairées
t.test(y~x, alternative = "two.sided", paired = TRUE)
```

3.4 Tests sur une ou plusieurs variances

3.4.1 Test χ^2 sur une variance

$$H_0 : \sigma^2 = 0.04 \text{ contre } H_1 : \sigma^2 \neq 0.04$$

```
EnvStats::varTest(x = cabeau$Cacao, sigma.squared = 0.04, alternative = "two.sided")
```

```
Chi-Squared Test on Variance

data:  cabeau$Cacao
Chi-Squared = 14.835, df = 9, p-value = 0.1911
alternative hypothesis: true variance is not equal to 0.04
95 percent confidence interval:
 0.03119472 0.21974978
sample estimates:
  variance
0.06593444
```

3.4.2 Test F de comparaison de 2 variances

$$H_0 : \sigma_1^2 = \sigma_2^2 \text{ contre } H_1 : \sigma_1^2 \neq \sigma_2^2$$

```
var.test(choco$Cacao~choco$Marque, ratio = 1, alternative = "two.sided")
```

```
F test to compare two variances

data:  choco$Cacao by choco$Marque
F = 5.6537, num df = 9, denom df = 9, p-value = 0.0166
```

```
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
  1.404294 22.761673
sample estimates:
ratio of variances
          5.653678
```

3.4.3 Test de Levene d'homogénéité de variances

Ce test permet de tester si 2 ou plus de 2 variances sont égales.

$H_0 : \sigma_1^2 = \sigma_2^2 = ... = \sigma_k^2$ contre H_1 : Au moins deux variances sont différentes

```
car::leveneTest(choco$Cacao, group = choco$Marque)
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  1  6.1977 0.0228 *
      18
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.5 Tests sur une ou plusieurs proportions

Pour les tests ci-dessous, utilisons le simple tableau de contingence pour 2 variables qualitatives X et Y de niveaux respectifs A1,A2,A3 et B1,B2 :

```
data=matrix(c(10,8,17,5,5,10),nrow=3,ncol=2)
dimnames(data)=list(c("A1","A2","A3"),c("B1","B2"))
pander(data)
```

	B1	B2
A1	10	5
A2	8	5
A3	17	10

3.5.1 Test χ^2 sur une proportion

Teste si la proportion lié au niveau `B1` de la variable Y est plus grande que 0.5. Ce test permet des hypothèses alternatives unilatérales >ou <. Il effectue aussi une correction de continuité.

$$H_0 : \pi_{B1} \leq 0.5 \text{ contre } H_1 : \pi_{B1} > 0.5$$

```r
nB1=sum(data[,1]) # Nb de données telles que Y=B1
n=sum(data) # Nb total de données
prop.test(nB1,n, p=0.5, alternative = "greater")
```

```
1-sample proportions test with continuity correction

data:  nB1 out of n, null probability 0.5
X-squared = 3.5636, df = 1, p-value = 0.02953
alternative hypothesis: true p is greater than 0.5
95 percent confidence interval:
 0.5164367 1.0000000
sample estimates:
        p
0.6363636
```

3.5.2 Test d'ajustement χ^2

On teste pour la variable X si la proportion d'individus dans chaque niveau est identique

$$H_0 : \pi_{A1} = \pi_{A2} = \pi_{A3} \text{ contre } H_1 : \text{ non } H_0$$

```r
dataY=apply(data,1,sum)
chisq.test(dataY,p=c(1/3,1/3,1/3))
```

```
Chi-squared test for given probabilities

data:  dataY
X-squared = 6.2545, df = 2, p-value = 0.04384
```

3.6 Tests de corrélation / indépendance

Test sur un coefficient de corrélation de Pearson

```r
cor.test(x = .., y = .., alternative = "two.sided") #x,y quantitatives
```

Test χ^2 d'indépendance

```r
chisq.test(data)
```

Tests de corrélation 2 à 2

```r
res <- Hmisc::rcorr(df_quanti)
                # df_quanti = data frame composé uniquement de variables quantitatives
```

~> `res$r` = matrice de corrélation ~> `res$p` = matrice des p-valeurs associées

3.7 Intervalles de confiance

Il y a 2 possibilités pour obtenir un intervalle de confiance dans R:

1) Calcul à la main Si on dispose de formules, nous pouvons calculer à la main les bornes inférieures et supérieures d'IC. Pour cela on a besoin d'estimations des paramètres et de quantiles de lois de probabilité bien connues.

Les quantiles sont obtenus via les fonctions suivantes:

- **Normale** ~> `qnorm(p = .., mean = .., sd = ..)`
- **Student** ~> `qt(p = .., df = ..)`
- **Chi_Carré** ~> `qchisq(p = .., df = ..)`

Un intervalle de confiance à 95% sur une moyenne des données chocolat de la société cabeau sera par exemple obtenu comme suit:

```
n=length(cabeau$Cacao)
mean(cabeau$Cacao)+c(-1,1)*qt(0.975,n-1)*sd(cabeau$Cacao)/sqrt(n)
```

```
[1] 5.879313 6.246687
```

2) Utiliser les fonctions de test Comme nous pouvons voir plus haut, certains tests calculent automatiquement les intervalles de confiance 95%. Il suffit donc de voir quel test donne l'IC recherché et d'utiliser des hypothèses nulles quelconques:

- **Moyenne** ~> test t sur 1 échantillon `t.test()`
- **Différence de moyennes** ~> test t sur 2 échantillons `t.test()`
- **Variance** ~> test Chi_Carré sur la variance `EnvStats::varTest()`
- **Ratio de variances** ~> test F de comparaison de variances `var.test()`
- **Proportion** ~> test de proportion à 1 échantillon `prop.test()`
- **Odds ratio** ~> test exact de Fisher `fisher.test()`
- **Corrélation** ~> test sur le coefficient de corrélation de Pearson `cor.test()`

Pour ces tests on peut ajouter l'argument `conf.level = ..` dans la fonction pour choisir soi-même le niveau de confiance de l'intervalle - **par défaut 95%**.

Extraire l'IC

```
res <- t.test(cabeau$Cacao, mu = 6, alternative = "two.sided")
ic <- data.frame(Estimate = mean(cabeau$Cacao),
                 Lwr = res$conf.int[1],
                 Upr = res$conf.int[2])
pander::pander(ic)
```

Estimate	Lwr	Upr
6.063	5.879	6.247

Chapter 4

Modeles lineaires

Objectif

Expliquer une variable réponse quantitative de distribution normale comme une fonction linéaire de variables explicatives quantitatives et/ou qualitatives + de termes d'ordres plus élevés.

La fonction de modélisation principale utilisée ici est le fonction `lm()` complétée par une série de fonctions génériques applicables aux résultats de `lm`. Des fonctions émanant des librairies `visreg` et `emmeans` sont aussi proposées pour représenter les résultats des modèles et faire de l'inférence sur des combinaisons linéaires des paramètres.

4.1 Modélisation LM

La fonction `lm` permet par exemple d'estimer les modèles de base suivants qui font tous partie de la famille des modèles linéaires.

```
# Régression linéaire simple - X quantitative
mod <- lm( Y ~ X, data = ..)
# Régression linéaire multiple - X1,X2,X3 quantitatives
mod <- lm( Y ~ X1 + X2 + X3, data = ..)
# Régression quadratique - X quantitative
mod <- lm( Y ~ X + I(X^2), data = ..)
# ANOVA 1 - X qualitative
mod <- lm( Y ~  X, data = ..)
# ANOVA 2 croisée - X1,X2 qualitatives
mod <- lm( Y ~ X1 * X2, data = ..)
# Modèle d'analyse de covariance - X1 quantitative, X2 qualitative
mod <- lm( Y ~ X1 * X2, data = ..)
```

Formules dans R

Le premier argument de la fonction `lm()` est un objet de type formule qui fournit à R la liste des effets à inclure dans le modèle linéaire.

- `Y ~ 1` = modéliser Y en fonction de l'intercept uniquement.
- `Y ~ X` = modéliser Y en fonction de X avec intercept.
- `Y ~ -1 + X` = modéliser Y en fonction de X sans intercept.
- `Y ~ X1 + X2` = modéliser Y en fonction de X1 et X2 avec intercept.
- `Y ~ X1 * X2` = modéliser Y en fonction de X1, X2 et de l'interaction X1:X2 avec intercept (équivalent au modèle `Y ~ X1 + X2 + X1:X2`) .
- `Y ~ I(f(X))` = modéliser Y en fonction de f(X), une fonction quelconque de X, avec intercept.

Tout cela peut bien entendu être mixé pour créer un modèle bien spécifique. Par exemple pour une régression quadratique sans intercept on utilisera `Y ~ -1 + X + I(X^2)`.

Note La réponse Y peut également être transformée par exemple en logarithme. Il faut utiliser dans ce cas directement `log(Y)` dans la formule sans l'entourer par la fonction `I()`.

4.2 Visualiser les données avant la modélisation LM

Voici un petit rappel du type de graphiques utiles pour représenter graphiquement des données avant modélisation.

Régression linéaire simple / quadratique ~> scatter plot Y ~ X.

Régression linéaire multiple ~> matrice de scatter plots Y ~ X1, Y ~ X2.

ANOVA 1 ~> boxplot Y ~ X.

ANOVA 2 ~> boxplot Y ~ X1*X2.

Modèle linéaire général ~> scatter plot Y ~ X1 par modalités de X2 où X2 est quantitative ou qualitative.

4.3 Fonctions utiles sur un objet de type `mod <- lm(..)`

Voici une liste non exhaustive de fonctions qui peuvent être appliquées à l'objet fourni par `lm(..)` pour obtenir des résultats dérivés du modèle estimé. L'utilisation de certaines de ces fonctions est illustrée ci-dessous.

