Away From There
Away From The Day

Statistiques

ligne = raw

Contexte

On a étudié l'observation des données, les tests, les regressions, les stats bivariées etc

En plurivarié, on a fait un peu d'ANOVA

Imaginons un tableau de notes : lignes (individus) → étudiants
colonnes (variables) → matières scolaires
Alors, on va regarder quelles lignes se ressemblent et quelles lignes diffèrent → typologie d'individus

De même pour les colonnes, et on aura → typologie de variables

L'analyse multivariée nous permettra de comprendre

L'étude de 3 dimension (soit pour l'instant 3 variables), est encore représentable... Mais un graphique 3D en nuage de point... Bah ouais le vrai en 2D...
Dès qu'on passe à 4D c'est plus possible.

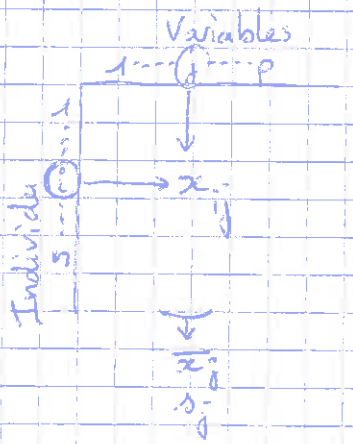
Sachant qu'en écologie ou en sociologie on a des tonnes de variables, ou même en ~~SIG~~ génomique (SIG)... Alors c'est un peu la merde.

Sommaire

- Analyse multivariée (methodes descriptives et théorie statistique)
- Analyse de données (ADD)
- Data mining
- Analyse de grands tableaux

les données

Un tableau c'est sous cette forme :



Le tableau peut être considéré comme une matrice de n lignes et p colonnes.

On va centrer-réduire les colonnes par colonne. (dans certains cas on ne fait que centrer)

x_{ij} devient $\frac{x_{ij} - \bar{x}_j}{s_{x_j}}$

On récupère une matrice "homogène" en valeurs

On peut aussi le faire par colonne...
oui, c'est débile, mais c'est ce que je comprends

↳ en fait son exemple est moyen parce que les variables et individus sont interchangeables. Il aurait dû prendre des variables de différentes nature.

ADD : pas connu dans Donjon et Dragons

Stat

m ind $\mathbb{R}^p$
p ind $\mathbb{R}^n$ → je sais pas trop what it means

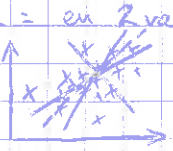
Les méthodes

Distance → Méthodes factorielles (on s'intéresse aux axes de vari°)
↳ ACP, AFC, ACM
→ Méthodes de classification (on s'intéresse aux groupes d'indiv)
↳ CAH

Analyses factorielle

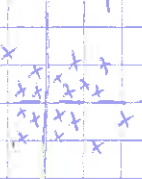
Elle repose sur un principe géométrique → Représent° optimale d'un nuage de point
L'écriture matricielle permet le traitement info.

RAPPEL = en 2 variable



on avait un modèle $y = ax + b$ avec $a = \frac{\Delta xy}{\Delta x^2}$
$b = \bar{y} - a\bar{x} - b$
Donc ça, ça nous donne une droite de reg qui minimise les écarts aux données théoriques, pour des x fixés.

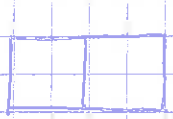
- Mais on peut aussi minimiser la distance horizontale et obtenir une autre rég
- Et pas seulement ! on peut minimiser la distance diagonale et obtenir l'axe principal du nuage !
- Les 3 droites passent toutes par le point de gravité
- Et le second axe lui est ⊥ et passe par le centre de gravité
⇒ Et là on peut changer de repère ! et → c'est ça l'analyse factorielle !



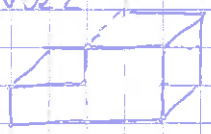
Voilà le nouveau graphique !
C'est un peu « centrés en fait

Illustration de comment les points de vue 3D c'est bien ou pas

Vue 1 → Mais qu'est-ce que c'est ?



Vue 2 → Ah ben oui



→ Les nuages de point, c'est pareil, ya des points de vue mieux.

Pour une matrice à p dimensions
1 → on recherche les p axes principaux
2 →
3 →

j'ai pas eu le tps de regarder

Notes de stats, R

Un axe représente...

code : > PCA(table) ça va, pas trop compliqué!

Combien on garde d'axes quand on regarde le diag. des valeurs propres?
Il sont rangés par force d'inertie.
On garde toujours le premier, au + grand %age d'informations.
Quand la barre n+1 est vraiment + petite que n, on garde les axes jusqu'à n.
Quand 2 barres qui se suivent ont une hauteur similaire, le "plan que forme ces axes est intéressant"

En gardant 2 axes, on peut tirer - un cercle des corrélations

- on projète les coordonnées + la projection est loin du centre, + l'axe dont il provient contribue à la formation de l'axe (parfois, les pts sont pondérés)

Variables factor map On essaie alors d'interpréter ce que sont les axes.
NB : quand des flèches sont petites, elles ne sont pas assez significatives pour être interprétées.

Hypersphère : ensemble de points dans un espace à + de 3 dimensions, situés à égale distance d'un centre.

Objectifs de la PCA
- typologie des variables (faire des groupes)
- typologie des individus
- réduire des dimensions
- hiérarchiser l'information

Il est question de sortir un document de synthèse avec le + d'infos possibles, on distingue variables actives et illustratives

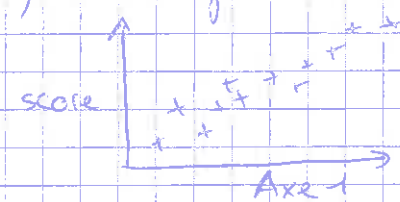
→ C'est bien d'histogrammer. On se fait une idée graph° des données. Good.
→ Faire un graphique "pairs()"
→ et une matrice des corrélations liste $ sous-objet

→ "round" nous permet de voir combien une valeur contribue à un axe.
On peut vérifier avec une variables factor map

eig = valeur propre (eigen value) NB : Σ valeurs propres = nb. variables!

Si on suspect un axe d'être quelque chose et qu'on peut vérifier par des données, let's go corrélation !

exemple :



⟹ L'axe 1 est représentatif des performances des sportifs.

NB aucun rapport → dater les doc
→ numéros de pages
→ en-tête (nom, travail)

TD.

On étudie un tableau :

individu	Sexe	"Type"*

* nouveau 3A ou ISARA 2A) oui, on étudie notre groupe.

On observe :

	H	F	
I	14	16	30
N	2	6	8
	16	22	38

et on se demande si les variables sont liées.

En théorie

	H	F	
I	12,63		30
N			8
	16	22	38

$P(H \cap I) = P(H) \cdot P(I)$

$= \frac{16}{38} \times \frac{30}{38}$

~~12,63~~

$Nb(H \cap I) = P(H \cap I) \times Nb$

$= \frac{16 \times 30}{38\ 38} \times 38$

$= \frac{16 \times 30}{38} = 12,63$

Le Chi² ou Khi² ou χ^2 mesure la différence entre le tableau observé et le tableau théorique. $\chi^2 = \sum_i \frac{(O_i - T_i)^2}{T_i}$

La valeur de ce χ^2 comparée à une valeur seuil nous permet de confirmer ou d'infirmer une hypothèse de dépendance.

On étudiera aussi les profils mais j'ai moyen capito.

- On peut calculer le $\chi^2_{partiel}$ $\left(\frac{(O_i - T_i)^2}{T_i}\right)$ pour estimer l'influence d'une "case" sur le χ^2 total
- On peut calculer le $\chi^2_{partiel}$ relatif $\left(\frac{\chi^2_{partiel}}{\chi^2_{total}}\right)$ → à mettre en %

12/10/2012 cours de stats (amphi bruyant)

"On va voir de quoi on parle... *regarde l'amphi* non, enfin de quoi JE parle, vous, vous parlez pas"

Objectif de la classification : Regrouper des individus semblables dans la classe

Partition = ensemble de classes disjointes qui classent tous les individus

Faire une classification = ajouter une variable qualitative au tableau initial

CAH

Classification Ascendante Hiérarchique → partition emboîtée
construction par le bas → Arbre

⚠ classification ≠ classement

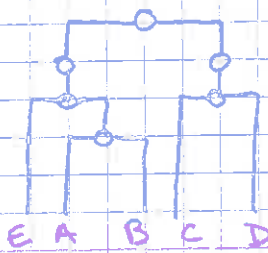
classer = mettre dans des classes connues à priori

classifier : identifier les classes (processus) à partir des données, puis partitionner.

Qualité d'une partition :
- compacité → ensemble cohérent les individus se ressemblent ds une classe
- séparabilité → classes bien distinctes

Hiérarchies, Arbres ou Dendrogrammes

Dendros → arbre en grec



- c'est donc l'ensemble de partitions emboîtées
- La forme est libre
- l'ordre des branches est libre
-

Pour construire une hiérarchie de la sorte on étudie :
- la ressemblance (similarité) $(d(a,b) = d(b,a)$ $d(a,c) = 0$ ssi $a=c$ $d(a,b) \leq d(a,c) + d(c,a)$
- ~~des critères~~ des critères de construction :
  - distance entre deux points
  - critère d'agrégation → Agrégation :
    - saut minimum
    - saut maximum
    - distance moyenne
    - inertie : Méthode de Ward

Distances : - classique (Pythagore généralisé)
- city-block ou "Manhattan" en suivant le quadrillage *
- binaire : nombre de différences

* mais le + court possible

Méthode City block → une fois qu'on a regroupé ceux à + petite distance, on fait les moyennes des 2 regroupés et on recommence

ACP → une seule solution
CAH → plusieurs solutions possibles
(y a aussi un C.A non hiérarchiques)
dans lequel on fixe le nb de classes

Critère d'inertie. Méthode de Ward
Décomposition de l'inertie: $inertie_{tot} = inertie_{inter} + inertie_{intra}$

$inertie_{intra}$ (dans un groupe) → critère de compacité

$inertie_{inter}$ (entre les groupes) → critère de séparabilité

⇒ on agrège en ↘ l'intra et ↗ l'inter.

Coupure de l'arbre (pour avoir des partitions)

coupure absolue → Pour un arbre donné, la coupure en n classes est unique.

critère de coupure → classes compactes et séparables
→ On coupe des branches assez loin de la racine et assez longues (gain d'inertie)

Finir la classification → nommer les classes

Le tableau à savoir lire

Les variables relevées sont celles qui sont particulièrement éloignées de la moyenne générale.

exp: dans une classe un groupe a des moyennes normales, sauf pour les maths où elles sont excellentes.

TD R

1-1 : Premiers pas
1-2 : Statistiques descriptives et manipulations de données
1-3 : Lois théoriques, distributions et formes de distribution
1-4 : Révisions, étude de cas et quelques statistiques bivariées
1-5 : Régression linéaire simple

2-1 : Échantillonnage - estimation
2-2 : Tests statistiques
2-3 : ANOVA
2-4 : Régression simple et multiple

3-1 : ACP, Analyse en Composantes Principales (PCA : Principal Composant Analyze)
3-2 : CAH, Classification Ascendante Hiérarchique (HC : Hierarchique clusters
HCPC :
3-3 : Étude de cas, subventions agricoles

TDR1

TD R - 1
premiers pas

Vincent PAYET

"Heureux les fêlés...car ils laisseront passer la lumière" — Yvan Audouard

1 Introduction

R est un environnement de programmation, un logiciel de gestion de données et de production de sorties graphiques. En quelques années il est devenu un outil majeure de l'analyse statistique. C'est un logiciel libre (licence GNU), distribué par le CRAN (Comprehensive R archive network) à l'adresse suivante : (`http://cran.r-project.org`). Vous pouvez l'installer sur votre machine personnelle sur de nombreux systèmes (Windows, Mac, Linux). Son utilisation est possible sans restriction y compris dans un contexte professionnel.

L'objectif de ce premier TD est de découvrir R, de poser quelques bonnes pratiques de travail avec ce logiciel et d'explorer certains aspects de bases de manipulation de données et de création de graphiques.

2 Première session

2.1 Bonne pratique 1 : le répertoire courant

Avant d'ouvrir R, crécz un répertoire de travail sur la machine local, par exmple TDR1 sur le bureau. Vous conserverez une copie de ce répertoire en fin de TD sur votre espace personnel (K ://). En revanche, ne faites pas le TD sur le serveur !

La première chose à faire après avoir lancer R est de modifier l'environnement de travail, via le menu fichier, "Changer le répertoire courant" : sélectionnez le dossier TDR1 que vous venez de créer.

On peut ensuite verifier que le changement est effectif avec la commande `getwd()`. C'est dans ce répertoire que vous poserez les fichiers de données à importer et où vous trouverez les éventuels exports et fichiers d'historique de vos commandes.

2.2 Premières commandes

Voici des commandes R. Le premier exercice est de les saisir dans la console (après l'invite de commande marquée par le signe >). Vous devez comprendre comment fonctionne chacune de ces commandes. Le signe dièse (#) marque les commentaires non pris en compte par le logiciel.

```
z <- 40   # on peut affecter une valeure à un objet
z*2       # on peut utiliser cet objet,
          # la valeure est renvoyée mais perdue.
x <- 1:10 #
x         #
y <- 3*x+5#
Y <- 3*x+5+rnorm(x,0,2)# idem, mais on aloute une erreur aléatoire
y         #
Y         #
```

La casse (lettre minuscule ou majuscule) compte. On a donc crée deux objets différents : y et Y. On peut ensuite faire des calculs sur nos objets :

```
mean(x)
moY <- mean(Y)
```

On peut manipuler du texte. Il faudra alors le placer entre guillements "".

```
phrase <- "Bonjour !"
phrase  # phrase est un objet. Cet objet contient le texte "Bonjour !"
```

Notez que vous pouvez circuler dans les commandes déjà tapées avec les flêches *haut* et *bas* du clavier.

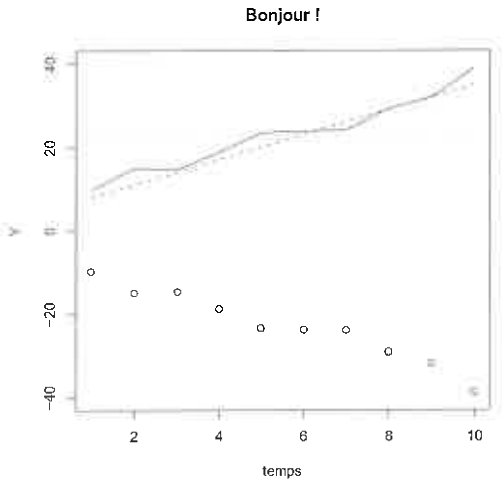
2.3 Premiers graphiques

On peut représenter facilement des données avec les commandes graphiques de bases, dont la fonction `plot()`.

```
> plot(x, Y)
> plot(x, Y, type = "l", xlab = "temps", ylab = "réponse")
```

Vous noterez que la fonction `plot()` écrase le graphique précédent. Au contraires, d'autres fonctions graphiques ajoutent des éléments au graphique déjà crée. Ainsi la fonction `lines()`, ci-dessous ajoute la droite d'équation $y = 3x + 5$ en pointillé (avec le paramètre `lty`).

```
> plot(x, Y, type = "l", xlab = "temps", ylim = c(-40, 40))
> lines(x, y, lty = 2)
> abline(h = moY, lty = 2, col = "red")
> points(x, -Y)
> title(phrase)
```



2.4 L'environnement de travail

Depuis le début de la session, vous avez crée beaucoup d'objet. On peut obtenir la liste de tous ces éléments avec la commande `ls()`. On peut ensuite en supprimer avec `rm()` : essayer par exmple `rm(z)`. Puis vérifiez que l'objet a bien disparu en relaçant la commande `ls()`. De plus si vous quittez R avec `q()`, il vous propose d'enregistrer la session. Essayer puis regardez le contenu du fichier .Rhistory dans votre répertoire de travail. Pour cela il faut forcer l'ouverture avec l'éditeur de texte. Vous trouverez aussi un fichier .Rdata qui contient les objets crées, celui-ci ne peut être lu.

2.5 Bonne pratique 2 : le script

Le script est un simple fichier texte (de type `.txt`) qui contient des commandes R et si possibles des commentaires (introduits par #). Le script permet de conserver les commandes saisies, le déroulement d'une analayse, des fonctions personnelles... Il devra avoir une extension de type .r ou .R et être dans le répertoire de travail. Il est judicieux de conserver plusieurs scripts et de les nommer de façon parlante. Il est conseillé aussi de commenter vos scripts.

→ on peut ouvrir le script dans R et le renvoyer dans la console avec Ctrl+R

Vous pouvez récupérer le fichier `TD1.session1.r` sur le serveur, le déposer dans votre répertoire de travail et l'ouvrir dans R (menu fichier, etc.). Pour la suite vous pouvez utiliser ce script en essayant les lignes unes à unes : quand le curseur est sur une ligne, la séquence CTRL+R envoie la ligne dans la console. Ajoutez vos codes et commentaires personnels à ce premier script.

2.6 Bonne pratique 3 et 4 : l'aide et un glossaire

Nous avons déjà vu plusieurs commandes. Il est conseillé de maintenir à jour un glossaire des commandes déjà vues. Pour cela utiliser par exemple un fichier de tableur (pour le tri alphabétique).

Par ailleurs pour obtenir de l'aide sur une nouvelle commande, on peut utiliser la commande help() qui peut être abrégée avec ? :

```
help(plot)
?plot
help.start() # permet d'ouvrir l'aide dans le navigateur (ce sont des fichiers locaux)
```

L'aide d'une commande contient notamment, une description de la commande, un ligne sur l'usage de cette commande et en fin de page, des exemples d'utilisation que vous pouvez tester dans la console.

```
plot                    package:graphics                    R Documentation

Generic X-Y Plotting

Description:
     Generic function for plotting of R objects.  For more details
     about the graphical parameter arguments, see 'par'.

Usage:
     plot(x, y, ...)
```

3 Quelques concepts

3.1 Les modes

Le mode est la nature de la variable telle que l'interprète le logiciel. Le mode peut être *numeric*, *character*, *logical*, *NULL*. On peut connaitre le mode par la commande `mode()`, le tester avec des commandes de type `is.numeric()`, ou le forcer avec les commandes de type `as.character()`. Quelques exemples :

```
z <- 1:10
mode(z)
is.character(z)
zchar <- as.character(z)
is.numeric(zchar)
z;zchar
```

l'objet *z* contient les chiffres 1 à 10, l'objet *zchar* contient ces chiffres considérés comme du texte, donc présentés entre guillements. Le ; permet de mettre deux commandes sur une même ligne.

3.2 Les vecteurs

Un vecteur contient un seul type de données mais plusieurs valeurs. Il est donc caractérisé par son mode et sa longueur. Il permet de stocker et manipuler une série statistique. Voici des façons de créer un vecteur :

TDR1


```
aa <- c(5,6,2,1,9);aa
length(aa)
mode(aa)
bb <- seq(1,10,by=0.5);bb
cc <- seq(2,10,length=5);cc
dd <- rep(c(2,5),3);dd
rep(c(2,5),each=3)
ee <- c("Alf","Joe","Hal","Nop","Quo")
ee
c(aa,bb)
c(aa,ee)
```

seq
rep

aa, *bb*, etc. sont des objets de type vecteur. `c()` permet de créer un vecteur, soit à partir de données ponctuelles, soit à partir de deux vecteurs existants. Dans ce cas on remarque qu'il n'y a qu'un mode par vecteur.