Résumé du modèle	`summary(mod)`

Mise à jour du modèle et Signification des coefficients liés	`drop1(mod, ..)` + `update(mod, ..)`
Coefficients + Intervalles de Confiance	`coef(mod)` + `confint(mod)` ou `car::Confint(mod)`
Résidus / Réponses prédites	`residuals(mod)`/`fitted(mod)`
Matrice de Variance-Covariance param.	`vcov(mod)`
Leverages	`hatvalues(mod)`
Matrice X de la régression	`mod.matrix(mod)`
Analyse de la variance (voir remarques plus bas)	`anova(mod)` TYPE I `car::Anova(mod, type = 2)` TYPE II `car::Anova(mod, type = 3)` TYPE III
Graphes des diagnostics	`plot(mod)`
Test d'homogénéité de la variance	`car::leveneTest(mod)`
Visualiser le modèle	`visreg::visreg(mod, ..)` `abline(mod, ...)` régression linéaire simple `emmeans::emmip(mod, ..)` modèle linéaire général
Prédictions	`predict(mod, ..)`
Graphe des moyennes	`sjPlot::plot_model(mod, type = "eff)`
Comparaisons de moyennes	`emmeans::lsmeans(mod,..)`
Contrasts linéaires	`lsm <- emmeans::lsmeans(mod,..)` + `emmeans::contrast(lsm, ..)`
Comparaisons des pentes	`emmeans::emtrends(mod,..)`

4.4 Données du chapitre LM

`calibration.xlsx`	Estimer une droite de calibration pour relier la surface en dessous du pic d'un chromatogramme en fonction de la concentration (1 variable quantitative)
`Volume_arbres.xlsx`	Modélisation du volume d'un tronc d'arbre en fonction de sa hauteur et de son diamètre (2 variables quantitatives)
`diet_pourR.xlsx`	Estimer un modèle d'ANOVA 1 pour expliquer la perte de poids d'une souris en fonction d'un régime alimentaire suivi (1 var. qualit. à 4 niv.)
`mais.xlsx`	Estimer un modèle d'ANOVA 2 pour expliquer le rendement d'une culture de maïs en fct du type d'engrais et de graine (2 var. qual. à 3 niv.)
`Digestion.xlsx`	Estimer un modèle linéaire général pour expliquer la digestion in-vitro d'un amidon en fonction de 3 variables : le type d'amidon (le standard) (var. qualit. à 3 niv.), le PH et la concentration en Enzyme (2 var. quant.)

4.5 Utiliser `summary()` sur lm

Exemple pour les données `Volume_arbres`

```
# Modèle complet pour les données Volume_arbres
mod <- lm(lnVolume ~ lnDiam * lnHauteur, data = Volume_arbres)
summary(mod)
```

```
Call:
lm(formula = lnVolume ~ lnDiam * lnHauteur, data = Volume_arbres)

Residuals:
      Min        1Q    Median        3Q       Max
-0.165941 -0.048613  0.006384  0.062204  0.132295

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      -3.6869     7.6996  -0.479    0.636
lnDiam            0.7942     3.0910   0.257    0.799
lnHauteur         0.4377     1.7788   0.246    0.808
lnDiam:lnHauteur  0.2740     0.7124   0.385    0.704

Residual standard error: 0.08265 on 27 degrees of freedom
```

```
Multiple R-squared:  0.9778,Adjusted R-squared:  0.9753
F-statistic: 396.4 on 3 and 27 DF,  p-value: < 2.2e-16
```

4.6 Utiliser drop1() / update() sur lm

On a le choix entre un test Chi_Carré, un test F ou une comparaison sur base du critère AIC.

```
drop1(mod, test = "Chisq")  # Chi_Carré
drop1(mod, test = "F")      # F
drop1(mod)                  # AIC
```

Options pour le test F

La fonction drop1() va toujours commencer par tester les effets d'interaction avant le reste. Un effet principal n'est jamais retiré avant un effet d'ordre plus élevé qui lui est lié.

```
res <- drop1(mod, test = "F")
pander(res)
```

Table 4.3: Single term deletions

	Df	Sum of Sq	RSS	AIC	F value	Pr(>F)
	NA	NA	0.1845	-150.9	NA	NA
lnDiam:lnHauteur	1	0.00101	0.1855	-152.7	0.1479	0.7035

Mise à jour du modèle

Ici l'interaction n'est pas significative donc on décide de l'enlever de notre modèle.

```
mod <- update(mod, .~. - lnDiam:lnHauteur)
```

Vérification

```
summary(mod)
```

```
Call:
lm(formula = lnVolume ~ lnDiam + lnHauteur, data = Volume_arbres)

Residuals:
      Min        1Q    Median        3Q       Max
-0.168561 -0.048488  0.002431  0.063637  0.129223

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.63162    0.79979  -8.292 5.06e-09 ***
lnDiam       1.98265    0.07501  26.432  < 2e-16 ***
lnHauteur    1.11712    0.20444   5.464 7.81e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.08139 on 28 degrees of freedom
Multiple R-squared:  0.9777,Adjusted R-squared:  0.9761
F-statistic: 613.2 on 2 and 28 DF,  p-value: < 2.2e-16
```

Maintenant tous les paramètres sont significatifs

4.7 Utiliser `anova()` / `Anova()` sur lm

4.7.1 Plan balancé

Rmq Par "balancé" on entend un nombre égal d'observations pour chaque niveau de nos facteurs

Utiliser le **modèle classique** pour calculer les **sommes de carrés de TYPE I** (via `anova()`)

```
anova(mod)
```

4.7.2 Plan non balancé

Ajuster un **modèle différent avec le codage "contr.sum"**

```
mod2 <- lm(Production ~ Graines + Engrais + Graines:Engrais,
              contrasts = list(Graines=c("contr.sum"), Engrais =c("contr.sum")),
              data = mais)
```

Utiliser ce modèle pour calculer les **sommes des carrés de TYPE III** (via `car::Anova()`)

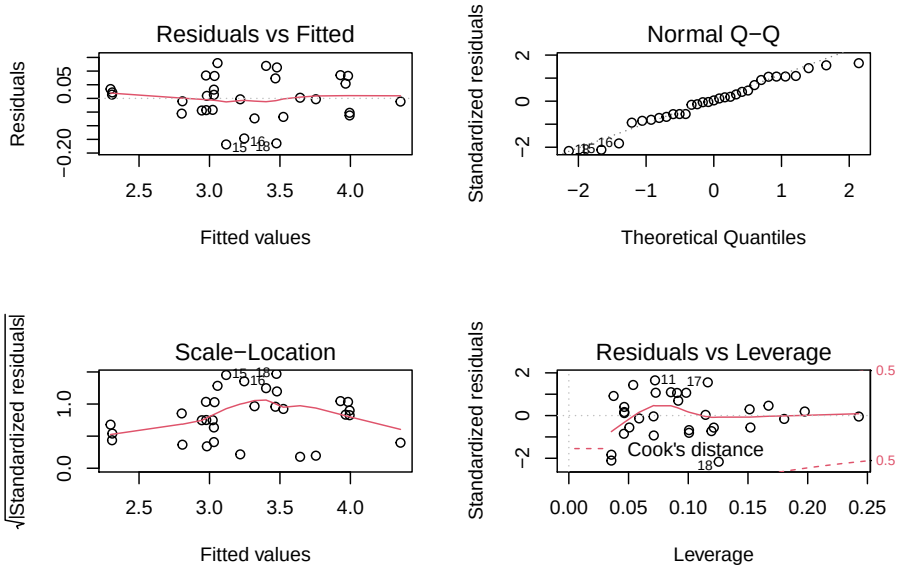
```
Anova(mod2, type = 3)
```

```
Anova Table (Type III tests)

Response: Production
               Sum Sq Df   F value    Pr(>F)
(Intercept)    172284  1 40274.299 < 2.2e-16 ***
Graines           432  2    50.532 1.278e-05 ***
Engrais           628  2    73.403 2.676e-06 ***
Graines:Engrais   244  4    14.240 0.0006255 ***
Residuals          39  9
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.8 Utiliser `plot()`sur lm

```
par(mfrow=c(2,2))
plot(mod)
```

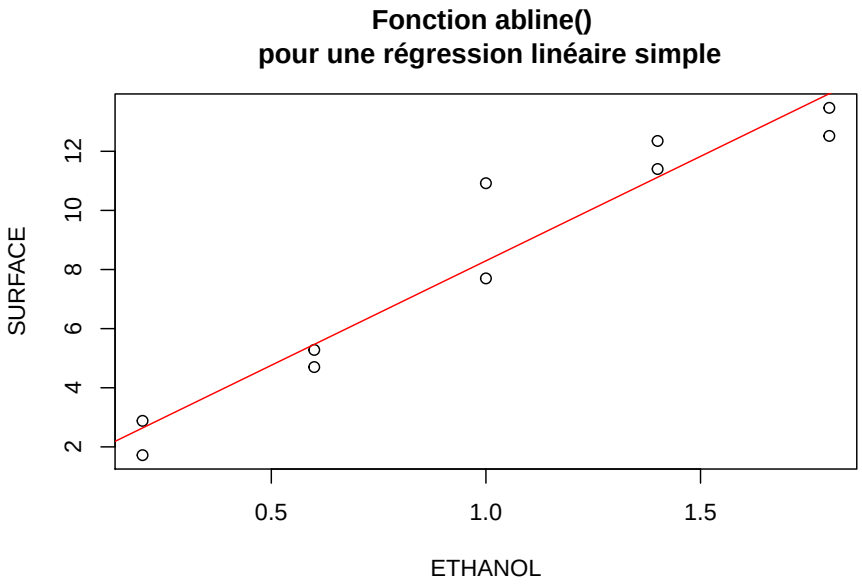



```
par(mfrow=c(1,1))
```

4.9 Utiliser `abline()` sur lm

Pour ajouter une droite de régression sur un simple scatter-plot (ou graphe X-Y). **Attention, ne fonctionne de cette façon que pour la régression linéaire simple**

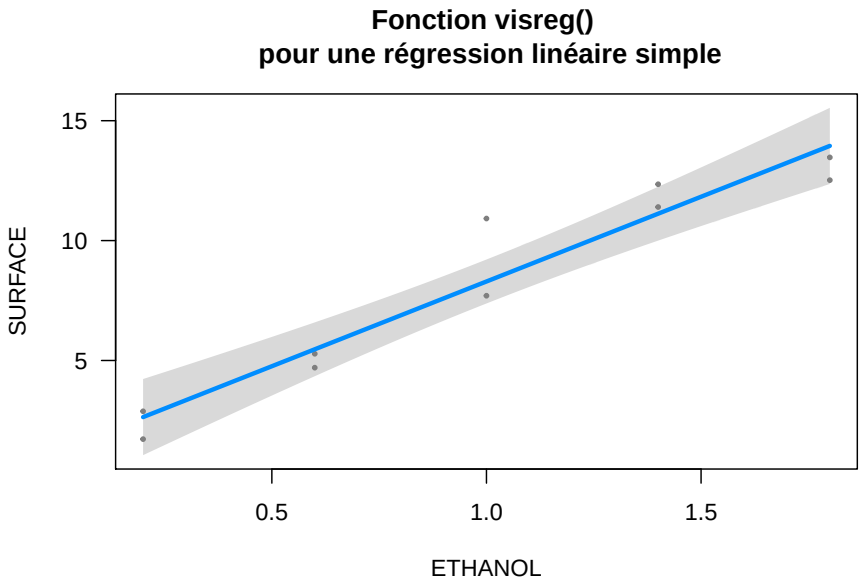
```
# Jeu de données calibration.xlsx
mod <- lm(SURFACE ~ ETHANOL, data = calibration)
plot(SURFACE ~ ETHANOL, data = calibration,
     main = "Fonction abline() \n pour une régression linéaire simple")
abline(mod, col = "red")
```



4.10 Utiliser `visreg()` sur lm

La fonction `visreg` est très utile pour représenter graphiquement les résultats d'une modélisation. Attention malgré tout : si le nombre de variables du modèle est plus élevé que le nombre de variables représentées, les points sur les graphiques ne sont pas les données de départ mais sont recalculées pour une valeur commune des variables qui ne sont pas sur le graphique (résidus partiels), il est donc conseillé de ne pas les dessiner. Les courbes/moyennes dessinées sont également dépendantes de ces valeurs.