3.3 actions sur les vecteurs

Sur les vecteurs, on peut faire des calculs vectorisés. Ces calculs ne nécessitent pas une boucle de programmation explicite :

la série aa : c(5,6,2,1,9)

```
aa*2 → multiplie chaque 1 par 2
aa*2:6 → multiplie le 1er terme par 2, le 2nd par 3... le 5e par 6
aa-mean(aa)
```

On peut aussi utiliser des fonctions spécialisées, notamment des fonctions de statistique, comme la moyenne avec `mean()` que nous avons déjà vu.

```
rev(aa) → reverse
min(aa);max(aa)
sort(aa) → trie
length(aa)
sum(aa)
cumsum(aa) → cumule au fur et à mesure
var(aa)
```

La fonction `var()` est particulière. En vous rappelant la formule de la variance, écrivez un calcul vectoriel de la variance et comparez avec la sortie proposée par R avec `var()`*Par exemple la somme des écarts à la moyenne serait `sum(aa-mean(aa))`. Nommez votre variance *varCalc* et comparez ça :

```
n <- length(aa)
varCalc*n/(n-1)
var(aa)
```

La variance proposée par défaut sur R est une variance estimée de population (voir dans l'aide). Elle est calculé en divisant la SCE par $(n-1)$.

↑ somme du carré des écarts

* `sum((aa-mean(aa))^2)/length(aa)` ça veut dire $s_x^2 = \frac{1}{n}\sum (x_i - \bar{x})^2$

3.4 Les matrices

Ce sont des tableaux de données, toutes de même mode. On peut les créer avec des vecteurs en lignes (`rbind()`)ou en colonnes(`cbind()`), ou avec la fonction `matrix()`. Une matrice est caractérisée par son mode et ses dimensions.

```
mat1 <- cbind(aa,ee,cc)
mat1
mode(mat1)
dim(mat)
mat2 <- matrix(1:16,ncol=4);mat2
?matrix # pour plus d'info
```

3.5 Les tableaux de données, data.frame

Un data frame contient plusieurs colonnes de mode différents. C'est une structure majeure. Les données seront importées dans ce format. Il existe d'autres structures (list, ts, fonctions) que nous verrons par la suite.

```
?data.frame
mesDonnees <- data.frame(aa,ee,cc,1:5)
mesDonnees        #
dim(mesDonnees)   # 5  4
str(mesDonnees)   # -> description approximative
names(mesDonnees)# "aa" "ee" "cc" "X1.5"
mesDonnees$aa     #
```

3.6 l'indexation et manipulation des données

On peut accéder à des éléments de vecteur, de matrice ou de data.frame par un système d'index, commentez chacune des lignes suivantes :

```
aa        #
aa[2]     # -> 2e valeur de la série aa
aa[2:5]   # -> les valeurs de la 2e à la 5e
aa[-3]    # -> les valeurs sauf la 3e
          #
mat2[2,3]#
mat2[,3] # -> colonne 3
mat2[1,] # -> ligne 1
mesDonnees[,2] # ?? pareil car ee = colone 2
mesDonnees$ee  # ??
```

On peut croiser des vecteurs différents. Ici on considère tous les éléments de *cc* pour lesquels les élements de même rang dans *aa* sont supérieurs à 3 :

```
cc[aa>3]
aa>3 # étant un vecteur de mode logical
```

Il n'est pas nécessaire d'avoir défini un data.frame avant.

mesDonnées :

aa	cc
5	2
6	4
2	6
1	8
9	10

TDR

3.7 l'import

On peut importer un tableau de données depuis un fichier texte avec la fonction `read.table()` qui crée un *data.frame*. Récupérer le fichier *t3var.txt*, placez le dans votre répertoire de travail. Ouvrez le avec un éditeur de texte pour savoir à quoi il ressemble, puis importez le dans R comme ceci :

```
t3var <- read.table("t3var.txt")
is.data.frame(t3var)
names(t3var)
head(t3var)
```

header = TRUE

Comment préciser que la première ligne contient des noms de colonnes ? Voir dans l'aide de la fonction. Après un appel correct à la fonction, on obtient avec `head(t3var)` :

```
  sexe poi tai
1    h  60 170
2    f  57 169
3    f  51 172
4    f  55 174
5    f  50 168
6    f  50 161
```

```
names(t3var)
head(t3var)
```

4 Exercice avec t3var

Pour faire l'exercice suivant il faut utuiliser les différentes fonction du TD : Puis répondez aux questions suivantes en utilisant R :

- Importer les données t3var
- Quels sont les noms des variables de ce jeu de données ?
- Définir le contexte statistique (combien d'individus, de variables, types des variables).
- Sélectionner les individus 1, 10, 20 et 66
- Sélectionner les femmes de plus de 170 cm
- Sélectionner les femmes de taille supérieure à la taille moyenne des femmes.
- Pour les individus 10 à 20, donner toutes les variables sauf la première.
- Donner la moyenne des poids, pour tous, puis par sexe.
- Représenter graphiquement les données avec `plot()` et `hist()`

```
t3var ← read.table("t3var.txt", header = TRUE)
names(t3var)
str(t3var)
t3var[1, 10, 20, 66]
t3var[
```

TDR 2

TD R - 2
Statistiques descriptives et manipulations de données

Vincent PAYET

"Il y a trois sortes de mensonges : les mensonges, les sacrés mensonges et les statistiques." — Mark Twain

→ Penser à définir un répertoire courant.
commenter les scripts

1 Introduction

Lors de cette séance nous allons voir comment calculer des statistiques simples et produire quelques graphiques à partir d'un jeu de données et comment manipuler des données. Manipuler, c'est-à-dire comment sélectionner certains individus ou certaines variables et donc utiliser un sous-ensemble du jeu de données mais aussi comment créer des nouvelles variables ou concaténer des données. C'est aussi créer des fonctions.

L'étape de mise en forme des données est toujours une étape clé d'une analyse statistique. Ce TD vous donne les éléments de base pour ce type de réalisation.

Le TD contient deux parties. Dans la première, on vous propose des exemples de code sur un premier jeu de données. L'exercice est d'essayer ces commandes et de comprendre ce qu'elles produisent et comment elles fonctionnent. Dans la seconde partie, vous allez produire vous même les commandes pour répondre à un exercice sur d'autres données en vous inspirant des exemples précédents.

Bonnes pratiques

Comme nous l'avons vu lors du premier TD, n'oubliez pas de :
- créer un répertoire de travail sur le disque local (C :/) ←
- changer le répertoire courant dans R avant toute autre action
- travailler dans un script
- commenter vos scripts
- consulter l'aide sur les commandes nouvelles ou mal comprises
- compléter votre glossaire de commande

2 Des Iris et des commandes à découvrir

Pour cet exercice, nous allons utiliser un des nombreux jeux de données présents par défaut dans R, les fameux Iris de Fisher. Citons l'aide :

> This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *I. versicolor*, and *I. virginica*

2.1 Import de données

Il est possible d'importer dans R des données au format texte. Les données sont disponibles sur e-campus dans deux fichiers *iris1.txt*, *iris2.txt*, vous pouvez les ouvrir avec un éditeur de texte pour visualiser leur contenu. Faites une copie dans votre répertoire de travail avant de les importer, par exemple :

```
> iris1 <- read.table("iris1.txt")
> is.data.frame(iris1)
> names(iris1)
> head(iris1)
```

Comment préciser que la première ligne contient des noms de colonnes ? Voir dans l'aide de la fonction `help(read.table)`. Après un appel correct à la fonction, que vous devez écrire, on obtient avec `head(iris1)` :

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

et avec `names(iris1)` :

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

2.2 Manipulations

Il faut d'abord concaténer nos deux data.frame, `iris1` et `iris2`, c'est-à-dire les mettre bout à bout ou l'un en dessus de l'autre. Essayez et choisissez à partir des commandes suivantes, le résultat sera stocké dans un objet du nom de `iris3` :

```
> cbind(iris1, iris2)
> rbind(iris1, iris2)
> dim(cbind(iris1, iris2))
> iris3 <- rbind(iris1, iris2)
```

Ces commandes `rbind()` et `cbind()` fonctionnent aussi pour des vecteurs ou des matrices.

Les commandes suivantes peuvent vous être utiles pour la suite. Vous devez les tester, les comprendre et décrire leur action.

```
> names(iris3)
> summary(iris3)
> dim(iris3)
> iris3[1, ]
> head(iris3)
> head(iris3[-2, ])
> iris3[-(2:140), ]
> iris3[, 3:5]
> iris3[c(1, 5, 10, 100), ]
> iris3[c(1, 5, 10, 100), 3:5]
> iris3$Sepal.Length
> iris3[iris3$Species == "virginica", ]
> iris3$Sepal.Length[iris3$Species == "virginica"]
> iris3[iris3$Sepal.Length > 7, ]
> iris3[iris3$Sepal.Length < 6 & iris3$Species == "virginica",
+     ]
> split(iris3$Sepal.Length, iris3$Species)
> liste1 <- split(iris3$Sepal.Length, iris3$Species)
> lapply(liste1, mean)
> tapply(iris3$Sepal.Length, iris3$Species, mean)
> sample(iris3$Sepal.Length, 5)
```

L'indexation (avec les crochets [ ]) et la sélection par les noms (avec l'opérateur $) permettent de choisir des sous-ensembles des données. Les fonctions de la famille `apply` permettent d'appliquer une fonction à un sous-ensemble de données. Nous aurons l'occasion de les utiliser.

2.3 Statistique

Quelques fonctions, surtout statistiques, appliquées aux longueurs de sépales de l'espèce *virginica* :

```
> slv <- iris3$Sepal.Length[iris3$Species == "virginica"]
> sort(slv)
> length(slv)
> mean(slv)
> sd(slv)
> var(slv)
> min(slv)
> max(slv)
> median(slv)
> quantile(slv, 0.25)
> quantile(slv)
> boxplot(slv)
> boxplot(iris3$Sepal.Length ~ iris3$Species)
> hist(slv)
> par(mfrow = c(2, 2))
> boxplot(slv)
> boxplot(iris3$Sepal.Length ~ iris3$Species)
> hist(slv)
> table(iris3$Species)
> plot(table(iris3$Species))
```

Remarque : Pour l'exercice nous utilisons un jeu de données modifié. Les données complètes sont présentes dans R et peuvent être appelées directement avec la commande suivante, l'aide permet d'en savoir plus :

```
> iris
> help(iris)
```

2.4 Fonctions

Dans R les fonctions sont des objets. On peut en créer de nouvelles et leur donner un nom. Par exemple pour créer une fonction qui renvoie un nombre multiplié par 10 (pour un changement d'unité) et que je nomme `f10()` :

```
> f10 <- function(x) {
+     return(x * 10)
+ }
```

On utilise ensuite la fonction ainsi :

```
> f10(5)
> f10(5:7)
> f10("a")
```

En s'inspirant du TDR1, on peut écrire une fonction `varP()` qui calcule la variance telle que nous l'avons vue en cours : $s_x^2 = \frac{1}{n}\sum(x_i - \bar{x})^2$

```
> varP <- function(x) {
+     effectif <- length(x)
+     sce <- sum((x - mean(x))^2)
+     return(sce/effectif)
+ }
```

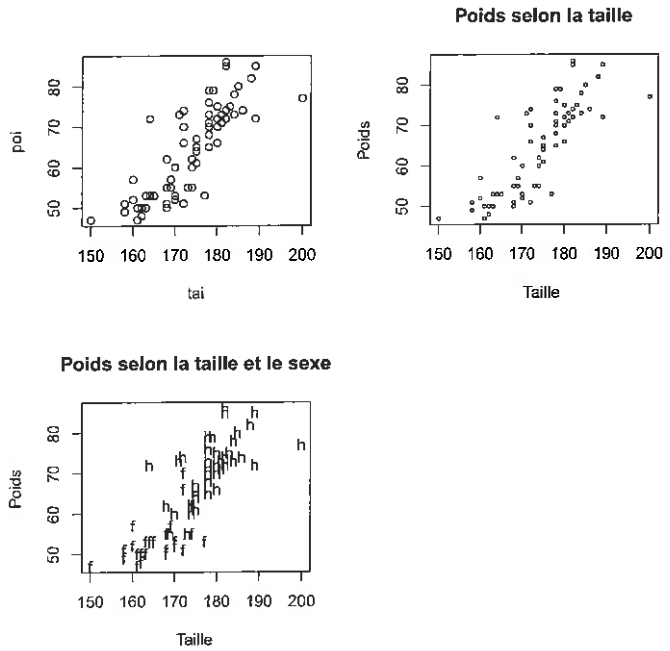
Vous pouvez comparer `var(1:4)` et `varP(1:4)`.

3 Exercice : un tableau de 3 variables

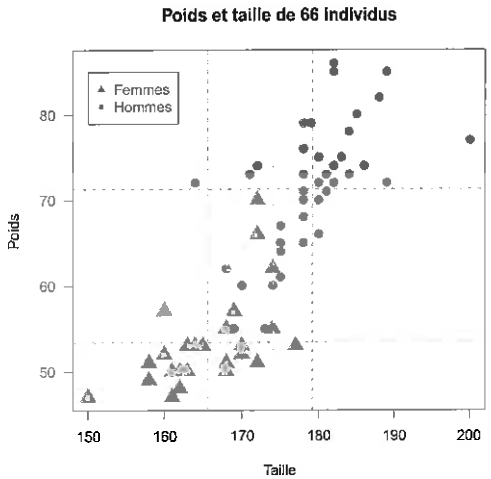
Nous allons utiliser le jeu de données *t3var*. C'est un tableau avec 3 variables. Les fichiers sont disponibles sur e-campus, en deux parties. Il s'agit d'importer ces données, de réaliser quelques manipulations et de résumer les données :

- Importer les données et les concaténer dans un objet `t3var`.
- Quels sont les noms des variables de ce jeu de données ?
- Définir le contexte statistique (combien d'individus, de variables, types des variables).
- Sélectionner les individus 1, 10 et 20.
- Sélectionner les femmes de plus de 170 cm, combien sont-elles ?
- Pour les individus 10 à 20, donner toutes les variables sauf la première.
- Sélectionner les femmes de taille supérieure à la taille moyenne des femmes, donner l'effectif de ce sous-groupe.
- Donner la moyenne des poids pour tous, puis par sexe.
- Donner la variance des poids pour tous, puis par sexe (la variance du cours).
- Écrire une fonction qui calcule l'indice de masse corporelle ($IMC = masse/taille^2$).
- Représenter les histogrammes des IMC pour les hommes et pour les femmes.
- Représenter graphiquement les données avec `plot()` .

```
> par(mfrow = (c(2, 2)))
> plot(poi ~ tai)
> plot(poi ~ tai, main = paste("Poids selon la taille"), xlab = "Taille",
+     ylab = "Poids", cex = 0.5)
> plot(poi ~ tai, main = paste("Poids selon la taille et le sexe"),
+     xlab = "Taille", ylab = "Poids", type = "n")
> points(tai, poi, pch = as.character(sexe))
```



Avec un peu de de recherche, on peut obtenir ça (en couleur sur la version pdf) :



Pour parvenir à ce résultat, on peut par exemple :
- représenter les hommes seuls
- ajouter les femmes avec `points()`)
- ajouter les moyennes avec `abline()`
- ajouter la legende avec `legend()`

TDR3

TD R - 3
Lois théoriques, distributions et formes de distribution

Vincent PAYET

"La loi est implacable, mais la loi est imprévisible. Nul n'est censé l'ignorer, mais nul ne peut la connaître." — Georges Perec, W. ou le souvenir d'enfance

1 Introduction

À travers la notion de distribution, ce TD est une introduction au maniement des lois théoriques avec R et un élement du cours sur les paramètres de formes. C'est aussi l'occasion de revenir sur la représentation graphique de série de données statistiques.

Bonnes pratiques

Nous nous permettons d'insister :
- Créer un répertoire de travail sur le disque local (C :/)
- Changer le répertoire courant dans R avant toute autre action
- Travailler dans un script
- Commenter vos scripts
- Consulter l'aide sur les commandes nouvelles ou mal comprises
- Compléter votre glossaire de commande

2 Distributions et lois théoriques

On appelle distribution un ensemble de données groupées associées à des effectifs. Dans le cas d'une variable continue, on regroupe les observations ou unités (ou encore individus) par classes. Les distributions obervées ou empiriques sont des réalisations d'observations ou d'expérience.

Pour modéliser ces distributions observées, on peut utiliser des distributions théoriques, décrites par des fonctions mathématiques. On parle alors de lois théoriques, elles peuvent être continues ou discrètes selon la nature de la variable qu'on veut modéliser. Vous connaissez entre autre les lois normales, de student,

de Fisher, de χ^2 (lois continues) ou les lois discrètes suivantes : binomiale, Poisson.

R propose de nombreuse lois théoriques, en voici quelques unes :

Lois	Nom dans R
Normale	norm
Student	t
Fisher	f
Uniforme	unif
Chi-deux	chisq
log-normale	lnorm
Binomiale	binom
Poisson	pois
...	...

Comment peut-on les utiliser ? les noms proposés ne sont pas des fonctions R. Il faut ajouter un des préfixes suivants : `d`, `p`, `q`, `r`. On obtient alors dans l'ordre la fonction de **densité**, la fonction de répartition avec les **probabilités**, les **quantiles** (une valeur de `t`) ou un nombre **aléatoire** (random en anglais).

On peut ainsi calculer la probabilité d'obtenir 2 pour une loi binomiale de paramètre n=5 et p=0,05 :

```
> dbinom(2, 5, 0.05)

[1] 0.02143438
```

On peut aussi chercher la probabilité d'obtenir au plus 2 pour cette même loi ($P(X \leq 2)$)

```
> pbinom(2, 5, 0.05)

[1] 0.9988419
```

Vous pouvez retrouver ces valeurs dans vos tables. Vous avez maintenant accès à toutes les tables possibles. On peut par exemple générer la table pour $X\ B(10, 0, 25)$:

```
> dbinom(1:5, 10, 0.25)

[1] 0.1877117 0.2815676 0.2502823 0.1459980 0.0583992
```

Il en va de même avec les autres lois, `pnorm`, `dnorm`, `pchisq`... `qnorm` permet d'obtenir une valeur de la variable normale centrée réduite pour une probabilité donnée. Pour quelle valeur de T avons nous $P(T \leq t) = 0,95$?

```
> qnorm(0.95)

[1] 1.644854
```

Ces différentes fonction ont bien entendue leurs fiches d'aide :

- `?pnorm` permet d'accéder à l'aide de d|p|q|rnorm et nous indique que nous pouvons utiliser les fonctions pour d'autres lois que la loi normale centrée réduite.
- `?qt` permet de savoir comment fixer le degré de liberté...

Des distributions presque empiriques

Les fonctions `rloi` permettent d'obtenir une ou des valeurs aléatoires suivants une loi particulière. Nous allons pouvoir faire des expériences de simulation. On dit aussi des expèriences *in silico*.

Soient 127 valeurs suivants une loi normale de moyenne 11 et d'écart-type 2,5 : `rnorm(127,11,2.5)`, on peut les arrondir à une seule décimale :

```
> notes <- round(rnorm(127, 11, 2.5), 1)
```

On peut voir là des notes possibles du QCM... Quelle est la moyenne observée ? certainement proche de 11. Voyons de plus près avec `mean(notes)` et `sd(notes)`. On peut aussi proposer un graphique :

```
> par(mfrow = c(2, 1))
> hist(notes, main = "")
> abline(v = mean(notes), col = "red", lty = 2)
> title("Distributions des notes")
> rug(notes)
> boxplot(notes, horizontal = TRUE, col = "grey")
```

Pour représenter des fonction de densités de variables discrètes on utilisera selon les cas :

- `plot(table(x))`
- `plot(x, type="h")`

Exercices

On peut reprendre les exercices 1, 4, 9, 10 et 18 du TD papier 2.

3 Paramètres de formes

Les paramètres de formes permettent de caractériser une distribution selon son degré de symmétrie et d'applatissement. Ils viennent en complèment des paramètres de positin et de dispersion.

Symétrie

On peut utiliser le coefficient s de Pearson :

$s = \frac{3(\bar{x}-\tilde{x})}{\sigma}$

s sera compris entre -1 et 1. Si S est négatif la courbe est étalée à gauche, si s est positif la courbe est étalée à droite, si s est proche de 0, la courbe est symétrique.

Il existe d'autres coefficients (s de Yule et Fisher γ_1).

$\gamma_1 = \frac{\mu_3}{\sigma^3}$ avec $\mu_3 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3$

Applatissement

On peut utiliser le coefficient γ_2 de Fisher :
$\gamma_2 = \frac{\mu_4}{\sigma^4}$ avec $\mu_4 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4$

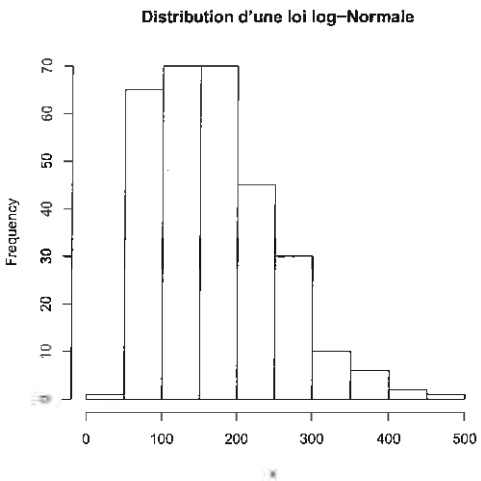
Exercices

Pour les distributions suivantes, vous devez :
- Générer une série de 300 valeurs → seq(1er valeur, ne valeur, by = pas)
- Représenter graphiquement la série observée rnorm(n valeurs voulues)
- représenter graphiquement la loi théorique
- Calculer moyenne et écart-type de cette série
- Calculer un coefficient de symétrie et d'applatissement
- faire un mini compte rendu sur word avec les graphiques et les calculs.

Les distributions :
- Poisson $\lambda = 5$
- Poisson $\lambda = 20$
- Binomiale $n = 5, p = 0,1$
- Binomiale $n = 100, p = 0,45$
- Normale $\mu = 100, \sigma = 10$
- Fisher à 3 et 4 ddl
- Log-normale de moyenne 5 et écart-type 0.5

A ← rpois(300,5)
plot table(A)

B ← dpois(1:15,5)
plot(B, type="h")



TDR4

TD R - 4
Révisions, étude de cas et quelques statistiques bivariées

Vincent PAYET

"Qui s'instruit sans agir, laboure sans semer." — Proverbe arabe

1 Introduction

Durant cette séance vous allez analyser un jeu de données et produire un compte-rendu à déposer sur e-campus. C'est l'occasion d'appliquer les connaissances vues depuis le début du cours et en particulier de vous entraîner pour le TP noté.

L'exercice est certainement *trop* long. Rendez impérativement votre document en l'état à la fin de la séance. Les questions non traitées peuvent vous servir de base de révision pour la suite. Une correction pour cet exercice sera déposée sur e-campus après le passage du dernier groupe.

En plus de cette fiche de TD, vous disposez sur e-campus :
- de scripts commentés comme corrections des TDR1 à R3.
- d'une fiche pour faire un test de χ^2 avec R
- d'un modèle de document pour votre compte-rendu.

Consignes

Vous réaliserez un compte-rendu par binôme avec un traitement de texte. Le document devra comporter vos noms, un titre, des numéros de page et sera structuré (c'est-à-dire avoir un plan apparent).

Vous renommerez le fichier d'après vos noms `NOM1-NOM2.doc` (ou `.odt`). Le document est à déposer sur la partie travaux du cours de 1A sur e-campus. Ces travaux ne seront pas notés, toutefois l'absence de fichier entraînera une pénalité pour votre travail de la semaine prochaine (0,8 x note). Avant de partir assurez-vous que votre travail est bien rendu.

Pour chaque question :
- Rappelez la question.
- Proposez les lignes de scripts utilisées pour répondre.
- Copiez les sorties logicielles strictement nécessaires pour répondre.

- Utilisez une police type `courrier` pour les entrées et sorties de R.
- Proposez des graphiques si nécessaire.
- Proposez un bref commentaire en français.

2 Pulse rate, before and after exercise

Le fichier `pulseRate.txt` contient des données mises à disposition par John Eccleston et Richard Wilson de l'Université du Queensland (Australie). Des étudiants d'un cours de statistique ont participé à une expérience simple. Ils doivent prendre leur pouls puis faire une minute d'exercice (courir sur place) ou rester assis. Les groupes sont constitués à pile ou face. Chacun reprend son pouls après cette minute. On dispose également de quelques informations sur les étudiants :

Variable	Description
taille	Taille en cm
masse	Masse en kg
age	Âge en années
sexe	Sexe - H : Homme, F : Femme
fume	Fumeur - O : Oui, N : non
OH	Buveur régulier d'alcool - O : Oui, N : non
sport	Exercice physique - B : beaucoup, M : un peu, P : peu
court	Action pendant la minute de test - 1 : courir, 2 : s'asseoir
pulse1	Première mesure de pouls
pulse2	Seconde mesure de pouls

1. Importez le jeu de données dans R.
2. Identifiez le contexte statistique : nombre d'individus, population, nombre et type de variable... Quelle est l'utilité du groupe qui reste assis ?
3. Réalisez un tri à plat : il s'agit de résumer chaque variable par des indicateurs [1], un tableau si nécessaire et (au moins) un graphique adapté au type de chaque variable. Proposez un bref commentaire sur le jeu de données. Vous traiterez l'âge comme une variable quantitative discrète (pourquoi peut-on vous proposer cette consigne ?). Dans cette partie les variances et écart-types ne doivent pas être estimés.
4. Étudiez la liaison entre la pratique d'un exercice physique et la consommation d'alcool dans ce groupe d'étudiants. On attend un tableau, un test, un graphique et un commentaire.
5. Créez une nouvelle variable $deltapulse = pulse2 - pulse1$. Cette étape est nécessaire pour prendre en compte que les deux mesures `pulse2` et `pulse1` ne sont pas indépendantes. Dans le groupe ayant couru, la pratique d'un sport explique-t-elle des différences de `deltapulse` ? Proposez la moyenne et l'écart-type estimé pour chaque sous-groupe, un graphique pour comparer les groupes et plus si affinité (calcul de η^2).
6. Que pensez-vous du lien entre `deltapulse` et la masse des individus ?
7. Étudiez le lien entre le poids et la taille (Il s'agit en fait du sujet du TDR5).

1. paramètres de forme y compris

TDR5

TD R - 5
Régression linéaire simple

Vincent PAYET

"R, c'est super" — un étudiant, après la révélation

1 Introduction

On vous propose quelques rappels de cours sur le croisement de deux variables qualitatives et deux applications :
- Comment calculer séparément les éléments d'une régression linéaire ?
- Comment faire et représenter la régression linéaire avec les fonctions R ?

2 La régression en quelques lignes

La régression linéaire simple permet d'étudier une liaison **linéaire** entre deux variables quantitatives. Il faut donc vérifier que le nuage de points est globalement étendu selon une droite. On peut représenter le nuage de points avec la fonction `plot()`.

Une des variables est dite contrôlée et permet d'expliquer l'autre. Avec des notations classiques de mathématique on notera X la variable explicative et Y la variable à expliquer.

On cherche donc la meilleure droite passant par l'ensemble des points. Cette droite est de la forme $y = ax + b$. On veut donc définir a et b. On appelle $\hat{y}_i$ la valeur sur la droite pour un x_i donné, on parle de valeur prédite. On appelle résidu et on note ϵ_i (lire epsilon i) l'écart entre la valeur observée de y_i et la valeur sur la droite :

$$\epsilon_i = \hat{y}_i - y_i \tag{1}$$

On peut alors construire un modèle statistique qui permet d'exprimer y_i en fonction de x_i sous la forme générale *réponse = modèle + erreur* :

$$y_i = ax_i + b + \epsilon_i \tag{2}$$

ϵ_i est la partie aléatoire associée au modèle, ou l'erreur du modèle. On pose comme hypothèse que les résidus suivent une loi normale centrée et d'écart-type σ_{res} : $\epsilon_i \sim N(0; \sigma_{res})$. Le modèle de la régression linéaire a trois paramètres :

- la pente a
- l'ordonnée à l'origine b
- la variance résiduelle σ^2_{res}

On peut mesurer la qualité de la corrélation par le coefficient de corrélation r, dit coefficient de Pearson (pour le différencier d'autres coefficients de corrélation). On peut ensuite mesurer la qualité du modèle avec le coefficient de détermination R^2 qui exprime la part de variation expliquée par le modèle linéaire. Par ailleurs, la qualité du modèle peut aussi être évaluée en étudiant la distribution des résidus. L'étude des résidus se fait classiquement par des graphiques :

- La normalité avec un histogramme et surtout avec une droite de Henry (`qqplot()`)
- la répartition avec un nuage de points représentant les résidus standardisés en fonction des valeurs prédites ($\hat{y}_i$). Les résidus standardisés sont les résidus divisés par l'écart-type résiduel σ_{res}.

2.1 Les calculs

La pente :

$$\hat{a} = \frac{cov(x,y)}{s_x^2} \tag{3}$$

L'ordonnée à l'origine :

$$\hat{b} = \bar{y} - \hat{a}\bar{x} \tag{4}$$

Le coefficient de corrélation :

$$r = \frac{cov(x,y)}{s_x s_y} \tag{5}$$

Il faut ensuite calculer les i valeurs prédites et les i résidus pour trouver σ^2_{res}. La figure 1 résume la situation.

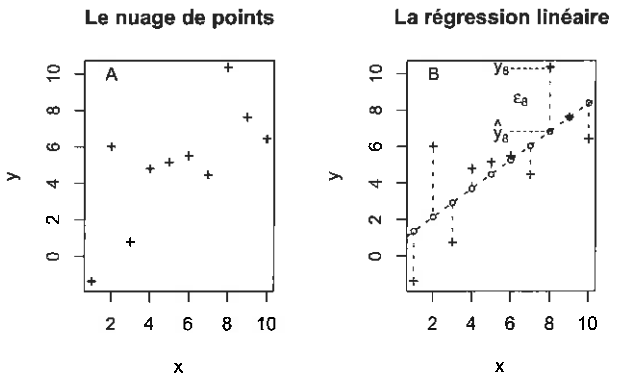


FIGURE 1 – Un nuage de points (A) et la régression linéaire (B)

3 Exercice 1. Applications des calculs

Nous allons faire les calculs en détail avec le logiciel à partir de 10 points suivants à peu près l'équation $y = x$ avec une erreur normale :

```
> x <- 1:10          # nos x contrôlés
> y <- x+2*rnorm(x) # nos y avec une erreur autour de y=x
> plot(x,y,pch="+")
```

- Représenter le nuage de points (`plot()`).
- Peut-on faire une régression linéaire ?
- Calculez les paramètres de la régression : a, b. La covariance se calcule avec `cov()`. Dans R on écrira par exemple : `a <- cov(x,y)/var(x)`.
- Calculez les valeurs prédites par le modèle, $\hat{y}_i$ et les résidus ϵ_i.
- Calculez le coefficient de corrélation r
- Calculez le coefficient de détermination R^2 de deux façons. Quelle est la part de variation expliquée par le modèle ?
- Calculez la variance des résidus σ_{res}
- Ajoutez une droite sur le graphique avec `abline()`
- Représenter les résidus standards en fonction des valeurs prédites (figure 2).

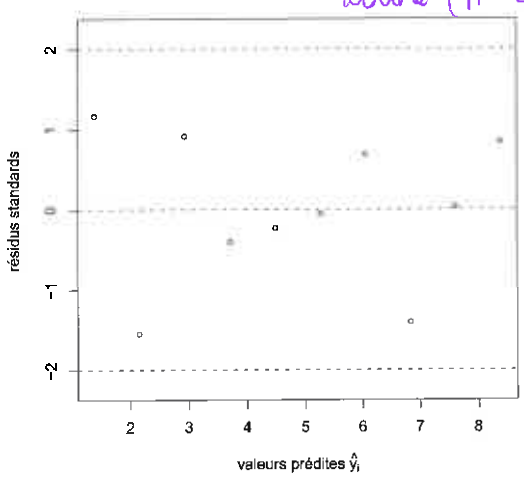


FIGURE 2 – Analyse des résidus

4 La fonction `lm`

On peut commencer par l'aide : `?lm`

```
Description:
```

```
'lm' is used to fit linear models.  It can be used to carry out
regression, single stratum analysis of variance and analysis of
covariance (although 'aov' may provide a more convenient interface
for these).

Usage:

lm(formula, data, ...
```

C'est bien la fonction qu'il nous faut. Comment rentrer une formule ? Avec le tilde $\sim$. Pour dire y en fonction de x, on écrit : $y \sim x$. Essayons la fonction :

```
> lm(y~x)                 # fonctionne mais renvoie peu d'informations

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     0.5933       0.7788
```

Identifiez ce que sont les coefficients `intercept` et `x`. On peut avoir plus de détails :

```
> modele1 <- lm(y~x)      # mettons le tout dans un objet
> summary(modele1)        # il y a plus d'informations

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-2.83218 -1.95768  0.02530  0.88656  3.76897

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.5933     1.6624   0.357   0.7304
x             0.7788     0.2679   2.907   0.0197 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.434 on 8 degrees of freedom
Multiple R-squared: 0.5137,     Adjusted R-squared: 0.4529
F-statistic: 8.451 on 1 and 8 DF,  p-value: 0.01968
```

Le summary renvoie la formule utilisée, un résumé des résidus en 5 valeurs, les coefficients estimés avec leur écart-type et un test de significativité et quelques autres valeurs. En particulier, le "Residual standard error" est l'écart-type estimé des résidus. Il n'est pas exactement égale au σ_{res} que vous avez calculez précédemment. Comparer cette valeur à $\sqrt{(SCE(residus)/(n-2))}$. Le $n-2$ est imposé pour respecter les degrés de libertés associés aux données. En revanche on retrouve le R^2 dans le summary.

En fait, `modele1` et `summary(modele1)` sont des listes d'objet dont voici les contenus :

```
> names(modele1)

 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

sqrt(var(e)*9/8)
R^2
summary(lm1)$sigma
names(summary(lm1))

plot(x,y)
plot(y~y, pch = "z")
abline(lm1, lty=4 col="red"
sd(e)

Chaque élément de lm1 est accessible
lm1$coefficients[2]

plot(x,y)
c'est comme
plot(y~x)
Droite -> abline(lm1)

```
> names(summary(modele1))

 [1] "call"          "terms"         "residuals"     "coefficients"
 [5] "aliased"       "sigma"         "df"            "r.squared"
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

On peut appeler chaque élément de `modele1` avec le dollar \$. On peut par exemple récupérer les résidus ou l'écart-type résiduel estimé :

```
> round(modele1$residuals,3) # résidus arrondis à 3 décimales

    1     2     3     4     5     6     7     8     9    10
-2.832  3.769 -2.250  0.991  0.572  0.134 -1.675  3.426 -0.083 -2.052

> summary(modele1)$sigma     # l'écart-type résiduel

[1] 2.433529
```

En général, les fonctions de R réalisent les calculs mais ne renvoient pas toute l'information à l'écran. Il faut aller chercher ce dont on a besoin.

On peut aussi représenter facilement la droite de régression avec la fonction `abline` avec un objet de type `lm` :

```
> plot(y~x)
> abline(modele1)
```

Enfin, on peut accéder au diagnostique des résidus :

```
> par(mfrow=c(2,2))
> plot(modele1)
```

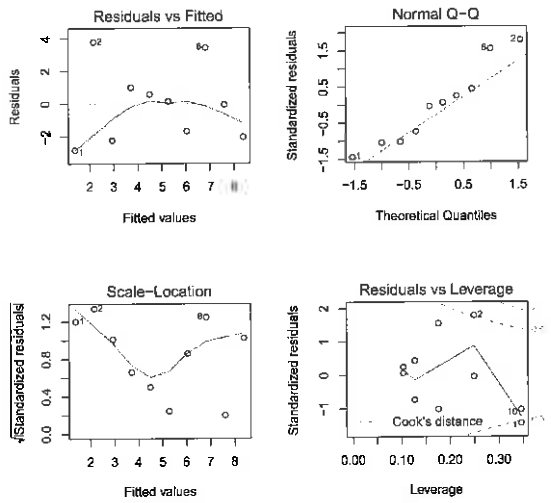


FIGURE 3 – Analyse des résidus par `lm()`

La fonction `plot` est une fonction générique. Selon l'objet qu'elle reçoit en entrée elle propose des graphiques différents, les mieux adaptés possibles aux

données, du point de vue du logiciel. Ici `plot(model)`, avec `model` un objet produit par `lm()`, renvoie quatre graphiques de diagnostique. Le premier est *presque* le graphique des résidus comme nous l'avons défini. Le second est une droite de Henry : plus les points sont alignés sur la droite en pointillés, plus les résidus suivent une distribution normale. Les deux suivants ne sont pas traités ici.

5 Le retour du pouls

La dernière question du TDR4 (pulse rate, *etc.*) est justement un croisement entre deux variables quantitatives. À vous de jouer... Vérifiez la forme du nuage avant toute chose et réalisez une régression sur des données suffisamment linéaires.

On exclut les valeurs extrêmes :

x ← Taille [Taille > 120]

Y ← Masse [Taille > 120]

Alexia MONSAVOIR
(43)

TD R - 2.1 & 2.1
Retour sur R
Échantillonnage – estimation

Vincent PAYET

Objectif : cette fiche propose un rappel sur les bases de R, une utilisation de la simulation pour illustrer le cours d'échantillonnage et l'introduction aux intervalles de confaince et aux tests statistiques avec R.

1 Introduction, retour sur R

R est un logiciel dédié à l'analyse de données, à la statistique et à la représentation graphique de données. C'est aussi un langage de programmation. Cela implique qu'il faut apprendre ce langage. Parmi les avantages de ce logiciel citons en deux :

- c'est un langage de programmation, ce qui permet de conserver une trace exhaustive de la démarche d'une analyse.
- c'est un logiciel libre, il est légalement téléchargeable et utilisable (et modifiable, même si cela vous concerne moins).

Pour installer R : `http://cran.r-project.org`

2 Xtrem pédagogie : R en 12 lignes

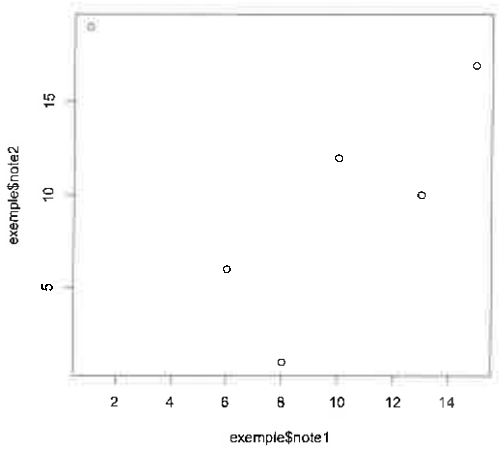
Il faut du temps pour apprendre un langage et il est probablement impossible de connaitre R intégralement avec ses milliers de modules complémentaires (nommés des *packages* , avec l'accent de Mr Bean). Toutefois, pour commencer à discuter de statistique avec votre ordinateur, il ne faut que quelques éléments de langage, le reste viendra avec la pratique. Voici le verbatim d'un échange de quelques commandes avec les commentaires. Les entrées sont précédées du symbol >, les sorties commencent par [1] (qui permet de connaitre l'indice de la valeur en début de ligne, pour les longues sorties). Les commentaires (parties non évaluées du programme) sont introduits par le symbole ♯.

Pour ces exemples, copiez le fichier de données `exemple.txt` sur le bureau.

```
> 40+2 # R comme une calcultarice
[1] 42
> x <- 6*7 # on peut stocker un résultat dans un objet
> z <- x+2 # et manipuler ces objets
> z # pour voir l'objet, équivalent à print(z)
```


2A_Stat_R_TD1 (2 ?)

```
[1] 44

> y <- c(1,4,6,7,10) # on peut stocker plusieurs valeurs dans un vecteur
> y*2 # et manipuler ce vecteur

[1]  2  8 12 14 20

> mean(y) # éventuellement avec des fonctions statistiques

[1] 5.6

> y[2] ; y[1:3]; y[-1] # On peut accéder par les index à une partie d'un objet

[1] 4

[1] 1 4 6

[1]  4  6  7 10

> # la fonction read.table sert à importer des données
> exemple <- read.table("exemple.txt",header=TRUE)
> exemple[,3:4] # on peut accéder aux lignes et/ou aux colonnes par les indices

  note2 groupe
1    12      1
2    17      1
3     1      2
4     6      2
5    10      3
6    19      3

> exemple$noms; exemple$Noms # on peut aussi accéder aux colonnes avec le symbole \$

NULL

[1] alf joe foo bar tim hal
Levels: alf bar foo hal joe tim

> # notez que Noms et noms ne sont pas équivalent. La casse compte.
> plot(exemple$note1,exemple$note2) # la fonction plot est une fonction graphique
```



Les noms suivis de parenthèse sont des fonctions. Entre paranthèses on trouve les paramètres des fonctions.