```
# Jeu de données calibration1.xlsx disponible sur moodle
mod <- lm(SURFACE ~ ETHANOL, data = calibration)
visreg(mod, main = "Fonction visreg() \n pour une régression linéaire simple")
```

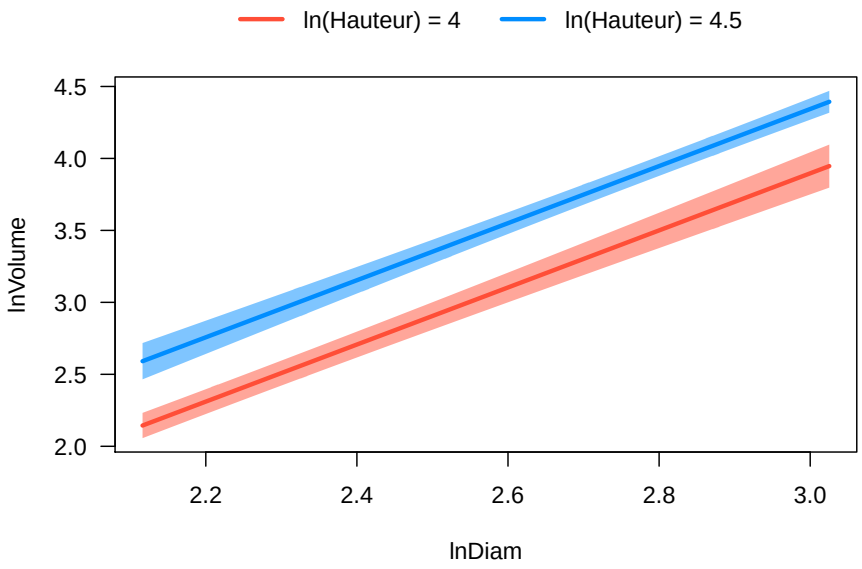


Rmq utilisation similaire pour une régression quadratique.

```
# Jeu de données Volume_arbres.xlsx
mod <- lm(lnVolume ~ lnDiam + lnHauteur, data = Volume_arbres)
```

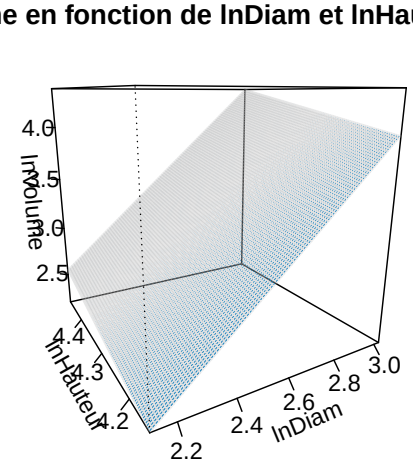
Volume en fonction du Diamètre pour deux niveau de la Hauteur

```
visreg(mod, xvar = "lnDiam", by = "lnHauteur", breaks = c(4.1,4.5),
       overlay = TRUE, band = TRUE, strip.names = c("ln(Hauteur) = 4", "ln(Hauteur) = 4.5"),parti
```

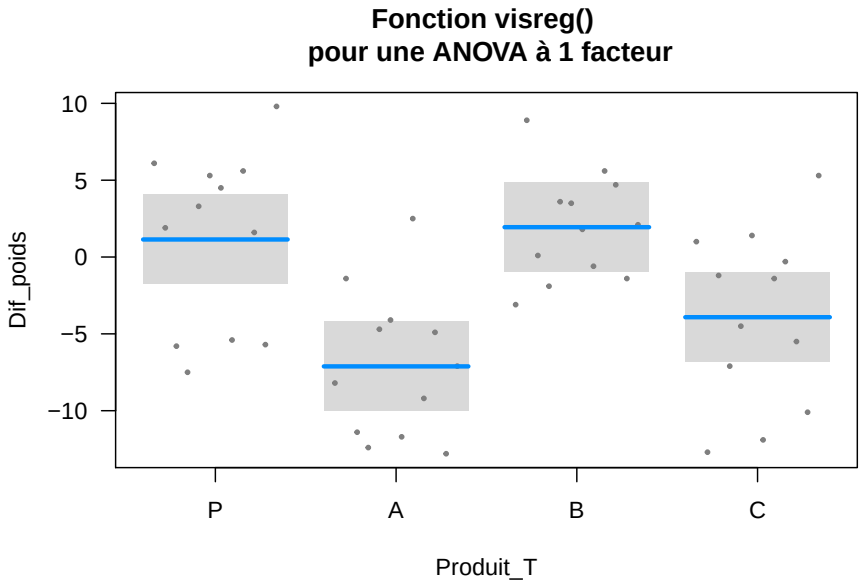


Régression linéaire multiple en 3 dimensions avec visreg2d

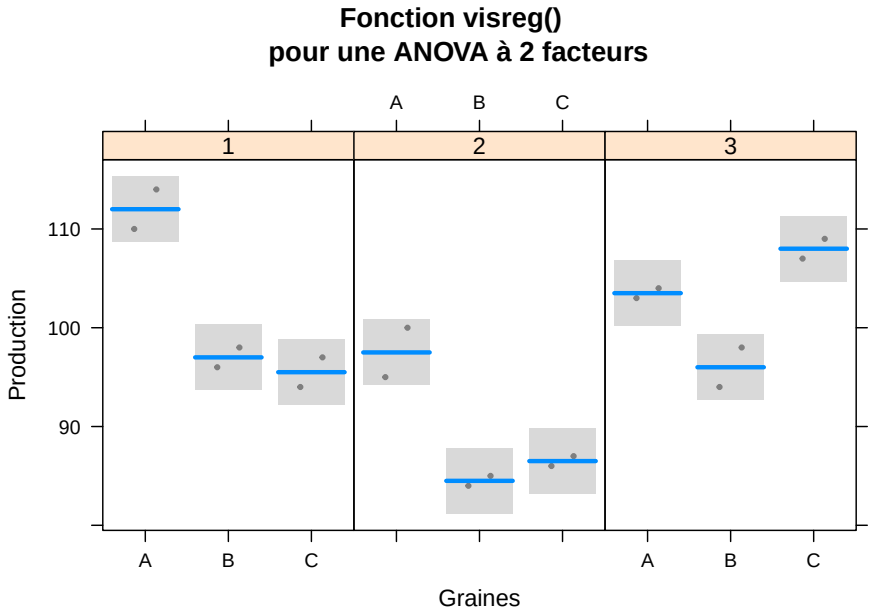
```
visreg2d(mod,"lnDiam", "lnHauteur", plot.type="persp",main="lnVolume en fonction de lnl
```



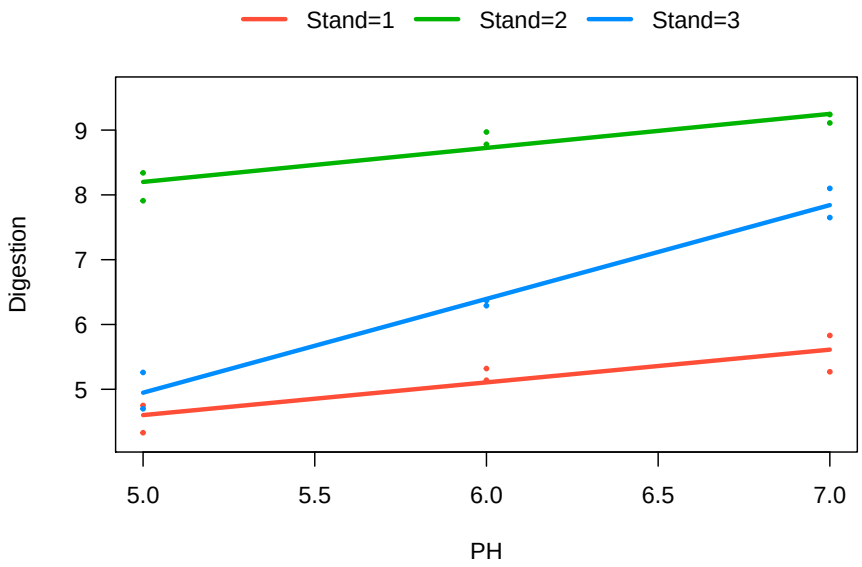
```r
# Jeu de données diet_pourR.xlsx
mod <- lm(Dif_poids ~ Produit_T, data = diet_pourR)
visreg(mod, main = "Fonction visreg() \n pour une ANOVA à 1 facteur")
```




```r
# Jeu de données mais.xlsx disponible sur moodle
mod <- lm(Production ~ Graines*Engrais, data = mais)
visreg(mod, "Graines", by = "Engrais",
       main = "Fonction visreg() \n pour une ANOVA à 2 facteurs")
```