Il y a aussi quelques bonnes pratiques à adopter rapidment. Elles sont explicitées dans le TD®1. La liste est la suivante :

- Définir un répertoire de travail (menu fichier, changer le répertoire courant)
- Uiliser des scripts pour conserver une trace de vos ananlyses, et de vos apprentissages
- Consulter l'aide pour les nouvelles fonctions avec le symbole ?, par exemple : `?mean`
- Constituer un glossaire des fonctions de bases vues pendant les TD.

3 Échantillonnage

La question est "que peut on attendre d'un échantillon ?". Votre cours vous a donné la théorie. Que peut-on vérifier ? la fonction `sample()` permet de faire un échantillon en partant d'une population :

```
> population <- 1:50
> sample(population,5) # vous pouvez jouer au loto..

[1] 27 43  5 40 35

> # avec ou sans répétition ?
> sample(population,20, replace=TRUE)

 [1]  8  4 22  3 24 38 45 28 24 29 18  4 14 43 43 31 30  9  6 17
```

Nous allons pouvoir faire des simulations de processus aléatoire avec cette fonction. Par exemple, si une variable ne suit pas une loi normale mais que les échantillons sont d'effectifs suffisant, on considère que la loi de la moyenne tend vers une loi normale. Peut-on le vérifier ?

3.1 Les lois de probabilité

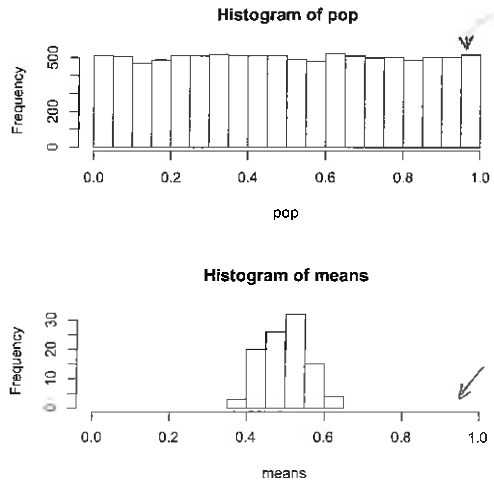
Avant cela, comment manipuler des lois ? Plusieurs lois sont utilisables, notamment la loi normale (`norm`) et la loi uniforme (`unif`). Pour une loi, on peut obtenir la fonction de répartition (p), les quantiles (q), la densité (d) ou un échantillon (r). Il faut pour cela écrire le nom de la fonction avec la lettre p, q, d ou r et le nom de la loi :

```
> pnorm(1.96) # quelle est la probabilité P(X<1.96) si X normale ?

[1] 0.9750021

> rnorm(5)  # 5 valeurs tirées au hasard pour un variable aléatoire normale

[1] -0.5659812  1.1641740 -0.7478144  0.1549822  0.1066874
```

3.2 Distribution d'échantillonnage de la moyenne

Soit X une variable aléatoire uniforme (par défaut entre 0 et 1). Nous allons créer une population de 10000 individus puis réaliser 100 échantillons de 30 individus. La comparaison des histogrammes permet de voir que la loi de la moyenne.

```
> pop <- runif(10000)
> means <- NULL
> # 100 échantillons de 30 :
> for (i in 1:100) {
+ means[i] <- mean(sample(pop,30))
+ }
> sd(pop); mean(pop)

[1] 0.2889423

[1] 0.5007644

> sd(means); mean(means)

[1] 0.05293667

[1] 0.5047566

> par(mfrow=c(2,1))
> hist(pop)
> hist(means,xlim=c(0,1))
```

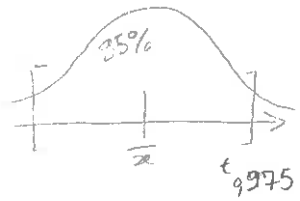


Comparez les moyennes puis les écart-types (`sd()` calcul l'écart-type estimé) de la population puis de la distribution des moyennes. Quelle est la formule du cours ?

la boucle `for` est connue et similaire à ceux que vous connaissez par ailleurs. R permet des écritures plus condensée. Essayer cette écriture à la place de toute la boucle :

```
> hist(replicate(100,mean(sample(pop,30) )),xlim=c(0,1))
```

Exercice : Refaire la même simulation avec une population d'origine suivant une loi normale.



$\hat{\sigma} = \sqrt{\frac{n}{n-1}} s_x$

$\hat{\sigma}^2 = \frac{n}{n-1} s_x^2 = \frac{n}{n-1} \frac{SCE}{n} = \frac{SCE}{n-1}$

4 Estimation

La connaissance d'un échantillon permet de calculer des valeurs ponctuelles d'estimation de paramètres de la, population. Rappelez les estimateurs ponctuel de la moyenne de la variance et d'une proportion. Quelle valeur renvoie la fonction `var()` ?

Ces estimations doivent si possible être accompagnée par des estimations par intervalle de confiance.

4.1 Intervalle de confiance

Pour une population de variance connue σ. l'écart-type de la variable moyenne, appellée erreur-standard vaut $\frac{\sigma}{\sqrt{n}}$. L'intervalle de confiance à 95% vaut :

$IC_{95\%} = \bar{x} + / - t_{0.975} \frac{\sigma}{\sqrt{n}}$

On obtient les valeurs des quantiles de la lois normales :

```
> qnorm(0.975)
[1] 1.959964
> qnorm(0.025)
[1] -1.959964
```

Exercice : Dans un centre de soin, on a mesuré la taille de 57 jeunes de 18 à 20 ans. On a trouvé $\bar{x} = 175$ et $s^2 = 27,5$. On veut estimer la taille des jeunes dans la population. Proposez une estimation par intervalle de confiance à 95% puis à 99%.

4.2 Une fonction pour estimer un IC

On pourrait réaliser une fonction qui retourne les bornes de l'intervalle de confiance. Une fonction est introduite ainsi :

```
> UnNom <- function(parametre1,parametre2){
+   tx <- rameter1/(parametre1+parametre2)
+   print(tx)
+   }
```

parametre

Exercice :

1. Que fait la fonction UnNom ?
2. Renommer là judicieusement.
3. Écrire une fonction pour calculer un intervalle de confiance. Le niveau de confiance est à entrer en paramètre.
4. Si la variance n'est pas connue, que devient la formule ? Comment modifier votre fonction pour prendre en compte ce cas ?

5 application

On a relevé les diamétres en cm de 15 arbres sur une parcelle d'agroforesterie :

```
c(9  8 11 10  8  6 11  9 10  9 12  8  6  9 14)
```

1. Donner une estimation de la moyenne et de la variance des arbres de la parcelle.
2. Donner une estimation par intervalle de confiance à 95% et 99%.
3. Si on suppose que le diamèter suit une loi normale, quelle est la probabilité de trouver un arbre avec un diamèetre supérieur à 15 cm ?

TDR2.2

TD R - 2.2
Tests statistiques

Vincent PAYET

Objectif : Rappels sur quelques tests statistiques communs, réalisation de ces tests avec R, les calculs et figures associés à ces tests.

Table des matières

Dans un contexte expérimental, il est classique de comparer les résultats de mesures réalisées sur deux sous-groupes (on parle d'échantillons). Il s'agit généralement d'appliquer un traitement différent à chaque échantillon et de suivre sur chaque individu une variable considérée comme une réponse. Même en cas de non action du traitement, il n'est pas raisonnable d'attendre des valeurs identiques dans chaque échantillon. On peut identifier plusieurs sources de variation : la variabilité inhérente au système considéré (notamment pour des expériences sur du vivant), les facteurs non contrôlés, la fluctuation d'échantillonnage, les imprécisions de la mesures...

Dans cette situation comment établir qu'une différence observée résulte de l'effet du traitement et non de ces autres sources de variation ? Les tests statistiques permettent d'aborder ces situations.

Les tests statistiques nécessitent de définir un couple d'hypothèses (de référence et alternative, nommées respectivement H0 et H1), un niveau de risque acceptable si on décide de rejeter l'hypothèse H0 (le risque de première espèce α) et une variable de test dont on connait le comportement si H0 est vraie. Il est nécessaire de conclure statistiquement (conservation ou rejet de H0) puis concrètement vis-à-vis du problème posé.

1 Comparaisons de moyennes

Contexte

Pour comparer deux échantillons en terme de moyenne, en particulier pour tester l'égalité des moyennes μ_1 et μ_2 on considère les hypothèses $H_0 : \mu_1 = \mu_2$ contre $H_1 : \mu_1 \neq \mu_2$ (ou $\mu_1 > \mu_2$ ou $\mu_1 < \mu_2$ en versions unilatérales).

La variable du test est :

$t = \frac{\mu_1 - \mu_2}{\sigma_{\Delta m}}$ avec $\sigma_{\Delta m}$ l'écart-type de la variable $\mu_1 - \mu_2$ qui dépend des informations sur les échantillons (variances connues ou non, effectifs...).

Le test est appelé couramment test-t ou test de student. Il s'agit d'un test paramétrique, c'est-à-dire que l'on fait des suppositions sur la distribution des données pour construire le test. En l'occurrence il faut vérifier que les données de chaque échantillon suivent une loi normale et que les variances des deux groupes sont comparables.

La démarche de la comparaison de moyennes est la suivante :

1. Description des données par des paramètres et des graphiques.
2. Vérification de la normalité.
3. Vérification de l'homogénéité des variances.
4. Test des moyennes.

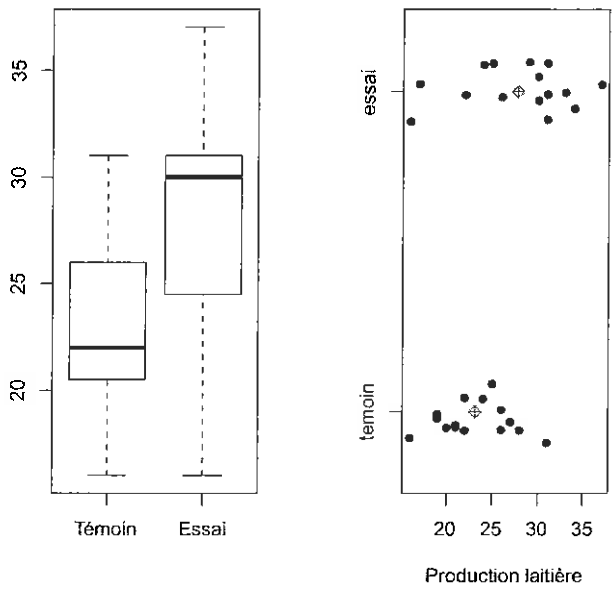
Un premier exemple

On suit la production laitière de deux lots de vaches élevées dans des conditions témoins pour un lot et dans des conditions devant améliorer leur production pour un second lot.

```
> essai <- c(22,37,31,17,31,16,29,31,30,33,24,26,34,30,25)
> temoin <- c(19,28,25,24,31,16,21,19,20,26,22,22,27,21,26)
> mean(essai);sd(essai)

[1] 27.73333

[1] 6.017435

> mean(temoin);sd(temoin)

[1] 23.13333

[1] 4.033196
```

On veut tester si les deux moyennes sont comparables afin de savoir si les conditions du lot essai a permis d'augmenter la production laitière. Commençons par regarder les données :

```
> par(mfrow=c(1,2))
> boxplot(temoin, essai,names=c("Témoin","Essai"))
> stripchart(c(temoin,essai)~gl(2,15), method="jitter",
+            pch=16, group.names=c("temoin", "essai"),
+            xlab="Production laitière")
> points(mean(essai),2,col="red",pch=9)
> points(mean(temoin),1,col="red",pch=9)
```

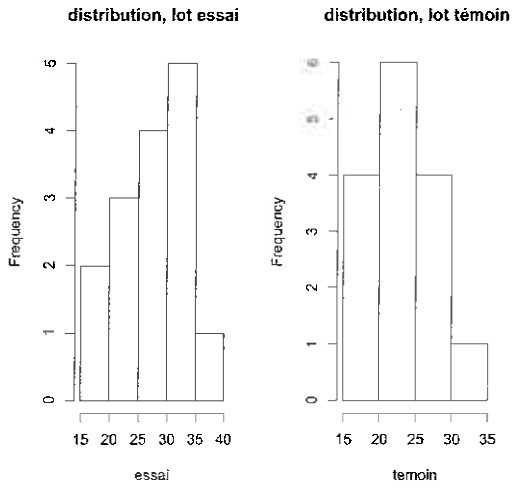


La production de lait du lot essai semble supérieure mais les deux distributions sont recoupantes et nous ne disposons pas de beaucoup de données. Nous nous pouvons répondre à la question sérieusement sans tests statistiques.

La normalité

Pour la normalité nous nous contenterons dans un premier temps d'une analyse graphique. Il faut bien regarder la normalité des deux lots séparément. Dans le cas où H1 est fortement vraie la distribution de l'ensemble des données sera non normale et bimodale !

```
> par(mfrow=c(1,2))
> hist(essai, main="distribution, lot essai")
> hist(temoin, main="distribution, lot témoin")
```



Les distributions sont globalement unimodale et en cloche. La normalité est importante mais les tests sont robustes à un certain écart à la normalité. Nous pouvons continuer.

Les variances

Le test F fait le rapport des variances estimées et compare le résultat à une variable de Fisher à $(n_1 - 1; n_2 - 1)$ ddl. La plus grande variance doit être au numérateur. L'hypothèse nulle suppose l'égalité des variances (le rapport vaut 1).

```
> var(essai);var(temoin)
[1] 36.20952
[1] 16.26667
> var(essai)/var(temoin)
[1] 2.225995
> qf(0.95,14,14)
[1] 2.483726
```

Le F calculé est inférieur au F théorique. On conserve l'hypothèse d'égalité des variances.

La comparaison de moyenne

On est dans le cas de variables normales (on a accepté cette hypothèse), de variances inconnues et avec de petits effectifs. Il faut estimer la variance commune $\widehat{\sigma^2}$:

$\widehat{\sigma^2} = \frac{n_1 s_1^2 + n_2 s_2^2}{n_1 + n_2 - 2}$

On a aussi besoin de la variance calculée d'un échantillon. On crée pour cela une fonction `varp()`. On réalise ensuite le calcul comme à la calculatrice :

```
> varp <- function(x){var(x)*(length(x)-1)/length(x)}
> n1 <- length(essai)
> n2 <- length(temoin)
> var1 <- varp(essai)
> var2 <- varp(temoin)
> mu1 <- mean(essai)
> mu2 <- mean(temoin)
> sigma <- (n1*var1+n2*var2)/(n1+n2-2)
> tcalc <- (mu1-mu2)/sqrt(sigma/n1+sigma/n2) ; tcalc
[1] 2.459361
> qt(0.95,n1+n2-2)
[1] 1.701131
```

Le t calculé est supérieur au t théorique : on peut rejeter H0 avec un risque de se tromper d'au plus 5%. Quelle est la probabilité d'observer une différence de moyennes au moins aussi grande ? On cherche $P(\mu_1 - \mu_2 > 2,45936)$

```
> 1-pt(2.45936,n1+n2-2)
[1] 0.01017900
> # Le risque est bilatéral :
> (1-pt(2.45936,n1+n2-2))*2
[1] 0.02035801
```

Avec un bon niveau de confiance, nous pouvons affirmer que les conditions expérimentales téstées améliorent la production laitière.

Le test (Student)

La fonction `t.test()` permet de réaliser le test. Retrouvons nous les valeurs calculées "à la main" (notamment t, ddl et p-value) ?

```
> t.test(essai, temoin)

        Welch Two Sample t-test

data:  essai and temoin
t = 2.4594, df = 24.466, p-value = 0.02135
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.7435654 8.4564346
sample estimates:
mean of x mean of y
 27.73333  23.13333
```

Dans l'aide de la fonction `?t.test`, on note plusieurs paramètres intéressants. On peut notament présiser si les variances sont consiérées commes égales :

```
> t.test(essai, temoin, var.equal=TRUE)

        Two Sample t-test

data:  essai and temoin
t = 2.4594, df = 28, p-value = 0.02036
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.7686502 8.4313498
sample estimates:
mean of x mean of y
 27.73333  23.13333
```

C'est mieux, n'est-ce pas ? Le test par défaut, avec `var.equal=FALSE` est le test de Welch qui permet de comparer deux groupes de variances différentes. On peut aussi préciser une valeur `mu` pour comparer une moyenne à une valeur ou préciser s'il s'agit de données appariées (avec `paired`).

2 Comparaisons de variances

Contexte

H0 : les variances des populations sont égales, $F = \frac{\widehat{\sigma_1^2}}{\widehat{\sigma_2^2}} = 1$.

Le test

La fonction `var.test()` permet de réaliser le test. Dans l'aide de la fonction on remarque un paramètre `alternative` permettant de spécifier si le test est uni- ou bilatéral.

Exemple

Avec l'exemple de la production laitière, le test s'écrit ainsi :

```
> var.test(essai, temoin, alternative="greater")

        F test to compare two variances

data:  essai and temoin
F = 2.226, num df = 14, denom df = 14, p-value = 0.07325
alternative hypothesis: true ratio of variances is greater than 1
95 percent confidence interval:
 0.8962323       Inf
sample estimates:
ratio of variances
          2.225995
```

Retrouvez les valeurs calculées précédemment. À quoi correspond la p-value ? Retouvez cette valeur.

3 Comparaisons de proportions

Le test

La fonction `prop.test()` permet de réaliser un test d'égalité de proportion. Pour les proportions, pensez aussi au test du χ^2 (Chi deux). Une fiche est disponible sur le cours de première année pour le χ^2.

4 Quelques autres tests

Pour tester la normalité : `shapiro.test()`

Pour des données non normales, test non paramètrique de comparaison de médiane : `wilcox.test()`

5 Exercices

Pour ces exercices, vous devez importer les fichiers de données à récupérer sur e-campus.

Exercice 1 On veut comparer le rendement en blé de deux parcelles travaillées avec ou sans labour. Données : Tillage.txt

```
> till <- read.table("Tillage.txt",h=T)
> dim(till)

[1] 148   2

> names(till)

[1] "culture"   "rendement"

> culture <- till$culture
> rdt <- till$rendement
> # calculons les moyennes
> tapply(rdt,culture, mean)

       L       NL
33.13514 29.59459
```

Les données sont ici fournies de façon "classique" en statistique : un tableau individus-variables. Combien de variables ici ? Combien d'individus ?

La fonction `tapply()` permet d'appliquer une fonction à une variable quantitative découpée par une variable qualitative.

Pour vous aider : plusieurs fonctions (`boxplot`, les tests aussi) acceptent une écriture du genre `rdt~culture` qui signifie `rdt` en fonction de `culture`. Testez `boxplot(rdt~culture)` pour voir.

Proposez une conclusion pour ces deux lots.

Exercice 2 On dispose de quatre groupe de patients d'effectifs variables et pour chaque groupe on a compté les fumeurs. On se demande si la proportion de fumeurs dans les populations dont proviennent les patients sont les mêmes ? (Données Fleiss (1981), p. 139).

```
> fumeurs  <- c( 83, 90, 129, 70 )
> patients <- c( 86, 93, 136, 82 )
```

Exercice 3 Si vous avez du temps, mettez en forme dans un fichier de traitement de texte vos deux exercices précédents.

TD R - 2.3
ANOVA

Vincent Payet

Objectif : Rappels sur l'analyse de la variance, réalisation avec R.