```
# Jeu de données Digestion.xlsx pour CCEnz=0.9 uniquement
digestion09=subset(digestion,subset=(CcEnz==0.9))
mod <- lm(Digestion ~ PH*Standard, data = digestion09)
visreg(mod, xvar = "PH", by = "Standard", overlay=TRUE, band = FALSE,
       strip.names=c("Stand=1", "Stand=2", "Stand=3"))
```

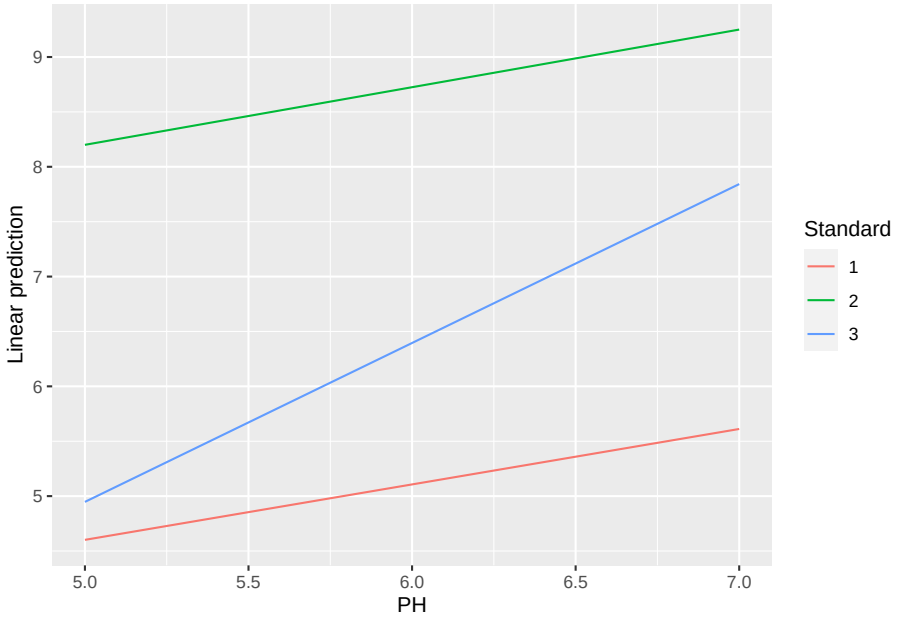


4.11 Utiliser `emmip()` sur lm

Le fonction `emmip()` du package emmeans permet de représenter des graphes d'interaction sur base d'un modèle ajusté.

Pratique pour un modèle linéaire général

```
# Jeu de données Digestion.xlsx disponible sur moodle
mod <- lm(Digestion ~ PH*Standard, data = digestion09)
emmip(mod, Standard ~ PH, cov.reduce = range)
```



4.12 Utiliser `predict()` sur lm

`Predict` calcule la réponse prédite par la modèle pour de nouvelles valeurs des variables explicatives ainsi que des intervalles de confiance ou de prédiction liés.

```
# Jeu de données Digestion.xlsx disponible sur moodle
mod <- lm(Digestion ~ PH*Standard, data = digestion09)
```

Obtenir les prédictions pour **(PH, Standard) = (5,“2”) et (6,“3”)**

```
xnew <- data.frame("PH" = c(5,6),
                   "Standard" = c("2", "3"))
pred <- predict(mod, newdata = xnew, interval = "prediction")
res <- data.frame(xnew, "Y_Fit" = pred[,1], "Lwr" = pred[,2], "Upr" = pred[,3])
pander(res)
```

PH	Standard	Y_Fit	Lwr	Upr
5	2	8.2	7.544	8.856
6	3	6.395	5.8	6.99

4.13 Utiliser `lsmeans()` et le package `emmeans` sur `lm`

`lsmeans` du package `emmeans` est utile pour calculer des combinaisons linéaires de paramètres de modèles linéaires et les résultats d'inférence liés (tests et IC). Cette fonction est spécialement utile pour les modèles contenant des variables qualitatives/catégorielles : estimation de moyennes marginales, comparaisons 2 à 2, constrates etc...

```
# ANOVA 2 sur les données mais.xlsx
mod <- lm(Production ~ Graines*Engrais, data = mais)
```

4.13.1 Table des moyennes

Pour les modalités de Graines

```
emmeans::lsmeans(mod, ~ Graines)
```

```
NOTE: Results may be misleading due to involvement in interactions
```

Graines	lsmean	SE	df	lower.CL	upper.CL
A	104.3	0.8444	9	102.4	106.2
B	92.5	0.8444	9	90.59	94.41
C	96.67	0.8444	9	94.76	98.58

Pour modalités croisées (interaction)

```
emmeans::lsmeans(mod, ~ Graines*Engrais)
```

Graines	Engrais	lsmean	SE	df	lower.CL	upper.CL
A	1	112	1.462	9	108.7	115.3
B	1	97	1.462	9	93.69	100.3
C	1	95.5	1.462	9	92.19	98.81
A	2	97.5	1.462	9	94.19	100.8
B	2	84.5	1.462	9	81.19	87.81
C	2	86.5	1.462	9	83.19	89.81
A	3	103.5	1.462	9	100.2	106.8
B	3	96	1.462	9	92.69	99.31
C	3	108	1.462	9	104.7	111.3

4.13.2 Comparaison de moyennes 2 à 2

```
lsmeans(mod, pairwise ~ Graines)$contrasts
```

```
lsm <- lsmeans(mod, pairwise ~ Graines*Engrais)
plot(lsm)
```

```
# Jeu de données mais.xlsx disponible sur moodle
lsm <- lsmeans(mod, pairwise ~ Graines*Engrais)
hyp <- list("C1 = A2" = c(0,0,1,-1,0,0,0,0,0),
            "A = C" = c(1/3,0,-1/3,1/3,1/3,0,-1/3,1/3,0,-1/3))
contrast(lsm$lsmeans, hyp)
```

```
 contrast estimate   SE df t.ratio p.value
 C1 = A2     -2.00 2.07  9 -0.967  0.3588
 A = C        7.67 1.19  9  6.420  0.0001
```

Rmq pour connaitre l'ordre des coefficients il suffit de faire `plot(lsm)` et lire de bas en haut les modalités listées sur l'axe y.

4.13.5 Utiliser `emtrends()` sur lm

Pour faire de l'inférence sur des pentes en analyse de convariance.

```
mod <- lm(Digestion ~ PH*Standard, data = digestion09)
visreg(mod, xvar = "PH", by = "Standard", overlay=TRUE, band = FALSE,
       strip.names=c("Stand=1", "Stand=2", "Stand=3"))
```

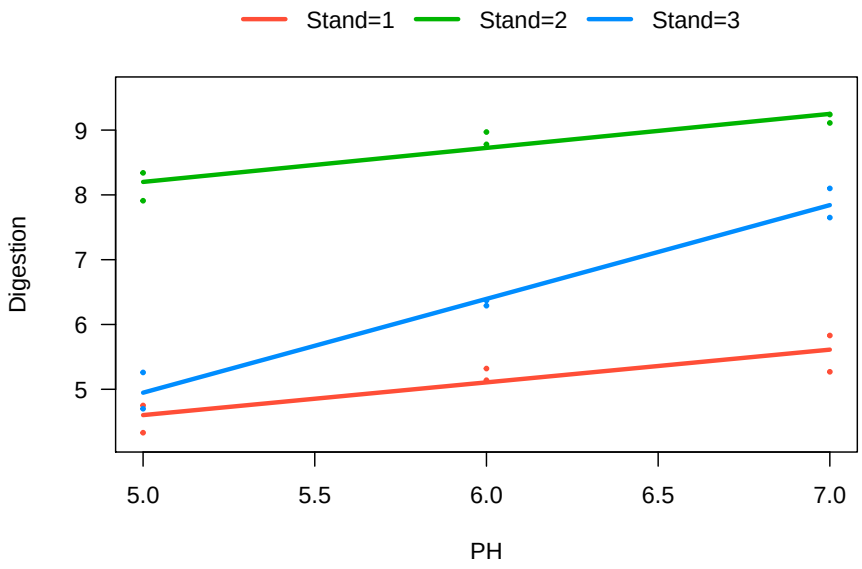


Table des pentes

```
emtrends(mod, pairwise ~ Standard, var = "PH")$emtrends
```

```
 Standard PH.trend    SE df lower.CL upper.CL
 1           0.505 0.126 12     0.23     0.78
 2           0.525 0.126 12     0.25     0.80
 3           1.448 0.126 12     1.17     1.72

Confidence level used: 0.95
```

Comparaison des pentes 2 à 2

```
emtrends(mod, pairwise ~ Standard, var = "PH")$contrasts
```

```
 contrast estimate    SE df t.ratio p.value
 1 - 2      -0.020 0.179 12 -0.112  0.9931
 1 - 3      -0.942 0.179 12 -5.271  0.0005
 2 - 3      -0.922 0.179 12 -5.160  0.0006

P value adjustment: tukey method for comparing a family of 3 estimates
```

Chapter 5

Regression logistique

Objectif

Expliquer une variable réponse binaire en fonction de variables explicatives quantitatives et / ou qualitatives + effets d'ordres plus élevés (interactions...).

La fonction de modélisation principale utilisée ici est la fonction `glm()` qui permet d'estimer des modèles linéaires généralisés très généraux. Elle s'utilise de façon très semblable à la fonction `lm()` et , comme pour `lm()`, de multiples fonctions générique s'appliquent aux résulats de `glm()`.

5.1 Modélisation GLM

Régression logistique simple X variable explicative qualitative ou quantitative

```
# si les réponses sous la forme d'un tableau nombre succès / échec / total
mod <- glm( cbind(succes,echec) ~ X, data = .., family = binomial)