Considérons des données pour lesquelles nous voulons expliquer une réponse quantitative (une mesure, un comptage...) par une ou plusieurs variables qualitatives (des facteurs). La question sous-jacente pour un seul facteur est : "le facteur A modifie t'il la réponse" ? Pour deux facteurs (et plus), on étudie l'effet de chaque facteur et leurs éventuelles interactions sur la réponse. Le cas le mieux décrit est celui où les facteurs modifient la réponse en moyenne mais non en variance. Pour traiter ce genre de situation, nous utiliserons une procédure appellée ANOVA : *ANalysis Of VAriance*.

L'ANOVA a été mise au point par Ronald Aymler Fisher (1925) et rapidement appliquée à des problèmes d'agronomie. Le principe de l'ANOVA est de scinder la variance totale d'une série de donnée en plusieurs sources de variation afin de comparer les moyennes de populations de plusieurs groupes. Il s'agit donc d'un outil de comparaison de k moyennes, (en général $k \geq 3$). Les sources de variations sont en général l'effet du (ou des) facteur(s) et une composante non expliquée par le facteur appelée erreur ou résidus.

l'ANOVA est aussi utilisée pour valider des modèles (exemple : modèle de régression multiple) et pour calculer des estimations des différentes composantes de la variances (exemples : évaluation de la capabilité, génétique quantitative...).

1 Le modèle

On décrit la relation entre la mesure y_{ij} du $j^{ième}$ individu du groupe i et la variable qualitative par le modèle suivant :

$$y_{ij} = a_i + \epsilon_{ij}$$

Avec a_i la moyenne des réponses du groupe i et ϵ_{ij} l'erreur (le résidus). On attribue donc à chaque individu la moyenne du groupe auquel il appartient. En considérant la moyenne globale des réponses que l'on note μ et l'effet de chque modalité du facteur α_i, on peut écrire le modèle de façon plus classique :

$$y_{ij} = \mu + \alpha_i + \epsilon_{ij}$$

La procédure d'ANOVA revient alors à tester les hypothèses suivantes :

1. H_0 : les a_i sont égaux à μ quelque soit i

2A_Stat_R_TD3

2. contre H_1 : Au moins une moyenne est différente

ou encore :

1. H_0 : les α_i sont ~~égaux~~ nuls quelque soit i
2. contre H_1 : Au moins un effet est différent de zéro

2 le code R pour l'ANOVA

Il existe plusieurs fonctions capables de réaliser la procédure d'anova dont la fonction `anova()`. L'aide de cette fonction est inhabituellement succinte (il n'y a même pas d'exemple).

Les données doivent être présentée sous la forme d'un tableau " individus-variables". Appelons `rep` la variable quantitative à expliquer et `fac` le facteur explicatif. On réalise l'ANOVA ainsi :

```
anova(lm(rep~fac))
```

`lm()` crée un objet de type modèle linéaire entre `rep` et `fac`. C'est sur cet objet que l'ANOVA est réalisée. Cet usage souligne la grande similitude entre l'équation du modèle de l'ANOVA et une équation de modèle linéaire. Le symbole ~ indique le sens de la relation : on étudie `rep` en fonction de `fac`.

Sur la même base on peut écrire le code pour une anova à deux facteurs :`fac1` et `fac2` :

```
anova(lm(rep~fac1+fac2))
```

Avec interaction ?

```
anova(lm(rep~fac1*fac2))
```

Trois facteurs ?

```
anova(lm(rep~fac1*fac2*fac3))
```

etc... On obtient donc facilement un tableau d'ANOVA. Il ne faut toutefois pas oublier le contexte et les étapes nécessaire pour réaliser une ANOVA complétement c'est à dire en vérifiant les conditions nécessaires et en obtenant une conclusion utile. Les étapes sont les suivantes :

1. Calcul de moyenne et variance par groupe
2. Représentation graphique des données
3. Normalité
4. Homoscédasticité
5. Anova
6. Analyse des résidus
7. Test de comparaison multiple de moyenne

3 Un exemple complet

3.1 Importer les données

Récupérez le fichier exo1.txt et importez-le :

```
> data <- read.table("exo1.txt",h=T,dec=",")
> dim(data)

[1] 15  2

> names(data)

[1] "Poids"   "elevage"

> summary(data)

     Poids          elevage
 Min.   :42.10   Min.   :1
 1st Qu.:43.90   1st Qu.:1
 Median :48.30   Median :2
 Mean   :49.80   Mean   :2
 3rd Qu.:54.65   3rd Qu.:3
 Max.   :59.10   Max.   :3
```

On dispose de 15 individus et deux variables. La variable elevage est un facteur codé en numérique. Il faudra la forcer comme un facteur.

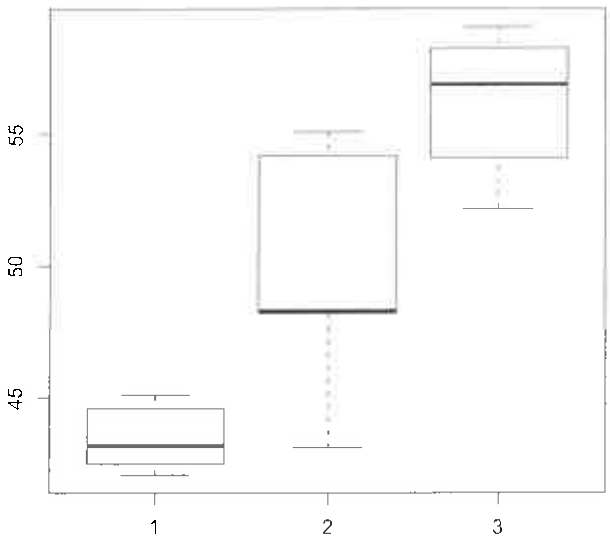
3.2 Calcul de moyenne et variance par groupe

La fonction `tapply()` permet d'appliquer une fonction (`mean`, par exemple) à une série de donnée découpée par un facteur :

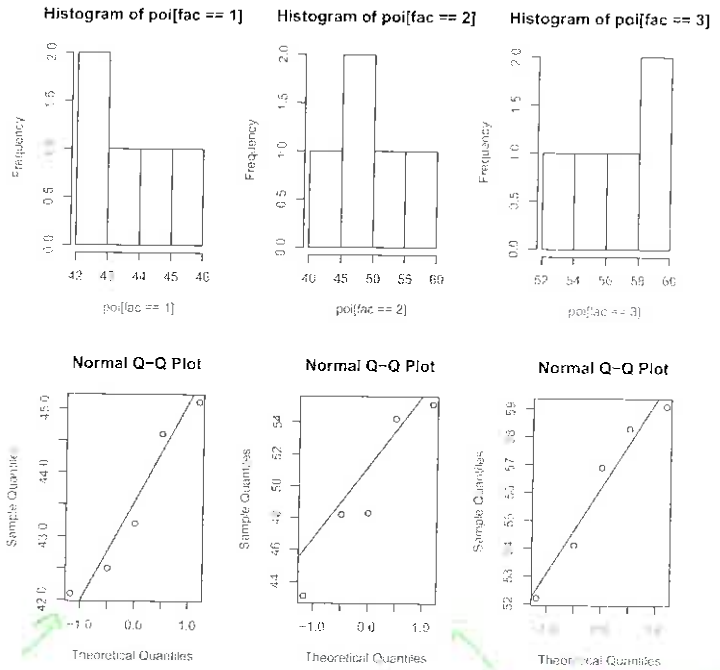
```
> poi <- data$Poids
> fac <- as.factor(data$elevage)
> tapply(poi,fac,mean)

    1     2     3
43.50 49.78 56.12

> tapply(poi,fac,var)

     1      2      3
 1.705 24.287  8.422

> table(fac)

fac
1 2 3
5 5 5
```

la fonction `table()` renvoie un tableau de contingence. Le plan est équilibré mais nous disposons de petits jeu de données.

3.3 Représentation graphique des données



3.4 Normalité



Pour des petits jeux de données, les qq-plot permettent de visualiser la normalité bien plus efficacement que les histogrammes. Si les données sont normales, les points doivent être globalement le long de la ligne.

On peut éventuellement faire un test, pour chaque groupe :

```
$`1`

        Shapiro-Wilk normality test

data:  X[[1L]]
W = 0.917, p-value = 0.5107


$`2`

        Shapiro-Wilk normality test

data:  X[[2L]]
W = 0.9142, p-value = 0.493


$`3`

        Shapiro-Wilk normality test

data:  X[[3L]]
W = 0.9308, p-value = 0.6019
```

D'après vous quelle est l'hypothèse nulle de ces tests ? Que concluez-vous ?

3.5 Homoscédasticité

```
        Bartlett test of homogeneity of variances
```

35. Homoscédasticité
Barklet

```
data:  poi and fac
Bartlett's K-squared = 5.2747, df = 2, p-value = 0.07155
```

si les variances sont pas homogènes, on doit peut changer les valeurs (les mettre en log par ex)

On conserve H_0, les variances sont homogènes.

3.6 Anova

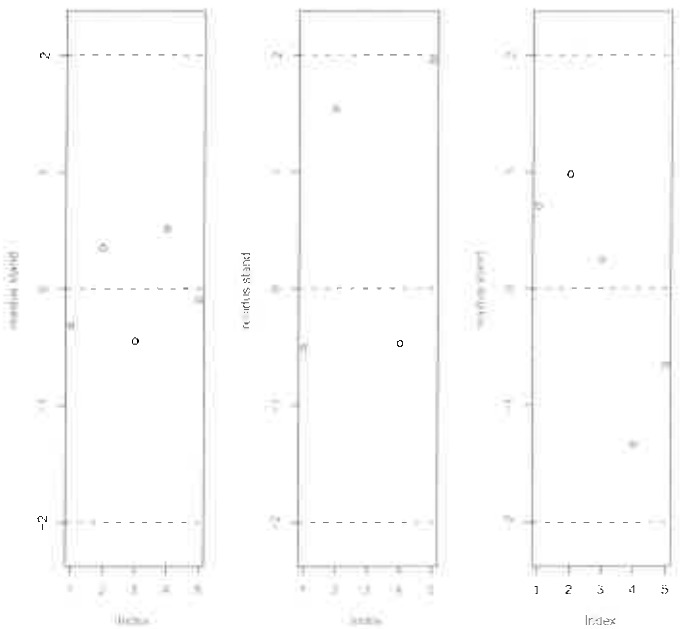
```
Analysis of Variance Table

Response: poi
          Df Sum Sq Mean Sq F value    Pr(>F)
fac        2 398.16 199.082  17.355 0.0002875 ***
Residuals 12 137.66  11.471
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Y-a-t'il un effet de l'élevage ?

→ ppds : comparaison multiple de moyennes

3.7 Analyse des résidus

```
$`1`
NULL

$`2`
NULL

$`3`
NULL
```



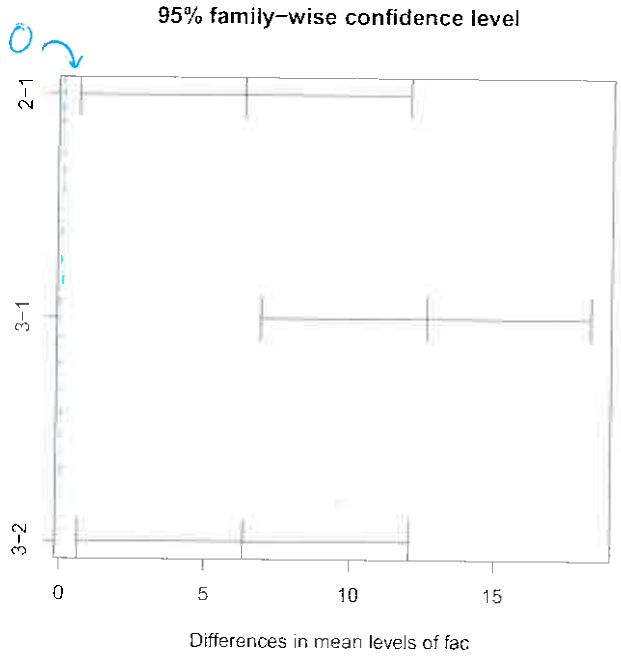
résidus entre -2 et 2 car si tout ça bien euh... ça suit la loi normale ?

Les résidus standardisés doivent se trouver principalement (à 95%) entre -2 et 2.

3.8 Test de comparaison multiple de moyennes

Le test permet de tester toutes les combinaisons de moyennes deux à deux. Lorsque l'intervalle des différences estimée ne contient pas zéro, les groupes sont statistiquement différents.

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = poi ~ fac)

$fac
      diff       lwr      upr     p adj
2-1  6.28 0.5652061 11.99479 0.0313196
3-1 12.62 6.9052061 18.33479 0.0002008
3-2  6.34 0.6252061 12.05479 0.0297915
```



4 Exercice

Vous devez réaliser une ANOVA complète des données `iris` disponible dans R avec au moins une des variables quantitatives disponibles. Pour vous aider vous disposez d'un script à adapter : `anova1.r`. Vous devez réaliser un rapide compte-rendu de TD dans un document de type word en intégrant les entrées et sorties nécessaires, les graphiques et vos conclusions.

R TD3 7/05/2012

ANOVA → analyses des variances.

But final → comparer des moyennes de différents groupes

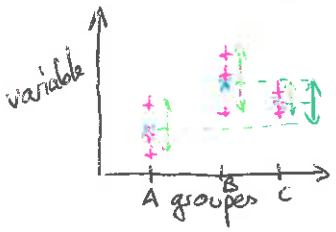
Données → la variable (une ou plusieurs) et les facteurs qui influent peut-être.

Question → Y a-t-il une influence du facteur ?

Pourquoi on analyse les variances ?

variance intra : variance entre les données d'un groupe

variance inter : variance entre les moyennes des groupes



↕ variance intra

× moyennes

↕ variance inter

H_0 → pas d'effet de facteur, toutes les moyennes sont égales

H_1 → au moins une moyenne diffère de(s) autre(s)

On étudie la normalité des groupes.

⚠ si on a peu de points → on peut pas tester la normalité

si on a bcp bcp de points → il suffit que ça soit unimodal.

facteur = quali systématique

↓

avec un facteur quanti tu fais une régression

TD R - 2.4
Régression simple et multiple

Vincent PAYET

"L'existence précède l'essence, mais de peu" — J.B BOTUL, 1929

Objectif : Rappels sur la régression linéaire simple et multiple, estimations ponctuelles et par intervalles de confiance des coefficients, sélection de variables et réalisation de tout ça avec R.

Table des matières

Introduction

Étudier le lien entre deux variables quantitatives x et y, c'est identifier ce qu'on peut savoir de la variable y en regardant l'autre variable, x. Souvent cette étude revient à rechercher une relation de type $y = f(x)$. Le cas le plus simple est de chercher une relation linéaire de type $y = bx + a$. On appelle cette relation une régression linéaire, simple quand on utilise un x unique et multiple si on utilise plusieurs régresseurs.

2A_Stat_R_TD4

1 Régression linéaire simple

Les calculs et l'approche descriptive de la RLS sont décrits dans la fiche TDR5.

1.1 Estimation d'un modèle

Le modèle de la regréssion c'est une droite et une erreur aléatoire ϵ (epsilon) :

$$y_i = bx_i + a + \epsilon_i \tag{1}$$

$$\text{avec} \quad \widehat{y_i} = bx_i + a$$

Faire la régression c'est d'abord estimer des valeurs de a et de b à partir d'observations. On note $\widehat{a}$ et $\widehat{b}$ les estimations. Il s'agit bien d'estimations dans la mesure où une répétition experimentale donnera des observations différentes, donc une droite différente et des coefficients différents.

L'aspect statistique de ce modèle réside aussi dans la distribution de l'erreur, des résidus. En effet on fait les hypothèses suivantes :

- $\epsilon_i \sim N(0; \sigma_{res})$
- Les ϵ_i sont indépendants.

On dit que les résidus sont indépendants et identiquement distribués (iid). Le modèle de la régression linéaire simple a trois paramètres : a, b et σ_{res}. La démarche est de calculer des valeurs pour ces paramètres puis de vérifier la plausibilité des hypothèses faites sur la distribution des ϵ pour valider le modèle.

De plus, on peut associer à ces estimations ponctuelles des intervalles de confiance et donc calculer d'une part un intervalle de confiance pour la droite de régression et d'autre part un intervalle de prédiction.

1.2 Un exemple en détail

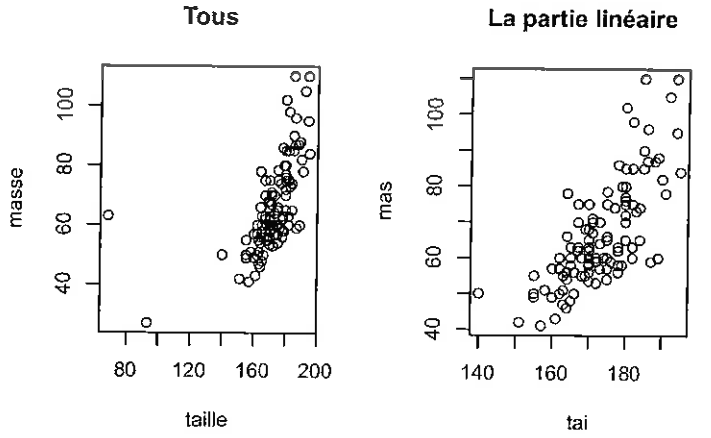
On importe un jeu de données pour étudier la relation entre le poids et la taille de quelques étudiants.

```
> pulse <- read.table("pulseRate.txt",header=TRUE)
> names(pulse)

 [1] "taille" "masse"  "age"    "sexe"   "fume"   "OH"     "sport"  "action"
 [9] "pulse1" "pulse2"

> dim(pulse)

[1] 109  10

> taille <- pulse[,1]
> masse  <- pulse[,2]
> par(mfrow=c(1,2))
> plot(masse~taille, main="Tous")
> tai <- taille[taille>130] *
> mas  <- masse[taille>130]
> plot(mas~tai, main="La partie linéaire")
```

→ masse en fct de la taille

NB : avant d'étudier statistiq^{ue} des données, on jète un oeil au graph

* on ne s'intéresse qu'aux individus de taille > 130



On veut étudier des données liées linéairement, on met donc de côté les deux petites tailles. Le nuage de points est déjà plus adapté à la regression linéaire. On peut passer aux estimations :

```
> lm1 <- lm(mas~tai)
> summary(lm1)

Call:
lm(formula = mas ~ tai)

Residuals:
     Min       1Q   Median       3Q      Max
-23.8116  -7.1624  -0.6126   5.7381  30.5267

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -121.17315   16.62910  -7.287 6.14e-11 ***
tai            1.08458    0.09581  11.320  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.03 on 105 degrees of freedom
Multiple R-squared: 0.5496,     Adjusted R-squared: 0.5454
F-statistic: 128.1 on 1 and 105 DF,  p-value: < 2.2e-16
```

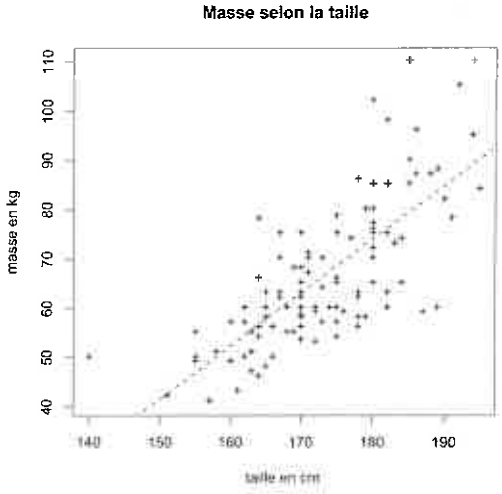
→ le tableau c'est comme le β (estimate), écart-type (std. Error), valeur t (t value)

→ écart type des résidus
→ R^2 multiple et R^2 ajusté

À partir de cette sortie logicielle, quel est la part des variations de la masse expliquée par les variations de la taille ? → 54,96 %

```
> plot(mas~tai, pch="+",xlab="taille en cm", ylab="masse en kg", main="Masse selon la taille")
> abline(lm1,lty=2)
```