# si les réponses sont sous la forme d'un vecteur de 1 et 0 : 1 ligne = 1 individu
mod <- glm( Y ~ X, data = .., family = binomial)
```

Régression logistique multiple

Quand il y a plusieurs variables explicatives avec ou sans effets d'ordres plus élevés (interactions...), l'équation du modèle s'écrit comme dans le cas des modèles linéaires `lm()`.

5.2 Visualiser les données avant modélisation GLM

Si X quantitative

- Plot p ~ X où p est la proportion de Y = 1 (=succès).
- Table de fréquence.

Si X qualitative

- Barplot Y ~ X
- Table de fréquence.0

5.3 Fonctions utiles sur un objet de type `mod <- glm()`

La plupart des fonction disponible sur les modèles de type `lm()` s'appliquent aussi aux modèles `glm()`.
Pour rappel :

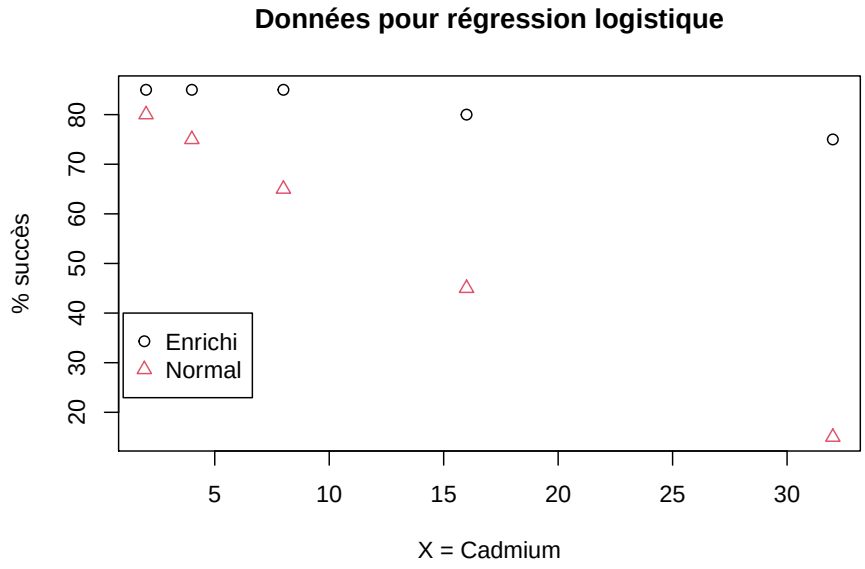
Résumé du modèle	`summary(mod)`
Mise à jour du modèle	`drop1(mod, ..)` + `update(mod, ..)`
Coefficients + IC	`coef(mod)` + `confint(mod)` ou `car::Confint(mod)`
Résidus / Valeurs ajustées	`residuals(mod, ..)` / `fitted(mod)`
Matrice Var-Cov des paramètres	`vcov(mod)`
Matrice X du modèle	`mod.matrix(mod)`
Visualiser la modélisation	`visreg::visreg(mod, ..)`
Prédictions	`predict(mod, ..)`

5.4 Données du chapitre GLM

`omega3Cadmium.txt`	Etude de l'interaction entre nutriments bénéfiques pour la santé Omega3 (1 var. quali. à 2 niv.) et contaminants a priori nocifs Cadmium (1 var. quanti.) sur la survie de cellules.

Ces données ne sont pas des données individuelles mais sont agrégées par groupe.

```
# Représentation graphique des données
PS=100*data[,"Survie_Oui"]/(data[,"Survie_Oui"]+data[,"Survie_Non"])
OM=as.numeric(data[,"Omega3"])

plot(data[,"Cadmium"],PS,col=OM,pch=OM,xlab="X = Cadmium",ylab="% succès",main="Données pour régr
legend(1,40,legend=levels(data[,"Omega3"]),col=1:2,pch=1:2)
```




```
# Régression logistique multiple avec interaction
mod <- glm(cbind(Survie_Oui, Survie_Non) ~ Cadmium*Omega3, data,  family = binomial)
```

5.5 Utiliser summary() sur glm

```
summary(mod)
```

```
Call:
glm(formula = cbind(Survie_Oui, Survie_Non) ~ Cadmium * Omega3,
    family = binomial, data = data)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.12008  -0.10088   0.01398   0.08504   0.16546
```

```
Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)           1.82160    0.41425   4.397  1.1e-05 ***
Cadmium              -0.02298    0.02267  -1.014   0.3107
Omega3Normal         -0.32059    0.54736  -0.586   0.5581
Cadmium:Omega3Normal -0.08021    0.03323  -2.413   0.0158 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 41.56439  on 9  degrees of freedom
Residual deviance:  0.10979  on 6  degrees of freedom
AIC: 38.758

Number of Fisher Scoring iterations: 4
```

5.6 Utiliser `drop1()`/`update()` sur glm

Utilisation similaire aux modèles linéaires `lm()`. La seule différence est dans les tests disponibles:

```r
drop1(mod, test = "Chisq")  # Chi_Carré
drop1(mod, test = "Rao")    # Rao
drop1(mod, test = "LRT")    # LRT
drop1(mod, test = "F")      # F
drop1(mod)                  # AIC
```

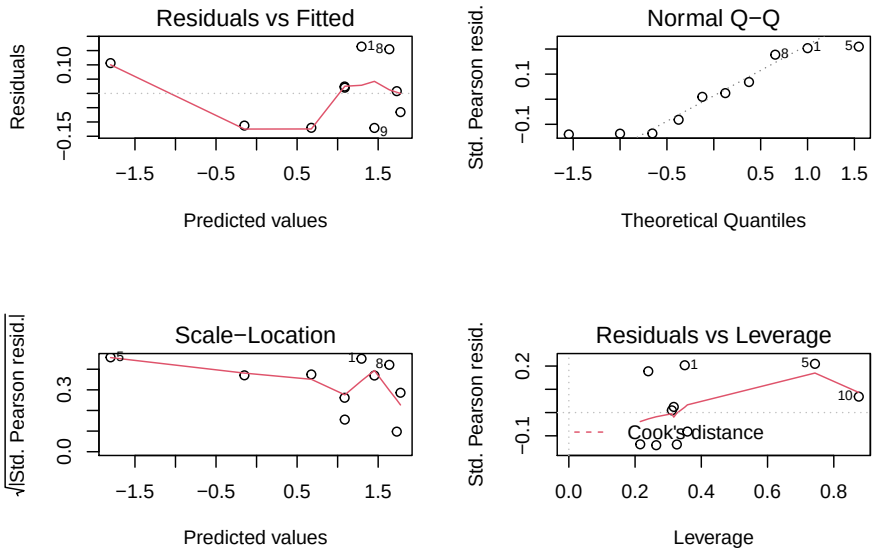
5.7 Utiliser `residuals()` sur glm

Pour obtenir les valeurs numériques des résidus

```r
residuals(mod, type = "deviance")      # Résidus de déviance
residuals(mod, type = "pearson")      # Résidus de Pearson
```

Graphe des résidus de pearson

```r
par(mfrow=c(2,2))
plot(mod)
```

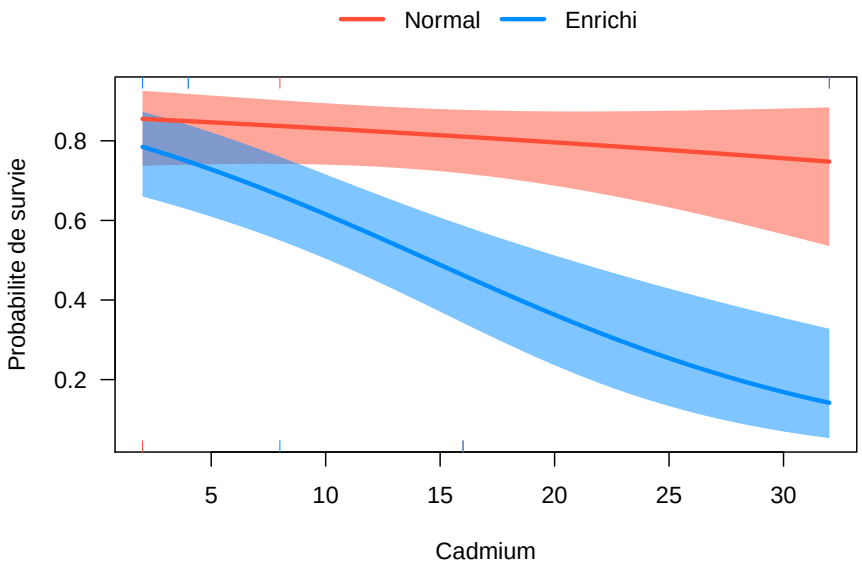



```
par(mfrow=c(1,1))
```

5.8 Utiliser `visreg()` sur glm

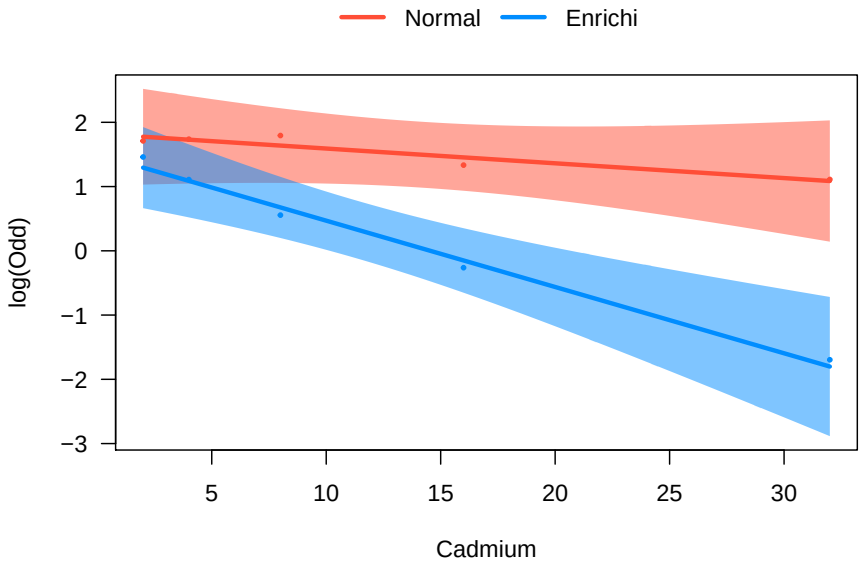
```
visreg(mod, scale = "response", xvar = "Cadmium", by = "Omega3", band = TRUE,
       ylab = "Probabilite de survie", strip.names=c("Normal", "Enrichi"), overlay= TRUE)
```

5.8.1 Probabilité de succès




```
visreg(mod, xvar = "Cadmium", band = TRUE, by = "Omega3",
       ylab = "log(Odd)", strip.names = c("Normal", "Enrichi"), overlay = TRUE)
```

5.8.2 Log(odds)



5.9 Utiliser `predict()` sur glm

Obtenir les prédictions pour **(Omega3, Cadmium) = ("Normal", 12) , ("Enrichi",12)**

```
xnew <- data.frame("Omega3" = c("Normal", "Enrichi"), "Cadmium" = c(12, 12))
pred <- predict(mod, newdata = xnew, type = "response", se.fit = TRUE)
res <- data.frame(xnew,
                  "Fit" = pred$fit,
                  "Lwr" = pred$fit - 1.96 * pred$se.fit,
                  "Upr" = pred$fit + 1.96 * pred$se.fit)
pander(res)
```

Omega3	Cadmium	Fit	Lwr	Upr
Normal	12	0.5653	0.4546	0.676
Enrichi	12	0.8243	0.7487	0.8999

Chapter 6

Analyse en composantes principales ACP

Objectif

~> Réduire le nombre de variables en transformant des variables corrélées en nouvelles variables décorrélées afin de mieux visualier les données. Les nouvelles variables, les composantes principales, sont des combinaisons linéaires des variables de départ.

Les packages privilégiés à utiliser pour faire de l'ACP sont `FactoMiner` et `factoextra`. `FactoMinerR` propose de multiples méthodes de réductions de dimension dans l'ACP. `factoextra` propose surtout des méthodes pour extraire et présenter graphiquement les résultats des analyses. `FactoMiner` n'est pas détaillé dans ce chapitre car l'ACP est déjà disponible dans le package de base de R : `stats`.

6.1 Fonctions utiles pour l'ACP

Réaliser l'ACP sur une matrice de données quantitatives `data`:

```
res <- prcomp(data, center = TRUE, scale = TRUE)
```

Importance des composantes	
Résumé	`summary(res)`
Valeurs propres et variances	`factoextra::get_eig(res)`
ScreePlot	`factoextra::fviz_screeplot(res, addlabels = TRUE)`

Variables	

Informations
-> Vecteurs propres de X'X
-> Autres informations
~~> Corrélation variables/CP
~~> Cos2/qualité de représentation
~~> Contributions
	
	`res$rotation`
	`var <- factoextra::get_pca_var(res)`
	`var$cor`
	
	`var$cos2`
	
	`var$contrib`

Cercle de corrélations	`factoextra::fviz_pca_var(res, ..)`
Visualiser les contributions des variables	`factoextra::fviz_contrib(res, choice = "var, ..)`

Individus	
Informations (6 premiers individus)	`ind <- factoextra::get_pca_ind(res)` ~> Coordonnées `head(ind$coord)` ~> Cos2 / qualité de représentation `head(ind$cos2)` ~> Contributions `head(ind$contrib)`
Score plot ou Carte des individus	`factoextra::fviz_pca_ind(res, ..)`
Visualiser les contributions des individus	`factoextra::fviz_contrib(res, choice = "ind", ..)`

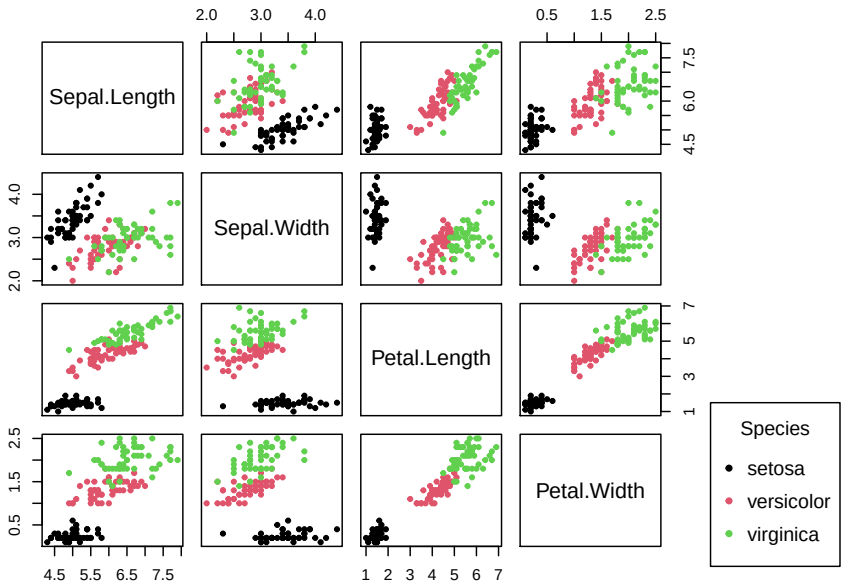
Autres	
Biplot variables-individus	`factoextra::fviz_pca_biplot(res, ..)`

6.2 Données du chapitre ACP

Les données `iris` sont communément utilisées pour illustrer les bases des méthodes multivariées. Elle fournissent, pour 150 iris, 4 variables liées aux dimensions de la fleur : `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`. Ces iris sont de 3 espèces : `setosa`, `versicolor`, `virginica`.

```
# Jeu de données iris disponible directement dans R
data <- iris
```

```
# Extraire la variable qualitative avec l'espèce
groupe <- as.factor(iris$Species)
# Isoler les 4 variables quantitatives (sans la 5eme colonne de iris)
data <- data[,-5]
# Visualiser les données
plot(data, col = groupe , main="Scatter plots des variables prises 2 à 2", oma=c(3,3,3,12), pch=2
par(xpd = TRUE)
legend("bottomright", title = "Species", legend = levels(groupe), col = c(1,2,3), cex = 0.8, pch
```

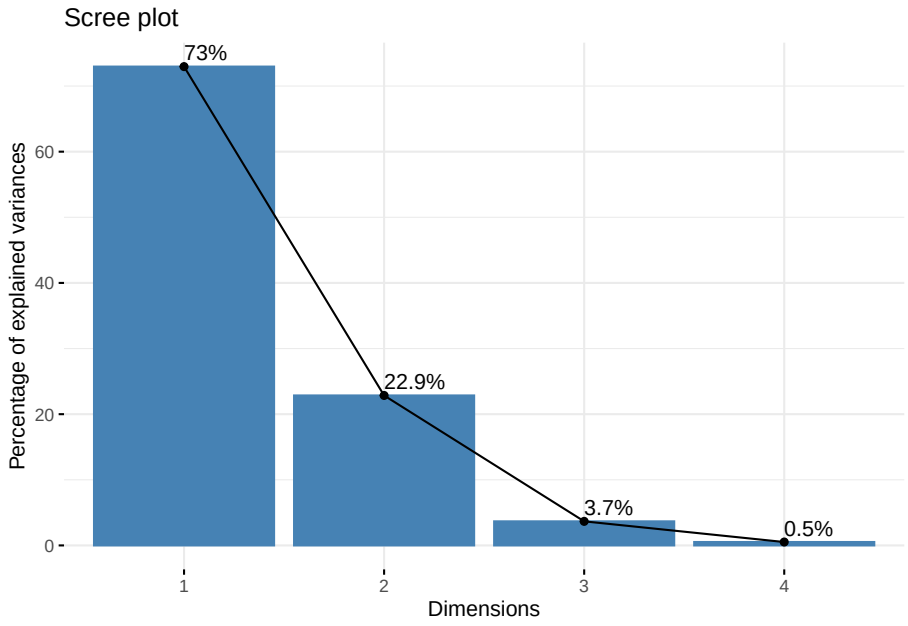


6.3 Utiliser prcomp()

```
res <- prcomp(data, center = TRUE, scale = TRUE)
pander(factoextra::get_eig(res))
```

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	2.918	72.96	72.96
Dim.2	0.914	22.85	95.81
Dim.3	0.1468	3.669	99.48
Dim.4	0.02071	0.5179	100

```
fviz_screeplot(res, addlabels = TRUE)
```

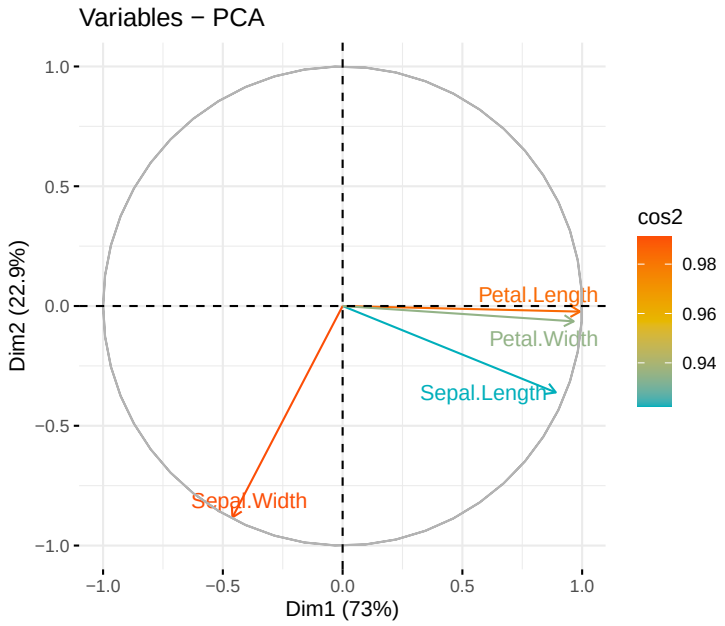


6.4 Utiliser `fviz_pca_var()`

Cercle des corrélations : représentation des corrélations entre les variables de départ et les composantes principales. Les flèches peuvent être colorées en fonction d'une autre variable, ici cos2.

L'option `repel` permet d'éviter que les noms des variables soient superposés.

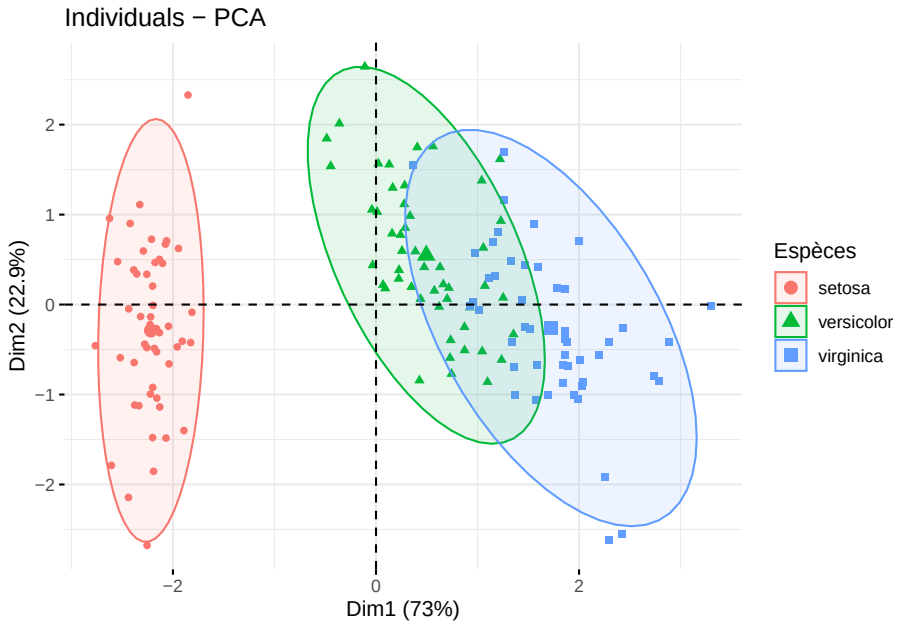
```
fviz_pca_var(res, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"), repel = TRUE )
```



6.5 Utiliser `fviz_pca_ind()`

Graphe des scores ou carte des individus : Projection des individus sur l'espace des 2 premières composantes principales.

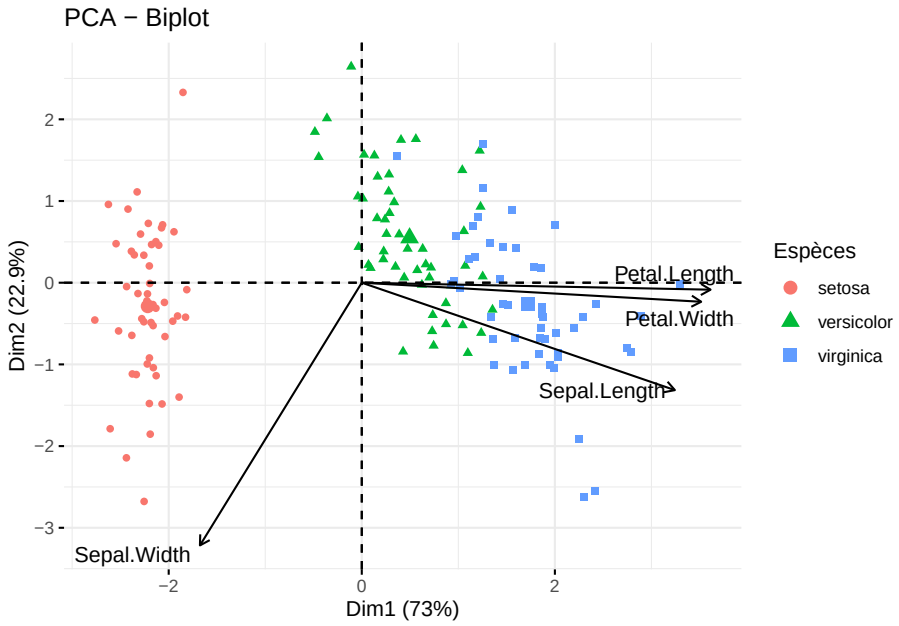
```
fviz_pca_ind(res, axes = c(1,2), geom.ind = "point", col.ind = groupe,
             addEllipses = TRUE, legend.title = "Espèces")
```



6.6 Utiliser `biplot()`

biplot : superposition du graphe des scores et des individus