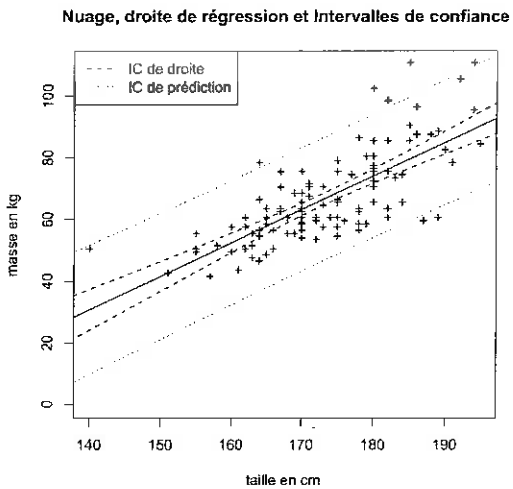
→ droite de régression



Votre cours indique des formules pour calculer des intervalles de confiance pour la droite et pour la prévision. Comment calculer et visualiser ces choses là ? On dispose d'une fonction `predict()`. Utilisée simplement, `predict(lm(mas ~ tai))` renvoie les valeurs $\widehat{y}_i$ sur la droite. En précisant des nouvelles valeurs de x on obtient les IC :

```
> plot(mas~tai, pch="+",xlab="taille en cm", ylab="masse en kg",ylim=c(0,110))
> new <- data.frame(tai = seq(130, 200, 0.5))
> ICprediction <- predict(lm(mas ~ tai), new, interval="prediction")
> ICdroite <- predict(lm(mas ~ tai), new, interval="confidence")
> matlines(new$tai,cbind(ICdroite, ICprediction[,-1]),
+          lty=c(1,2,2,3,3), type="l", ylab="predicted y",col=1)
> legend("topleft",lty=2:3,c("IC de droite","IC de prédiction"))
> title("Nuage, droite de régression et Intervalles de confiance")
```

création d'une séquence de 130 à 200 de pas 0,5

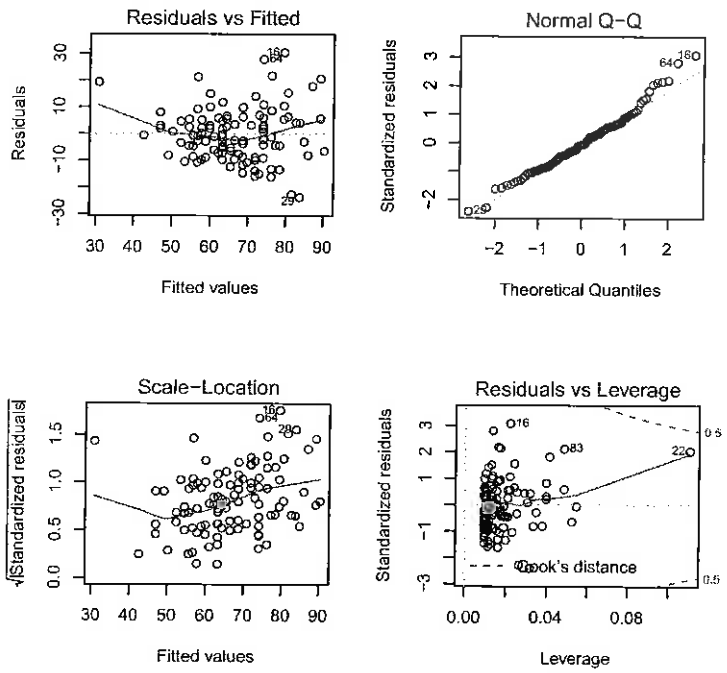


dataFrame : tableau de données, indispensable pour prédict

Ces intervalles de confiance sont des branches d'hyperbole. Plus on approche des limites du domaine d'étude moins l'estimation est certaine.

On obtient une analyse graphique des résidus avec la fonction `plot()` :

```
> par(mfrow=c(2,2))
> plot(lm1)
```



La régression est assez correcte. La dispersion des résidus augmente légérement avec les valeurs (au delà de 70) mais la tendance n'est pas très forte, les résidus sont normaux et il n'apparait pas de points infulents.

Globalement, on montre une relation assez forte mais un outil peu puissant pour les prédictions. Il serait judicieux d'améliorer le modèle. Quelles sont les pistes possibles : soitprendre en compte d'autres variables explicatives (régression multiple ou analyse pour des catégories différentes, on peut séparer les individus par sexe), soit chercher une relation non linéaire.

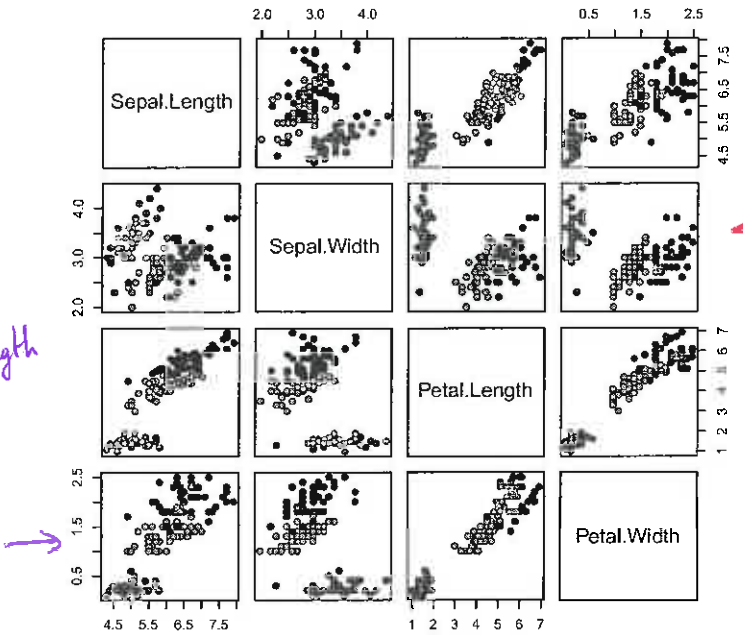
2 Régression linéaire multiple

Il s'agit d'expliquer y avec plusieurs x. Pour l'exemple, on va ré-utiliser les données `iris` incluent dans R :

```
> irisQT <- iris[1:4]
> pairs(irisQT, main = "Anderson's Iris Data -- 3 species",
+       pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])
```

ce graph est trop cool

toutes les abscisses de cette colonne sont Petal width



toutes les ordonnées de cette ligne sont sepal.width

Sur ce graphe on a l'impression que sepal.length et petal.length sont corrélés

Dans tous les cas, il faut commencer par un point de vue univarié et bivarié. Il faut explorer les données avant de se jeter sur une méthode. La statistique n'est pas une boîte à outils. La fonction `pairs()` permet de visualiser rapidement les croisements. On peut l'améliorer facilement, voir l'aide, riche en code, de cette fonction.

2.1 Le modèle de la régression multiple

Le modèle 1 se généralise facilement :

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \epsilon_i \quad \text{avec } i = 1..n \text{ individus} \tag{2}$$

Il ne s'agit plus d'une droite mais d'un plan de regression (pour deux regresseurs), d'un volume (pour trois regresseurs) ou d'un hyperplan au delà. On peut compléter le modèle avec des interactions. Pour k regresseurs il y a 2^k coefficients dans le modèle complet... Ça augmente vite. Avec deux regresseurs, ça fait quatre coefficents :

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \epsilon_i \tag{3}$$

Avec 10 regresseurs, il y en a $2^{10} = 1024...$

La réalistaion de la régression multiple, c'est à dire l'estimation des coefficients avec R est sans difficulté. On généralise aussi facilement :

```
lm(rep~fac1+fac2) ## Modèle additif, sans interaction
```

Pour construire un modèle avec interaction entre les deux facteurs :

```
lm(rep~fac1*fac2)  ## Modèle multiplicatif, avec interaction
```

Dans le cas des données `iris`, aucune variable n'a le rôle naturel de réponse ou de régrésseurs. On peut néamoins tester :

```
> lm2 <- lm(irisQT[,1]~irisQT[,2]*irisQT[,3]*irisQT[,4])
> summary(lm2)

Call:
lm(formula = irisQT[, 1] ~ irisQT[, 2] * irisQT[, 3] * irisQT[,
    4])

Residuals:
     Min       1Q   Median       3Q      Max
-0.78826 -0.23215  0.01429  0.18533  0.74079

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                       1.73247    0.87187   1.987   0.0488 *
irisQT[, 2]                       0.73953    0.27311   2.708   0.0076 **
irisQT[, 3]                       1.04893    0.49247   2.130   0.0349 *
irisQT[, 4]                      -2.18321    1.48898  -1.466   0.1448
irisQT[, 2]:irisQT[, 3]          -0.13072    0.16361  -0.799   0.4256
irisQT[, 2]:irisQT[, 4]           0.45030    0.48976   0.919   0.3594
irisQT[, 3]:irisQT[, 4]           0.18729    0.22641   0.827   0.4095
irisQT[, 2]:irisQT[, 3]:irisQT[, 4] -0.04066    0.07567  -0.537   0.5919
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3119 on 142 degrees of freedom
Multiple R-squared: 0.8648,     Adjusted R-squared: 0.8581
F-statistic: 129.7 on 7 and 142 DF,  p-value: < 2.2e-16
```

peu significatif tout ça...

Les interactions ne sont pas significatives, on peut déjà simplifier le modèle :

```
> lm2 <- lm(irisQT[,1]~irisQT[,2]+irisQT[,3]+irisQT[,4])
```

→ ce modèle est mieux

Ici, on peut tester step

2.2 La colinéarité

Lorsque les regresseurs sont trop corrélés entre eux la regression multiple fonctionne mal. Un des facteurs peut sembler agir à l'inverse de son effet individuel, etc. On parle de problèmes de colinéarité. Voici quelques critères pour résoudre ces problèmes.

- Si deux régresseurs i et j sont tels que $|r_{i,j} > 0,8|$, il faut supprimer un des deux (ils portent la même information)
- Si deux régresseurs i et j sont tels que $r_{i,j}^2 > R^2$, avec R^2 le coefficient de détermination de la régréssion linéaire multiple. Il faut supprimer un des deux.
- Critère VIF : On considère le R_j^2 de la régréssion de la variable j expliquée par les autres regresseurs. Si la grandeur $1/(1 - R_j^2)$ est forte il y a un problème de colinéarité. La limite est fixée selon les auteurs à 4, 5 ou 10. On trouve les VIF sur la première diagonale de l'inverse de la matrice des corrélations.

L'important est de repérer les problèmes et de simplifier le modèle. Dans tous les cas on voit qu'il faut selectionner les variables afin de selectionner le bon modèle.

```
> cor(irisQT) # La matrice des corrélations

             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

On repère tout de suite des problèmes de colinéarité.

```
> solve(cor(irisQT[,2:4]))      # solve() pour l'inverse

             Sepal.Width Petal.Length Petal.Width
Sepal.Width    1.2708149     1.323467  -0.8090419
Petal.Length   1.3234665    15.097572 -14.0523751
Petal.Width   -0.8090419   -14.052375  14.2343350

> lm4 <- lm(irisQT[,3]~irisQT[,2]+irisQT[,4])
> 1/(1-summary(lm4)$r.squared)  # On trouve bien la valeur de VIF... Très forts dans ce cas.

[1] 15.09757
```

2.3 La sélection de variables

La sélection de variables et de modèles et un domaine complexe. Une des idées est de tenter d'optimiser pas à pas un critère de qualité du modèle en ajoutant des régresseurs (méthode ascendante) ou en partant du modèle complet et en le simplifiant (méthode descendante). Le R^2 n'est pas un bon candidat car il augmente mécaniquemeent avec le nombre de régresseurs. Les critères courants sont nommés BIC et AIC (pour Bayesian Information Criterion et Akaike Information Criterion). Sans entrer dans les détails ces critères sont utilisables avec R, la fonction `step()` cherche le bon modèle :

```
> step(lm(irisQT[,1]~irisQT[,2]+irisQT[,3]+irisQT[,4]) , k=2)

Start:  AIC=-343.04
irisQT[, 1] ~ irisQT[, 2] + irisQT[, 3] + irisQT[, 4]

             Df Sum of Sq    RSS     AIC
<none>                    14.445 -343.04
- irisQT[, 4]  1    1.8834 16.329 -326.66
- irisQT[, 2]  1    9.4353 23.881 -269.63
- irisQT[, 3]  1   15.4657 29.911 -235.86

Call:
lm(formula = irisQT[, 1] ~ irisQT[, 2] + irisQT[, 3] + irisQT[,     4])

Coefficients:
(Intercept)  irisQT[, 2]  irisQT[, 3]  irisQT[, 4]
     1.8560       0.6508       0.7091      -0.5565
```

Dans ce cas, la procédure ne parvient pas à eliminer des variables et conserve le (mauvais) modèle proposé en entrée dans lequel `Petal.Width` agit en négatif alors que ce n'est pas le sens de la relation entre `Sepal.Length` et `Petal.width` (cf. le graphique fait avec `pairs()`). La sélection de variables ne nous évite pas de régler les problèmes de colinéarité.

3 Exercice

On dispose de données sur des voitures. Analyser les données et proposer un modèle pour expliquer au mieux la consommation de ces véhicules. On attend que vous testiez les conditions d'applications, normalité, homoscédasticité des variances, distribution des résidus, colinéarité, *etc.* Il est conseillé également de commencer par décrire le contexte statistique et de proposer une rapide analyse univariée du jeu de données ; des graphiques sont strictement nécessaires.

Les données sont disponibles sur le serveur, dans le fichier `cars.csv`[1].

1. données empruntées à R. Rakotomalala, de l'université Lyon2

TD - 3.1
ACP
Analyse en Composantes Principales

Vincent PAYET

"De l'argile, nous faisons un pot, mais c'est le vide à l'intérieur qui retient ce que nous voulons." — Lao-Tseu, Extrait du Tao Te King

Objectif : Les bases de l'ACP et sa réalisation avec le package FactoMineR.

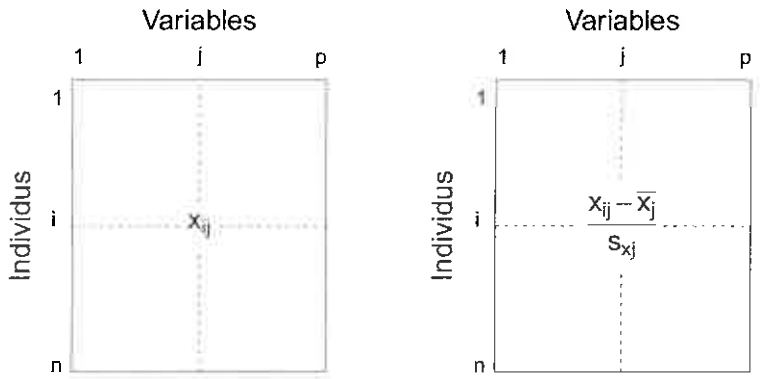
Table des matières

Introduction

L'analyse en composantes principales est la plus connue des méthodes d'analyse multivariée. Elle permet d'analyser un tableau rectangulaire de données numériques. Il s'agit d'appliquer la méthode factorielle, le plus souvent sur les données normées. Les résultats majeurs de cette analyse sont :

- Une typologie des individus
- Une typologie des variables
- Une réduction de nombre de dimensions
- Un nouveau jeu de variables synthétiques, indépendantes et hiérarchisées.

1 Données



On dispose d'un tableau de n individus en ligne décrits par p variables en colonne. Ce tableau est généralement normé, c'est-à-dire centré et réduit à partir des moyennes et écart-types de chaque colonne. Ce tableau peut être vu comme une matrice nommée X. Deux points de vue sont possibles sur cette matrice : elle contient n points "ligne" de l'espace $\mathbb{R}^p$ ou p points "colonnes" de $\mathbb{R}^n$.

2 Le principe de l'ACP

Faire une ACP, c'est avant tout appliquer une procédure de calcul : une diagonalisation d'une matrice symétrique, c'est-à-dire la recherche de valeurs propres et vecteurs propres. X n'est ni carrée ni symétrique, $dim(X) = (n, p)$.

Avec X matrice centrée réduite et D matrice diagonale des poids

On définit $C = X^t DX$

C est la matrice de variance-covariance (X centrée) ou de corrélation (X centrée-réduite), elle est symétrique donc diagonalisable. C est de dimension (p, p). La recherche des p valeurs propres, λ et des p vecteurs propres V_j associés se fait sur C.

$$\text{Il existe un vecteur V et une valeur } \lambda \text{ tels que} \quad XV = \lambda V$$

On nomme alors :

$$U \quad \text{matrice des vecteurs propres}$$
$$\Lambda \quad \text{matrice diagonale des valeurs propres}$$

La diagonalisation est là :

$$C = U\Lambda U^y \leftrightarrow U^t C U = \Lambda$$

La matrice diagonale des valeurs propres et la matrices des vecteurs propres associés :

$$\Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_p \end{pmatrix} \qquad U = \begin{pmatrix} U_1 & U_2 & \dots & U_p \\ & & & \end{pmatrix}$$

On obtient ainsi une nouvelle base indépendante. Chaque vecteur propre définissant un axe principal. Pour hiérarchiser les axes il suffit de classer les valeurs propres par ordre décroissant. Chaque valeur propre représente la part d'inertie portée (projetée) sur l'axe associé :

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$$

Il s'agit ensuite d'obtenir les coordonnées des points dans la nouvelle base et de projeter sur les premiers axes. Projeter sur le plan formé par les axes 1 et 2 c'est simplement représenter le nuage des points en utilisant leurs coordonnées sur ces axes.

Sans entrer dans les détails, le point de vue des variables est similaire et on obtient la même décomposition en valeurs et vecteurs propres. Notons qu'en utilisant C on s'intéresse surtout aux liaisons linéaires entre les variables initiales. Il est inopportun de faire une ACP sur des données qui s'éloignent fortement du cas linéaire.

3 Les sorties de l'ACP

La partie calculatoire précédente est assurée par un programme et débouche sur des sorties numériques et surtout graphiques qui rendent l'ACP vraiment intéressante.

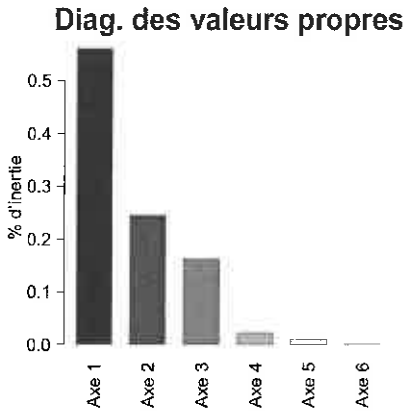
3.1 Valeurs propres

Pour p variables de départ, on obtient p nouveaux axes et autant de valeurs propres λ. La somme de ces valeurs propres vaut p pour une ACP normée. C'est la variance totale du nuage. Pourquoi ? Cherchez la variance de chaque variable normée...

Il est judicieux de les classer, de calculer la participation en pourcentage par rapport au total et de proposer un graphique dit "diagramme des valeurs propres".

λ	%	% cumul.
3.37	56.12	56.12
1.47	24.52	80.64
0.97	16.22	96.86
0.13	2.17	99.02
0.06	0.94	99.96
0.00	0.04	100.00

TABLE 1 : Tableau des valeurs propres, pourcentages d'inertie et pourcentages cumulés.



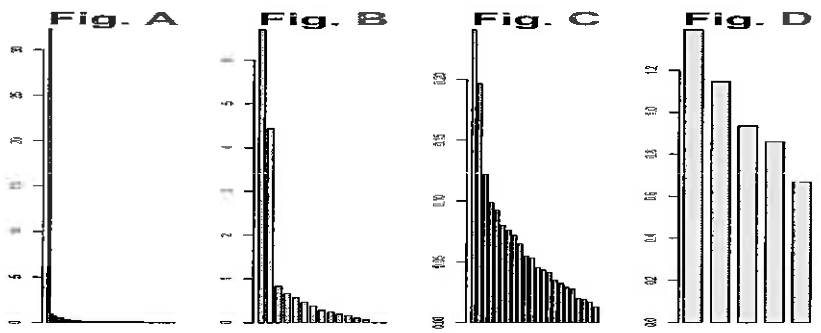
3.1.1 Choix du nombre de dimensions

Combien d'axes faut-il conserver ? Et lesquels ?

On garde toujours les axes les plus importants. Le premier forcément, associé au second si on veut pouvoir dessiner un plan. Pour certaines utilisations, des auteurs préconisent l'utilisation du seul premier axe qui fournit une ordination des individus. Ce cas est marginal pour nous. La lecture du diagramme de valeurs propres va ensuite nous aider. On recherche, en partant du premier, des axes qui expliquent des parts de variance comparables. Chaque saut dans le graphe indique une coupure possible.

Pour le graphe de la partie 3.1, on peut ne garder que le premier axe (56% d'inertie totale expliquée). On peut ensuite conserver l'axe 2. Le plan (1-2) explique alors 81% de l'inertie. Il faut toutefois conserver le troisième axe également qui est du même ordre de grandeur que le deuxième.

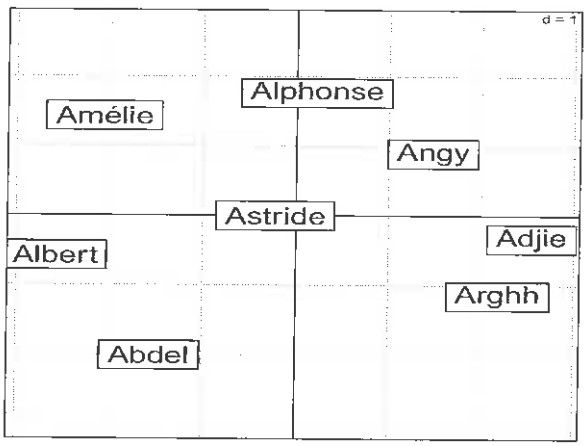
Voici 4 cas caractéristiques :



Pour la fig. A, il n'apparaît qu'une variable sous-jacente. Les p variables de départ sont toutes corrélées. C'est un cas rencontré en morphométrie : tout le corps grandit proportionnellement (ou presque...). On peut toutefois utiliser le plan (1-2) pour la représentation, sans interpréter l'axe2. Sur la fig. B on conserve deux axes, tous les autres ne contiennent que du bruit. Idem sur la fig. C, toutefois le bruit est plus important. Sur la fig. D aucune variable n'est vraiment importante. Il n'y avait aucune relation linéaire entre les 5 variables de départ, l'ACP n'est peut-être pas judicieuse.

3.2 Plan factoriel des individus

Il s'agit de représenter les individus de départ sur un plan qui conserve au mieux les distances entre ces points de $\mathbb{R}^p$. Des individus éloignés diffèrent (du point de vue des données) et des individus proches se ressemblent probablement (mais peuvent avoir subit un rapprochement artificiel par projection). On recherche sur ce graphique des groupes d'individus en s'intéressant d'abord aux individus loin du centre du nuage : ce sont eux qui contribuent le plus aux axes. Le centre du nuage représente l'individu moyen.

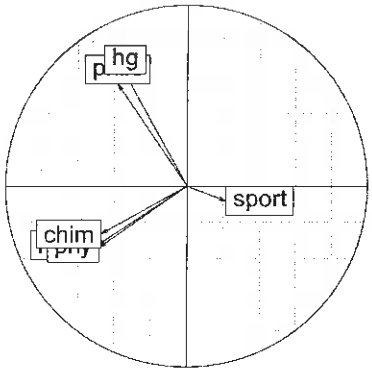


3.3 Cercle des corrélations

Le cercle des corrélations est la projection du nuage des variables. Les axes sont les mêmes qu'en 3.2 et on pourra rapprocher les deux analyses. Les variables sont bien représentées si elles sont proches du cercle. Les angles entres les vecteurs (les flèches) nous montrent les corrélations entre les variables :

- Un angle faible pour deux variables corrélées positivement
- Un angle vers 180° pour deux variables corrélées négativement
- Un angle vers 90° pour deux variables indépendantes.

La proximité avec les axes suit la même logique, on peut ainsi repérer si les axes sont des variables sous-jacentes qui résument les données. Cette lecture va nous permettre de nommer les axes.



4 Réalisation de l'ACP avec

Il existe des fonctions de pour réaliser l'ACP, dont `prcomp()` et `princomp()`. Toutefois ces fonctions sont limitées et nous allons utiliser des extensions qui proposent des fonctions spécialement conçues pour certaines tâches. On appelle ces extensions des *packages*. Un *package* fournit une bibliothèque (*library*, en anglais) de fonctions et de jeux de données. Pour l'analyse multivariée on peut citer les *packages* suivants :

FactoMineR. Développé à Agrocampus Ouest. Le *package* que nous allons utiliser. [2][3]

ade4. Développé à l'Université de Lyon. Il propose des fonctions adaptées à l'analyse de données écologiques [1].

4.1 Le *package* FactoMineR

Le *package* FactoMineR n'est pas présent par défaut, il faut le télécharger et l'installer. Tout se fait avec cette commande (attention, les majuscules comptent).