```
#Biplot variables - individus
fviz_pca_biplot (res, col.ind = groupe, label = "var",
                 col.var = "black", repel = TRUE, legend.title = "Espèces")
```

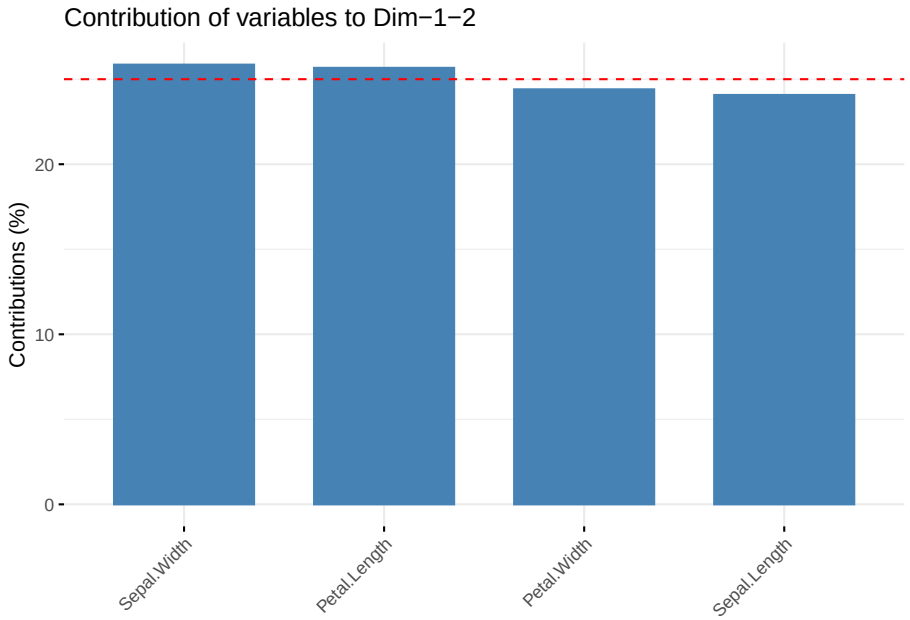


6.7 Utiliser `fviz_contrib()`

6.7.1 Pour les variables

Afficher les contributions des variables aux CP 1 et 2.

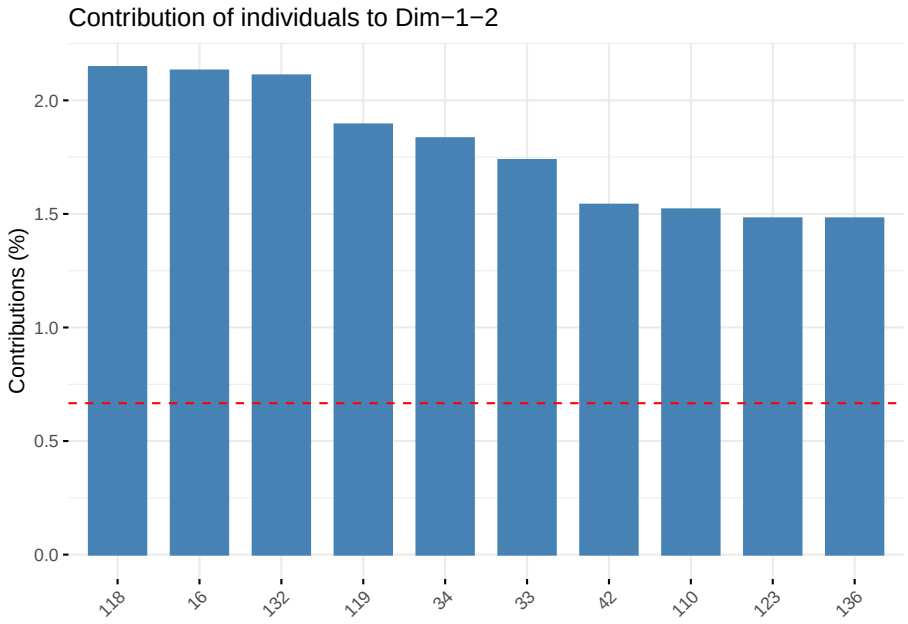
```
fviz_contrib(res, choice = "var", axes = c(1,2))
```



6.7.2 Pour les individus

Afficher les 10 individus avec les plus grandes contributions aux CP 1 et 2.

```
fviz_contrib(res, choice = "ind", axes = c(1,2), top = 10)
```


Contribution of individuals to Dim−1−2
Contributions (%)
2.0
1.5
1.0
0.5
0.0
118
16
132
119
34
33
42
110
123
136

Chapter 7

Clustering

Objectif

~> Diviser un ensemble de données en différents groupes sans connaissance a priori sur les groupes (non supervisé). ~> Les observations d'un même groupe sont supposées partager des caractéristiques communes. La distance euclidienne est en général utilisée pour mesurer la proximité entre individus.

7.1 Fonctions utiles pour le clustering

Standardiser les données avant le clustering

Non prévu dans les options de `kmeans`

```
data <- scale(data)
```

Clustering ascendant hiérarchique - CAH

```
# Clustering hiérarchique = CAH
resCAH <- factoextra::hcut(data, k = .., hc_method = ...,
hc_metric = "euclidean", stand = TRUE)
```

Méthodes de calcul des distances entre groupes `hc_method`: "single", "complete", "average", "ward.D", "ward.D2".

Clustering kmeans

```
# Clustering non hiérarchique = K-Means
resKM <- kmeans(data, centers = .., nstart = 20)
```

Résultats CAH et fonctions liées	
Infos sur le clustering	Résultats : `resCAH`

Résultats CAH et fonctions liées	
~> Nb de groupes proposé par `hcut` (si non fourni)	`resCAH$nbclust`
~> Taille des groupes	`resCAH$size`
~> N° cluster pour chaque donnée	`resCAH$cluster`
Visualiser le clustering dans l'espace des CPs	`factoextra::fviz_cluster(resCAH)`
Dendrogramme	`factoextra::fviz_dend(resCAH, rect = TRUE)`

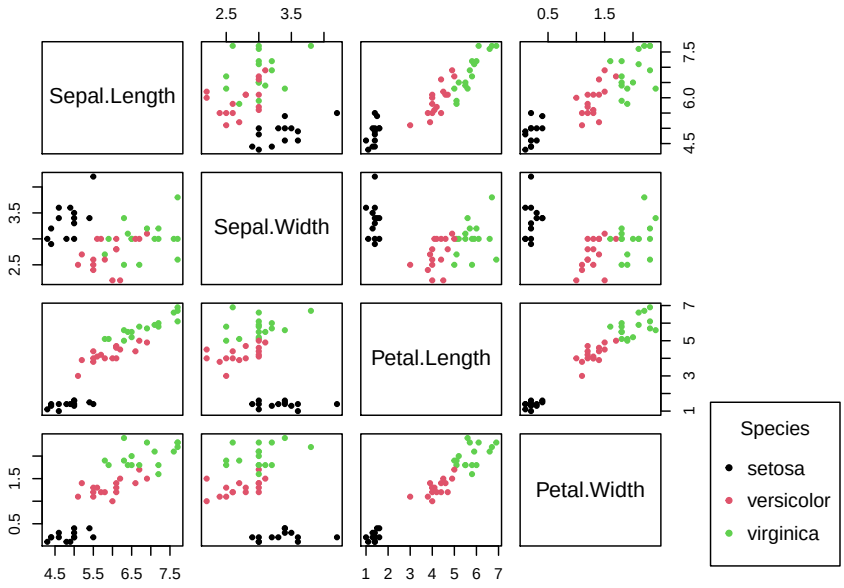
Résultats K-Means et fonctions liées	
Infos sur le clustering	Résultats:`resKM`
~> Membres des clusters	Groupes `resKM$cluster`
~> Centres des clusters	`resKM$centers`
~> Somme totale des carrés	`resKM$totss`
~> Sommes des carrés Within	`resKM$withinss`
~> Total des sommes des carrés Within	`resKM$tot.withinss`
~> Sommes des carrés Between	`resKM$betweenss`
Visualiser le clustering dans l'espace des CPs	`factoextra::fviz_cluster(res, ..)`

7.2 Données du chapitre Clustering

Les données `iris` sont communément utilisées pour illustrer les bases des méthodes multivariées. Elle fournissent, pour 150 iris, 4 variables liées aux dimensions de la fleur : `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`. Ces iris sont de 3 espèces : `setosa`, `versicolor`, `virginica`.

```
# Jeu de données iris disponible directement dans R dont on sélectionne aléatoirement
data <- iris[sample(1:150,45),]
# Extraire la variable qualitative avec l'espèce
groupe <- as.factor(data$Species)
# Isoler les 4 variables quantitatives (sans la 5eme colonne de iris)
data <- data[,-5]
```

```
# Visualiser les données
plot(data, col = groupe , main="Scatter plots des variables prises 2 à 2", oma=c(3,3,3,12), pch=2
par(xpd = TRUE)
legend("bottomright", title = "Species", legend = levels(groupe), col = c(1,2,3), cex = 0.8, pch
```



7.3 Application des fonctions hcut() et kmeans()

```
resCAH <- factoextra::hcut(data,  hc_method = "average",
hc_metric = "euclidean", stand = TRUE)
resCAH
```

```
Call:
stats::hclust(d = x, method = hc_method)

Cluster method   : average
Distance         : euclidean
Number of objects: 45
```

```
datas=scale(data)
resKM <- kmeans(datas, centers=3,nstart=20)
resKM
```

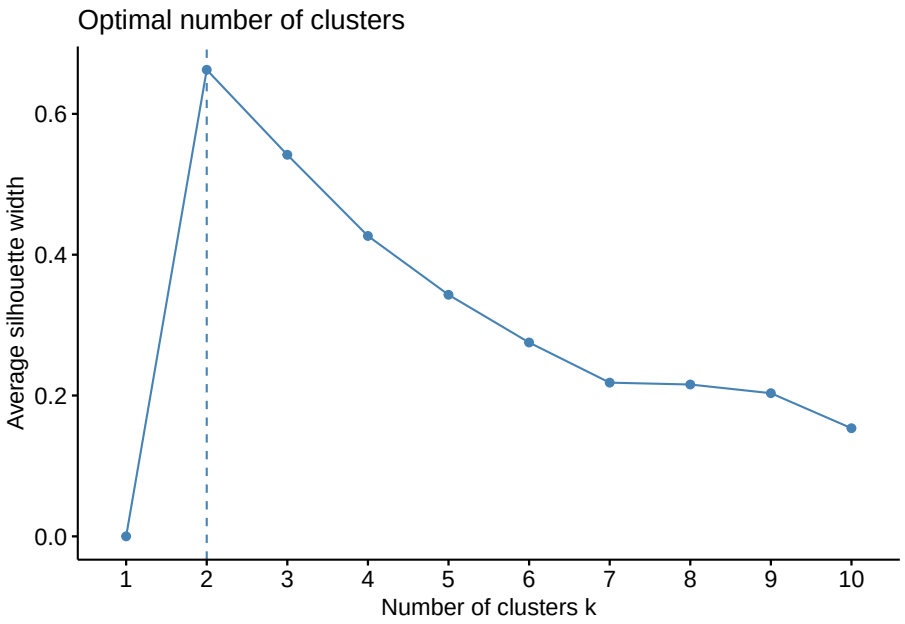
```
K-means clustering with 3 clusters of sizes 18, 14, 13
```

```
Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1   -0.0635247  -0.8123590    0.2248319   0.2000685
2    1.1384947   0.2507954    1.0080508   1.0116469
3   -1.1381139   0.8547174   -1.3968989  -1.3664838

Clustering vector:
 14  50 118  43 150 148  90  91 143  92 137  99  72  26   7  78  81 147 103 117
  3   3   2   3   1   2   1   1   1   1   2   1   1   3   3   2   1   1   2   2
 76  32 106 109 136   9  41  74  23  27  60  53 126 119 121  96  38  89  34  93
  1   3   2   1   2   3   3   1   3   3   1   2   2   2   2   1   3   1   3   1
 69 138 130  63  13
  1   2   2   1   3

Within cluster sum of squares by cluster:
[1] 15.29246 12.59662 10.97006
 (between_SS / total_SS =  77.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

7.4 Aide au choix du nombre de clusters

`factoextra` offre 3 méthodes (`wss`, `silhouette` et `gap_stat`) pour aider au choix du nombre idéal de clusters pour les données analysées. Le dendogramme peut aussi aider en CAH. La fonction `Nbclust` de la librairie `Nbclust` offre aussi une panoplie de critères supplémentaires pour répondre à la question.

```
factoextra::fviz_nbclust(data, FUNcluster =factoextra::hcut, method = "silhouette",hc_
```



Et avec le package `NbClust` (sans sorties)

```
NbClust::NbClust(data, distance = "euclidean", method = "average")
```

7.4.2 K-means

```
factoextra::fviz_nbclust(data, FUNcluster =kmeans, method = "silhouette")
```

7.5 Utiliser `fviz_dend()`

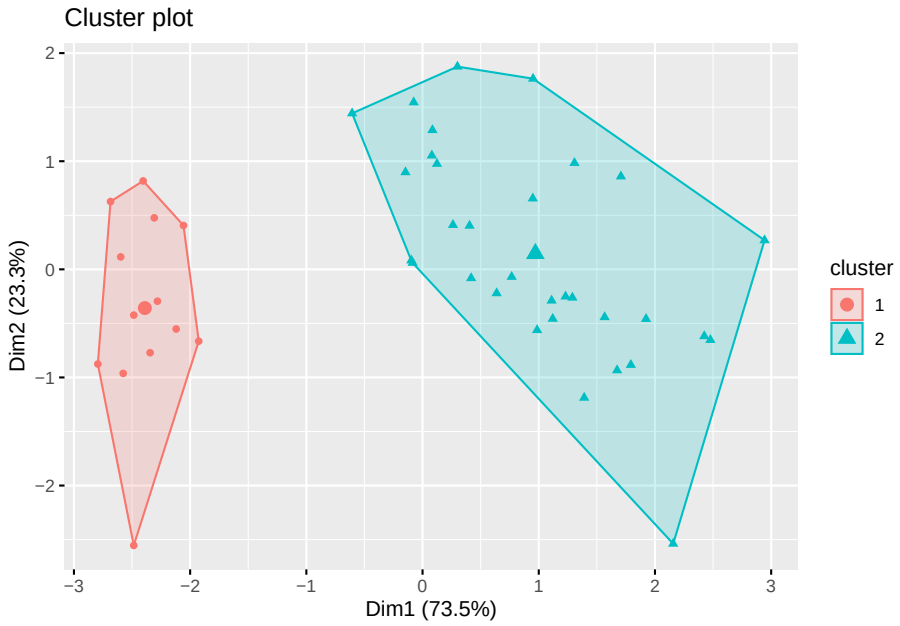
```
factoextra::fviz_dend(resCAH,rect=TRUE)
```

7.6 Utiliser `fviz_cluster()`

Présente une projection des données des clusters dans l'espace des premières composantes principales.

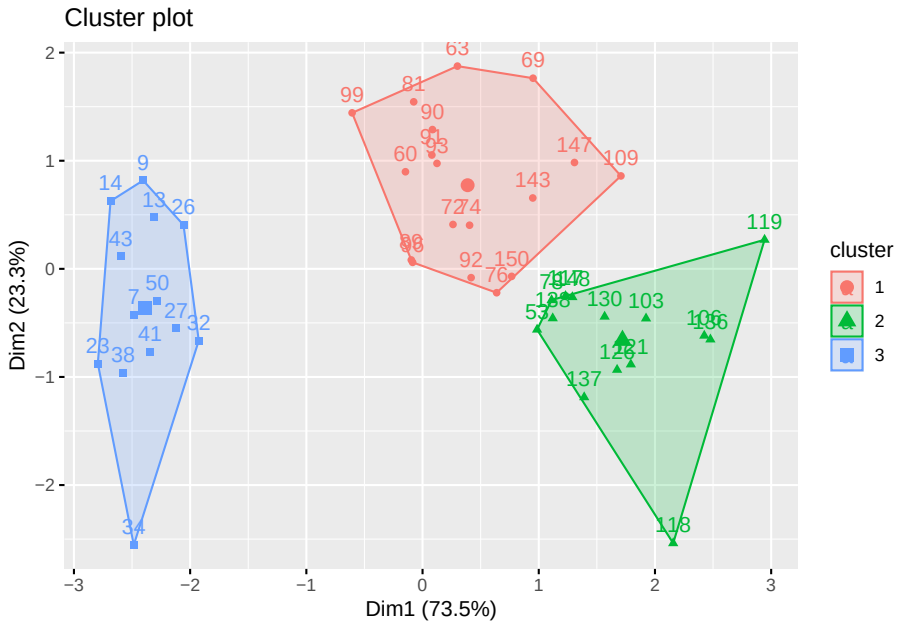
Avec CAH avec 2 clusters

```
fviz_cluster(resCAH, geom = "point", ellipse = TRUE)
```



Avec kmeans avec 3 clusters (il faut fournir les données comme argument)

```
fviz_cluster(resKM, datas)
```



Chapter 8

Modeles mixtes

Objectif

Expliquer une variable réponse quantitative Y en fonction d'effets fixes et/ou d'effets aléatoires.

8.1 Modélisation LMM

```
# ANOVA 1 aléatoire
mod <- lmer(Y ~ 1 + (1|Z), data = ..)

# ANOVA 2 aléatoire
mod <- lmer( Y ~ 1 + (1|Z1) + (1|Z2), data = ..)

# ANOVA 2 aléatoire hiérarchisée
mod <- lmer( Y ~ 1 + (1|Z1) + (1|Z2:Z1), data = ..)

# Modèle mixte
mod <- lmer( Y ~ 1 + X + (1|Z1) + (1|X:Z2), data = ..)
```

8.2 Fonctions utiles sur un objet de type `mod <- lmer()`

Résumé du modèle	`summary(mod)`
Valeur des effets aléatoires	`lme4::ranef(mod)`
Valeur des effets fixes	`lme4::fixef(mod)`

Coefficients du modèle	`coef(mod)`
Résidus / Valeurs ajustées	`residuals(mod, ..)` / `fitted(mod)`
Ecart-type des résidus	`lme4::sigma(mod)`
Matrice de design des effets fixes	`model.matrix(mod)` ou `lme4::getME(mod, "X")`
Matrice de design des effets aléatoires	`lme4::getME(mod, "Z")`
Tests d'hyp sur effets aléatoires	`lmerTest::ranova(mod)`
Graphes des diagnostiques	`plot(mod)` residus vs valeurs ajustées `lattice::qqmath(mod)` qq-plot des résidus `plot(cook.distance(mod))` distance de Cook pour chaque obs
Estimations + IC	`emmeans::emmeans(mod, ..)`
Comparaisons 2 à 2	`lmerTest::ls_means(mod, pairwise = TRUE)`