```
> install.packages("FactoMineR")
```

L'installation n'est à faire qu'une fois. En revanche il faut charger le *package* en mémoire à chaque nouvelle session de R avant d'utiliser les fonctions. Pour charger une bibliothèque de fonctions on utilise la fonction `library()` :

```
> library(FactoMineR)
```

On peut obtenir une rapide description du *package* et la liste des fonctions avec la commande :

```
> help(package="FactoMineR")
```

La commande de FacotoMineR pour faire une ACP se nomme `PCA()`, d'après le nom anglais de la méthode. Utilisons une première fois la fonction `PCA()` sur des valeurs aléatoires pour explorer les sorties proposées :

```
> set.seed(42)
> datAlea <- matrix(rnorm(20),ncol=4)
> PCA(datAlea,graph=FALSE)

**Results for the Principal Component Analysis (PCA)**
The analysis was performed on 5 individuals, described by 4 variables
*The results are available in the following objects:

   name                 description
1  "$eig"               "eigenvalues"
2  "$var"               "results for the variables"
3  "$var$coord"         "coord. for the variables"
4  "$var$cor"           "correlations variables - dimensions"
5  "$var$cos2"          "cos2 for the variables"
6  "$var$contrib"       "contributions of the variables"
7  "$ind"               "results for the individuals"
8  "$ind$coord"         "coord. for the individuals"
9  "$ind$cos2"          "cos2 for the individuals"
10 "$ind$contrib"       "contributions of the individuals"
11 "$call"              "summary statistics"
12 "$call$centre"       "mean of the variables"
13 "$call$ecart.type"   "standard error of the variables"
14 "$call$row.w"        "weights for the individuals"
15 "$call$col.w"        "weights for the variables"
```

la fonction `set.seed()` permet d'utiliser des méthodes pseudo-aléatoires de façon répétable. On crée ensuite une matrice de 5 lignes et 4 colonnes avec des valeurs normales.

La fonction renvoie 4 objets principaux :

Nom	Description
$eig	les valeurs propres
$var	les résultats pour les variables
$ind	les résultats pour les individus
$call	un résumé des statistiques

Pour accéder à ces objets il est judicieux de conserver le résultat de l'ACP en assignant l'analyse à un objet . On peut lire les valeurs propres ainsi :

```
> aleaCP <- PCA(datAlea,graph=FALSE)
> aleaCP$eig
```

On remarque que l'objet `aleaCP$eig` est un tableau. Comment n'obtenir que les valeurs propres ?

```
> aleaCP$eig[,1]

[1] 1.6692186 1.5584118 0.6068096 0.1655600
```

5 Un exemple en détail

5.1 Données et contexte

Nous utilisons un jeu de données du *package* ade4 : seconde. Il s'agit d'un tableau de 8 notes pour 22 élèves de seconde. Les variables sont les suivantes :

```
[1] "HGEO" "FRAN" "PHYS" "MATH" "BIOL" "ECON" "ANGL" "ESPA"
```

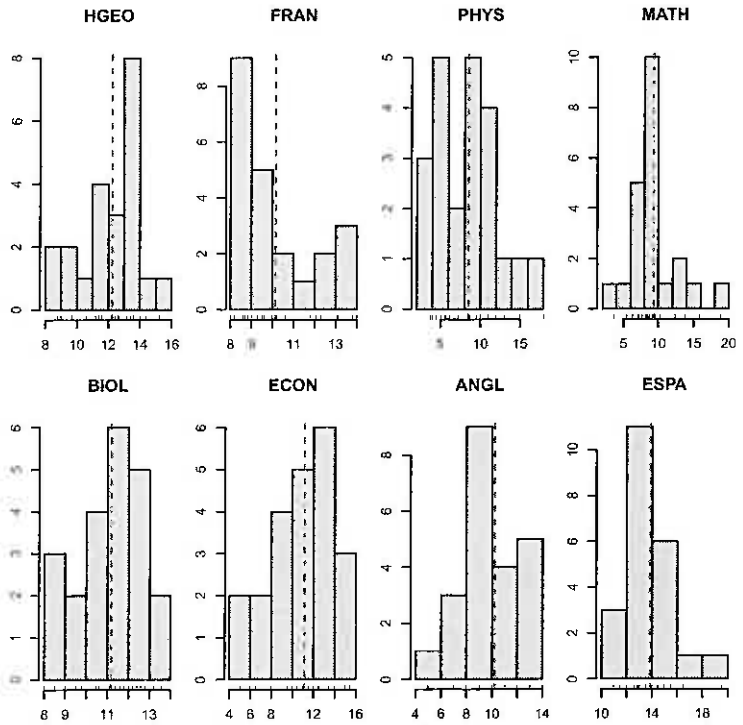
Pour les besoins du cours nous ajoutons une (fausse) information de sexe sur les données et une note de musique à l'aide de la fonction `sample()` :

```
> sexe <- sample(c("H","F"),22,rep=T)
> MUSI <- sample(10:14,22,rep=T)
> sexe;MUSI

 [1] "H" "H" "H" "F" "H" "F" "F" "F" "F" "F" "H" "H" "H" "F" "H" "F" "F" "H" "H"
[20] "F" "F" "F"

 [1] 13 12 14 10 11 14 13 11 10 10 11 12 10 13 10 11 12 10 12 10 11 13
```

5.2 Analyses uni- et bi-variées

Des préliminaires obligatoires. Pour mener à bien une analyse multivariée il faut connaître ses données. Les quelques propositions ici n'épuisent pas les possibles :

```
> par(mfrow=c(2,4),mar=c(2,2,4,1))
> for( i in 1:8){
+   hist(seconde[,i],main=names(seconde)[i],xlab="",ylab="",col="lightgray")
+   rug(seconde[,i])
+   abline(v=mean(seconde[,i]),lty=2,col="red")
+ }
```



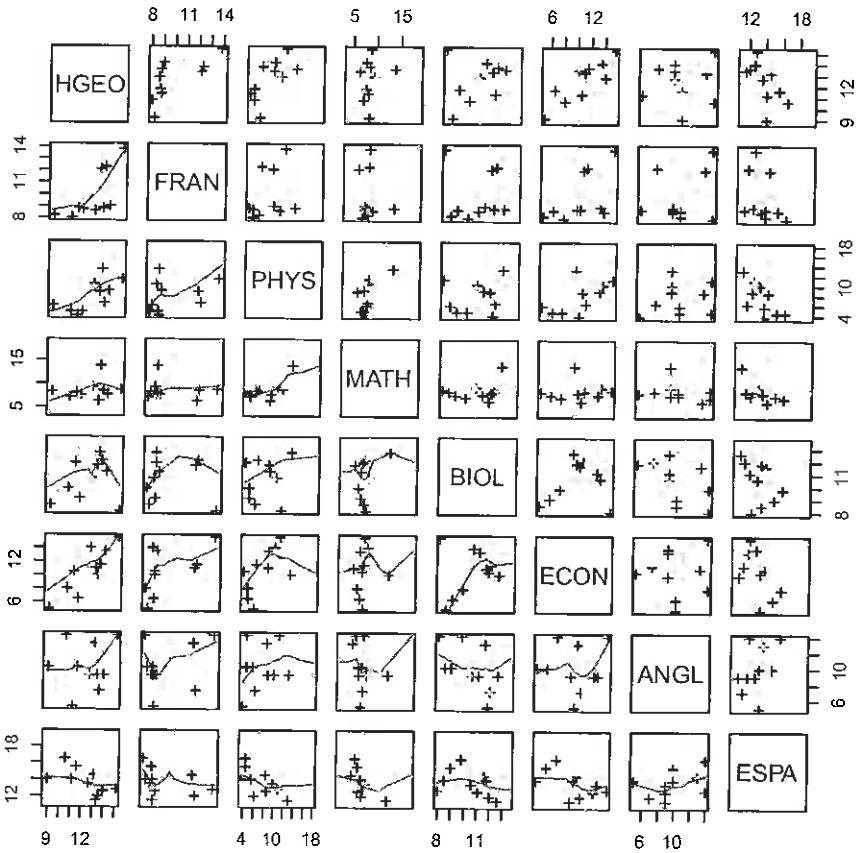
Notez la boucle `for` qui permet de parcourir les colonnes du tableau. Il est souvent possible d'éviter les boucles explicites avec . La fonction `apply()` permet d'appliquer

une fonction aux lignes ou colonnes d'un tableau. On obtient les moyennes en colonnes en fixant la valeur MARGIN à 2 (MARGIN=1 pour les lignes). On pourrait faire de même avec des fonctions graphiques (non traité ici).

```
> round(apply(seconde,MARGIN=2,mean),2)
 HGEO  FRAN  PHYS  MATH  BIOL  ECON  ANGL  ESPA
12.23 10.14  8.53  9.32 11.18 11.18 10.23 13.88
```

Pour l'analyse bivariée de données numériques, la fonction `pairs()` donne une vue d'ensemble. On pourrait appeler `pairs(seconde)` mais on propose quelques améliorations. Notez au passage la fonction `ifelse()` utilisée ici pour définir un vecteur de couleurs et un vecteur de symboles.

```
> couleur <- ifelse(sexe=="H","blue","pink")
> symbol <- ifelse(sexe=="H",3,18)
> pairs(seconde,lower.panel=panel.smooth,pch=symbol,cex=0.8,col=couleur)
```



Ce dernier graphique propose tous les croisements deux à deux. Au croisement de HGEO et FRAN (deuxième case de la première ligne), on lit le nuage de point de HGEO

en fonction de FRAN. Nous avons ajouté une courbe de régression non linéaire sur la partie du bas. FRAN et HGEO sont corrélés, MATH et PHYS aussi. Ce graphique est symétrique. Il joue le même rôle que la matrice des corrélations :

```
> round(cor(seconde),2)

      HGEO  FRAN  PHYS  MATH  BIOL  ECON  ANGL  ESPA
HGEO  1.00  0.68  0.63  0.53  0.27  0.65  0.33 -0.10
FRAN  0.68  1.00  0.56  0.44  0.16  0.46  0.33 -0.09
PHYS  0.63  0.56  1.00  0.70  0.42  0.42  0.33 -0.34
MATH  0.53  0.44  0.70  1.00  0.32  0.18  0.25 -0.13
BIOL  0.27  0.16  0.42  0.32  1.00  0.36 -0.14 -0.09
ECON  0.65  0.46  0.42  0.18  0.36  1.00  0.22 -0.06
ANGL  0.33  0.33  0.33  0.25 -0.14  0.22  1.00  0.39
ESPA -0.10 -0.09 -0.34 -0.13 -0.09 -0.06  0.39  1.00
```

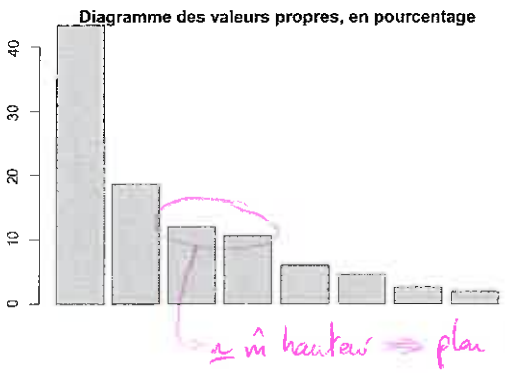
5.3 L'ACP en action

On réalise l'ACP avec PCA() comme on l'a déjà vu. Par défaut la fonction dessine deux graphiques, le nuage des individus et le cercle des corrélations, on ne les affiche pas tout de suite (avec le paramètre graph=FALSE).

```
> acpsec <- PCA(seconde,graph=FALSE)
```

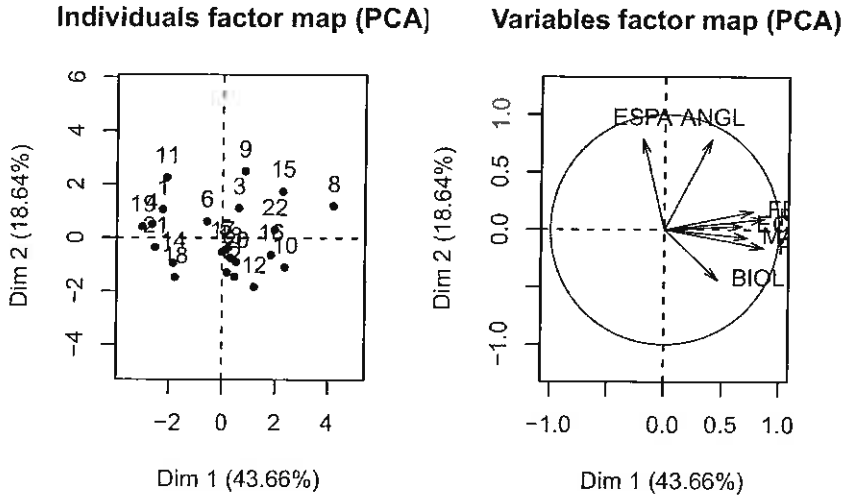
Comment choisir le nombre d'axes à étudier ? Il faut le graphique des valeurs propres :

```
> barplot(acpsec$eig[,2],main= "Diagramme des valeurs propres, en pourcentage")
```



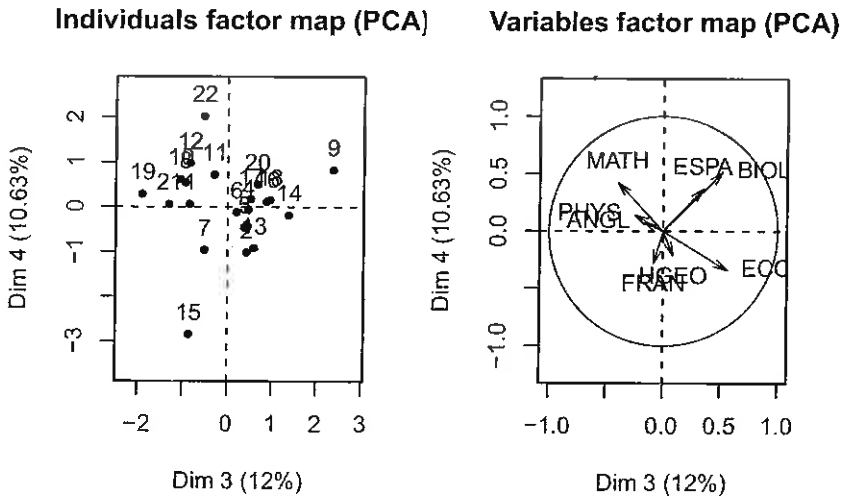
Ici on va conserver assurément les deux premiers axes.

```
> par(mfrow=c(1,2))
> plot.PCA(acpsec)
> plot(acpsec,choix="var")
```



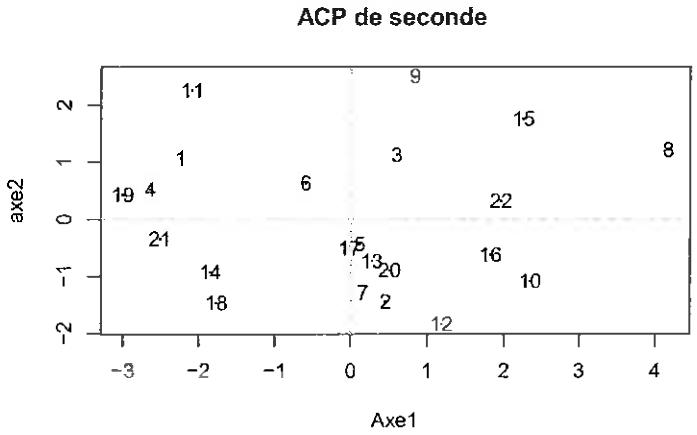
Nous pourrions aussi utiliser les axes 3 et 4. Leur forte similarité incite d'ailleurs à regarder le plan (3-4) et non les plans (1-3) et (1-4). Comment pourrions nous voir ce qui se passe sur les axes suivants ?

```
> par(mfrow=c(1,2))
> plot(acpsec,axes=c(3,4))
> plot(acpsec,axes=c(3,4),choix="var")
```



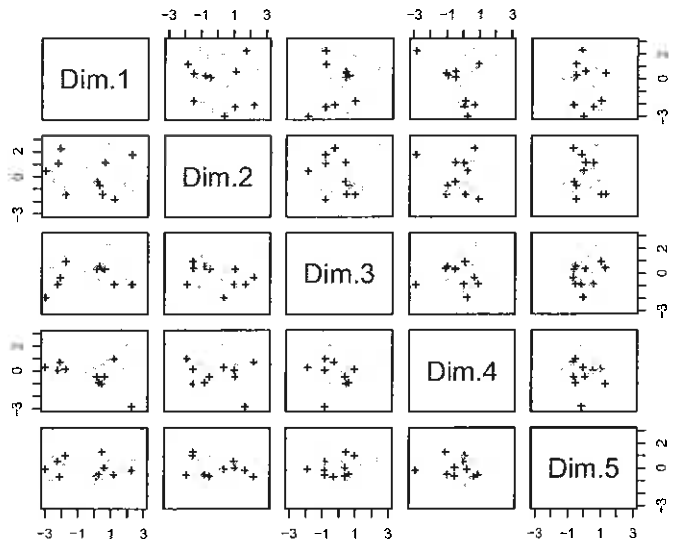
On peut aussi faire ses propres graphiques. Les coordonnées sont utilisables comme n'importe quelles données.

```
> x <- acpsec$ind$coord[,1]
> y <- acpsec$ind$coord[,2]
> plot(x,y,pch=".",main="ACP de seconde ", xlab="Axe1", ylab="axe2")
> text(x,y,1:22)
> abline(v=0,lty=2, col="grey")
> abline(h=0,lty=2, col="grey")
```



On peut d'ailleurs voir tous les plans rapidement :

```
> pairs(acpsec$ind$coord,xlim=c(-3,3),ylim=c(-3,3),col=couleur,pch=symbol,cex=0.7)
```



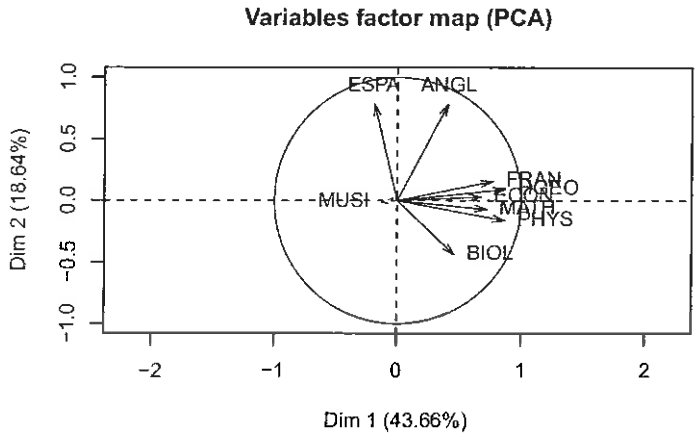
On remarque que le nuage de points est effectivement plus étalé sur le plan(1-2) et moins étalé ensuite. C'est exactement ce qu'on demande à l'ACP.

5.4 Données supplémentaires

Il est possible d'utiliser des variables ou des individus supplémentaires pour aider à l'interprétation. Ces données ne participent pas à la définition des axes principaux et à la décomposition de l'inertie du nuage. Les résultats de l'ACP sont les mêmes avec ou sans ces informations supplémentaires, en revanche elles permettent d'enrichir l'interprétation.

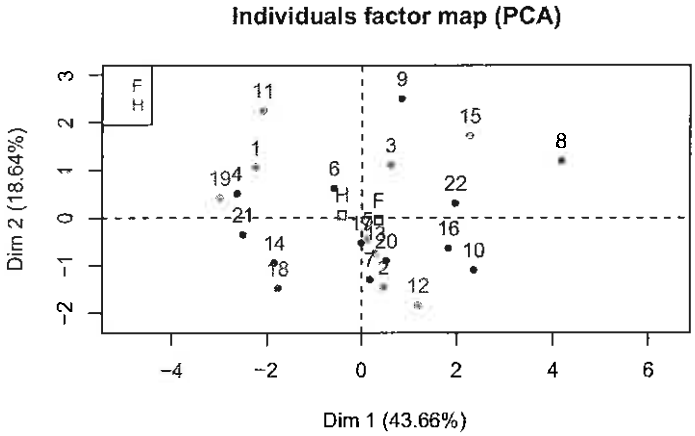
Les variables quantitatives supplémentaires sont projetées sur le cercle des corrélations :

```
> sec2 <- data.frame(seconde, MUSI, sexe)
> head(sec2)

  HGEO FRAN PHYS MATH BIOL ECON ANGL ESPA MUSI sexe
1 11.6  8.7  4.5  7.6  9.0  6.5   10 15.5   13    H
2 13.6 12.3  6.2  8.5 12.0 11.5    7 12.0   12    H
3 13.2 12.1  8.5  6.3 11.6 11.0   13 14.5   14    H
4  8.8  8.2  5.0  3.7 10.6 10.0   11 14.5   10    F
5 12.7  8.6 10.0  9.3 10.6 14.0    9 13.5   11    H
6 12.4 10.0  5.5  9.2 10.5 11.0   10 15.0   14    F

> acpsec2 <- PCA(sec2,quanti.sup=9, quali.sup=10,graph=FALSE)
> plot(acpsec2,choix="var")
```



Les variables qualitatives sont représentées par des couleurs ou des symboles sur le plan factoriel des individus :

```
> plot(acpsec2, habillage = 10)
```



Les individus supplémentaires sont projetés sur le plan factoriel des individus (non traité ici).

5.5 Contribution et $\cos^2$

On accède aux contributions et qualités de représentation via l'objet crée par `PCA()` :

```
> round(acpsec$var$contrib,2)

     Dim.1 Dim.2 Dim.3 Dim.4 Dim.5
HGED 21.74  0.57  0.80  5.19  3.28
FRAN 17.03  1.67  0.70  9.23 35.07
PHYS 21.43  1.77  6.24  2.39 10.37
MATH 14.92  0.35 16.05 21.17  4.09
BIOL  6.07 12.89 27.95 31.37  0.09
ECON 13.02  0.06 32.57 13.53  7.79
ANGL  4.78 41.32  2.81  0.69 28.07
ESPA  1.02 41.39 12.89 16.44 11.22

> round(acpsec$var$cos2,3)

     Dim.1 Dim.2 Dim.3 Dim.4 Dim.5
HGED 0.759 0.009 0.008 0.044 0.016
FRAN 0.595 0.025 0.007 0.078 0.170
PHYS 0.748 0.026 0.060 0.020 0.050
MATH 0.521 0.005 0.154 0.180 0.020
BIOL 0.212 0.192 0.268 0.267 0.000
ECON 0.455 0.001 0.313 0.115 0.038
ANGL 0.167 0.616 0.027 0.006 0.136
ESPA 0.036 0.617 0.124 0.140 0.054
```

5.6 Interprétation

La démarche de l'ACP vous semble-t-elle encore un peu compliquée ? Prenez une grande respiration : le meilleur est encore à venir. En effet, les calculs qui aboutissent à l'ACP sont simples, il s'agit fondamentalement d'une diagonalisation d'une matrice carrée. Ces calculs sont justes infiniment longs pour un humain et presque rien pour nos machines. Mais lorsque le calcul est fait, il reste tout à faire : l'interprétation.

Une partie de la démarche peut-être décrite de façon régulière :

- Combien de dimensions garder ? Voir plus haut.
- Nommer les axes. On peut commencer par le cercle des corrélations.
- Chercher des groupes d'individus.
- Utiliser les variables supplémentaires pour enrichir l'interprétation
- ... la suite dépend du jeu de données, du savoir faire, de l'expérience. L'analyse multivariée est-elle un art ?

5.7 L'autre bout de l'interprétation

Nous n'avons encore pas trop insisté sur le tableau de données que nous mettons en entrée de la procédure d'ACP. Pourtant le choix des variables est primordial et constitue l'autre part d'humain, donc de savoir faire et, en première approche au moins, d'approximation. Il faut faire des choix en début d'analyse : quelles variables seront actives, c'est à dire participantes à la partie calculatoire de l'analyse ? Lesquelles seront illustratives, c'est-à-dire utilisées uniquement pour l'interprétation ?

Un exemple : pour une enquête sur la consommation pour laquelle nous disposons d'informations sur les pratiques des consommateurs et de descripteurs (âge, genre, CSP, ancienneté du rapport client, etc.), il serait certainement judicieux de faire l'analyse sur les pratiques et de conserver les descripteurs pour l'interprétation.

Les données ne sont pas toujours aussi clairement séparées. Quant à la création du questionnaire, c'est encore une autre compétence.

6 À vous de jouer !

Vous allez réaliser une ACP sur le jeu de données `olympic` du *package* ade4. il faut pour cela installer ce *package*. Les données sont ensuite appelées avec `data()`. Le jeu de données contient deux objets, des résultats dans `$tab` et un nombre de points dans `$score` :

```
> data(olympic)
> names(olympic)

[1] "tab"   "score"

> olympic$score

 [1] 8488 8399 8328 8306 8286 8272 8216 8189 8180 8167 8143 8114 8093 8083 8036
[16] 8021 7869 7860 7859 7781 7753 7745 7743 7623 7579 7517 7505 7422 7310 7237
[31] 7231 7016 6907

> head(olympic$tab,4)

    100 long  poid haut   400   110  disq perc  jave   1500
1 11.25 7.43 15.48 2.27 48.90 15.13 49.28  4.7 61.32 268.95
2 10.87 7.45 14.97 1.97 47.71 14.46 44.36  5.1 61.76 273.02
3 11.18 7.44 14.20 1.97 48.29 14.81 43.66  5.2 64.16 263.20
4 10.62 7.38 15.02 2.03 49.06 14.72 44.80  4.9 64.04 285.11
```

L'ACP est à faire sur `olympic$tab`. Attention pour l'interprétation, une épreuve réussie correspond parfois à une valeur forte (*long*, *poid*...), parfois à une valeur faible

(*100m...*). Pour compléter l'analyse, demandez-vous aussi comment utiliser les scores en relation avec l'ACP.

Consignes : Produire un mini rapport sur un A4 recto-verso avec le contexte, les sorties (numériques et/ou graphiques) nécessaires pour conclure et vos conclusions. Le script commenté, doit être ajouté en annexe. Pour la mise en forme des textes copiés de , définissez et utilisez un style avec une police mono comme `Courrier New`.

Le rapport est à rendre sur e-campus, cours d'analyse multivariée, partie travaux.

Références

[1] D. Chessel, A.B. Dufour, and J. Thioulouse. The ade4 package-I- One-table methods. *R News*, 4 :5–10, 2004.

[2] F. Husson, S. Lê, and J. Pagès. *Analyses de données avec R*. PUR, 2009.

[3] S. Lê, J. Josse, and F. Husson. Factominer : an r package for multivariate analysis. *Journal of Statistical Software*, 25(1), 2008.

TD - 3.2
CAH
Classification Ascendante Hiérarchique

Vincent PAYET

« [...] mais vous êtes peut-être trop rationnel et pas assez raisonnable. »
— Volle, Parador 2009

Objectifs : Fiche d'exercice sur la CAH. À l'issue de la séance vous devez savoir faire une CAH, l'interpréter et exporter des données hors de .

Table des matières

1 Exportation

Plusieurs fonctions permettent d'exporter des données. Pour un tableau de données de type `data.frame` on peut utiliser la fonction `write.table()`. Voici quelques exemples de fonctionnement. Notez que les 4 premières lignes sont ici juste pour créer un jeu de données à exporter (on crée un `data.frame` avec 2 colonnes de nombres et une colonne de noms) :

```
> x <- 1:10
> y <- rnorm(x)
> Nom <- LETTERS[1:10]
> df <- data.frame(x,y,Nom)
> write.table(df,"df1.txt")
> write.table(df,"df2.txt",dec=",")
> write.table(df,"df3.csv",dec=",",sep=";")
> write.table(df,"df4.csv",dec=",",row.names=FALSE)
```

1. Où sont les fichiers crées ? Dans votre répertoire de travail, s'il n'est pas définit... on recommence.
2. Les fichiers `*.txt` s'ouvrent avec un éditeur de texte. On peut aussi forcer leur ouverture avec un tableur (menu ouvrir depuis le tableur).
3. Les fichiers `*.csv` s'ouvrent avec un tableur. On peut aussi forcer leur lecture avec un éditeur de texte. Ce sont des fichiers textes contenant des champs (ou colonnes) séparés par un symbole. Historiquement, les champs sont séparés par des virgules et CSV signifie *comma-separated value*.
4. Les différentes options utilisées ici doivent être identifiées. Pour plus d'options, consulter l'aide : `?write.table`

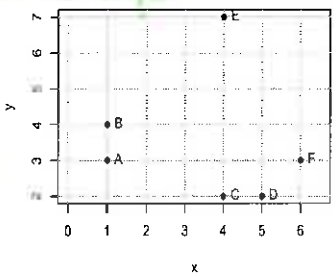
2 la CAH en action

On présente deux façons de réaliser la CAH. La première (partie 2.1 à 2.3) permet de détailler les étapes. La seconde (2.4 et 2.5) utilise le *package* FactoMineR. Son utilisation est rapide et propose des sorties intéressantes, notamment un arbre 3D superposé à une carte factorielle. Les CAH de FactoMineR se font forcément sur les axes d'une analyse factorielle.

2.1 Distances

À partir d'un tableau de données, on veut des distances. On utilise et représente ici des points en deux dimensions. L'objet `LETTERS`, prédéfini dans R, est pratique pour crée des noms rapidement. la fonction `t()` renvoie la transposée d'un tableau.

```
> Cpoints <- matrix(c(1,3,1,4,4,2,5,2,4,7,6,3),ncol=2,byrow=TRUE)
> n <- nrow(Cpoints)
> rownames(Cpoints) <- LETTERS[1:n]
> colnames(Cpoints) <- c("x","y")
> t(Cpoints)

  A B C D E F
x 1 1 4 5 4 6
y 3 4 2 2 7 3

> plot(Cpoints,type="n",xlim=c(0,6.5))
> abline(v=0:7,col="grey")
> abline(h=0:7,col="grey")
> points(Cpoints,pch=16)
> text(Cpoints,LETTERS[1:n],pos=4)
```



La fonction `dist()` permet de calculer la matrice des distances selon plusieurs méthodes :

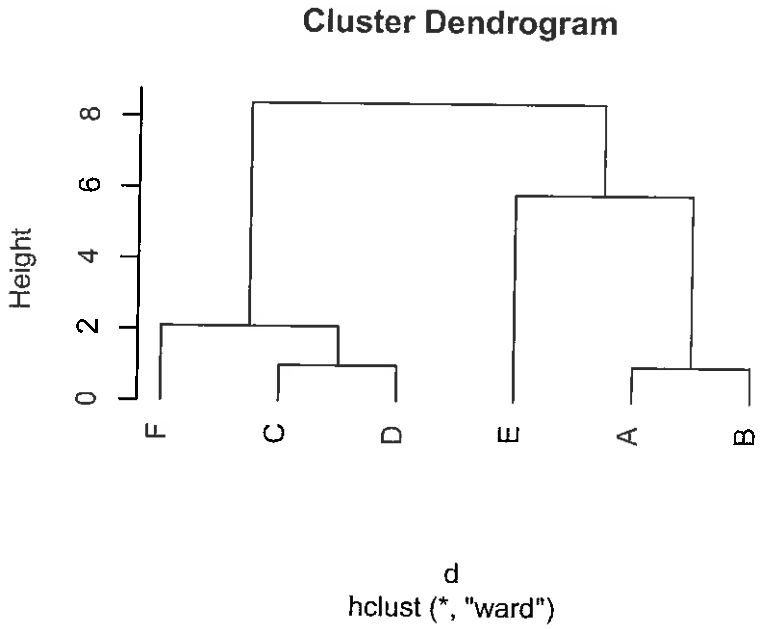
```
> dist(Cpoints) #  Par défaut, la distance classique

         A        B        C        D        E
B 1.000000
C 3.162278 3.605551
D 4.123106 4.472136 1.000000
E 5.000000 4.242641 5.000000 5.099020
F 5.000000 5.099020 2.236068 1.414214 4.472136
```

```
> dist(Cpoints,method="manhattan")

  A B C D E
B 1
C 4 5
D 5 6 1
E 7 6 5 6
F 5 6 3 2 6
```

2.2 Arbres

La classification hiérarchique se fait avec `hclust()` (*hierarchical **cluster***).

```
> d <- dist(Cpoints)
> hc <- hclust(d, method="ward")
> plot(hc, hang=-1)
```



Pour les options de paramètres (`hang`, `method`...) voir l'aide `?plot.hclust`. Essayer aussi la fonction `rect.hclust()`.

2.3 Partitions : coupure de l'arbre

On va conserver le découpage en classes en coupant l'arbre à une certaine hauteur (`h`) ou pour un nombre déterminé de classes (`k`).

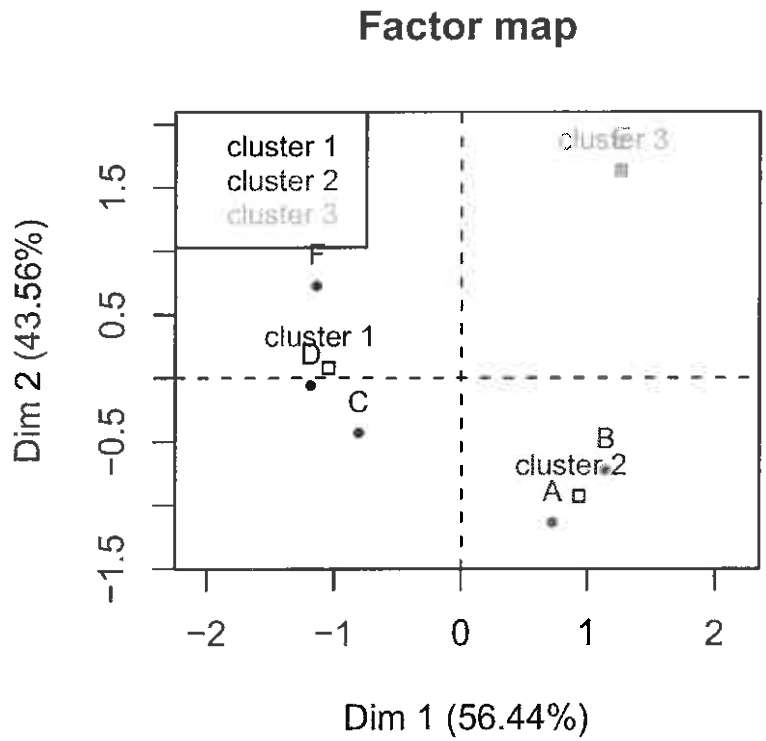
```
> cutree(hc,h=3)

A B C D E F
1 1 2 2 3 2

> cutree(hc,k=3)
```

```
A B C D E F
1 1 2 2 3 2

> classes <- cutree(hc,k=3)
> data.frame(Cpoints,classes )

  x y classes
A 1 3       1
B 1 4       1
C 4 2       2
D 5 2       2
E 4 7       3
F 6 3       2
```

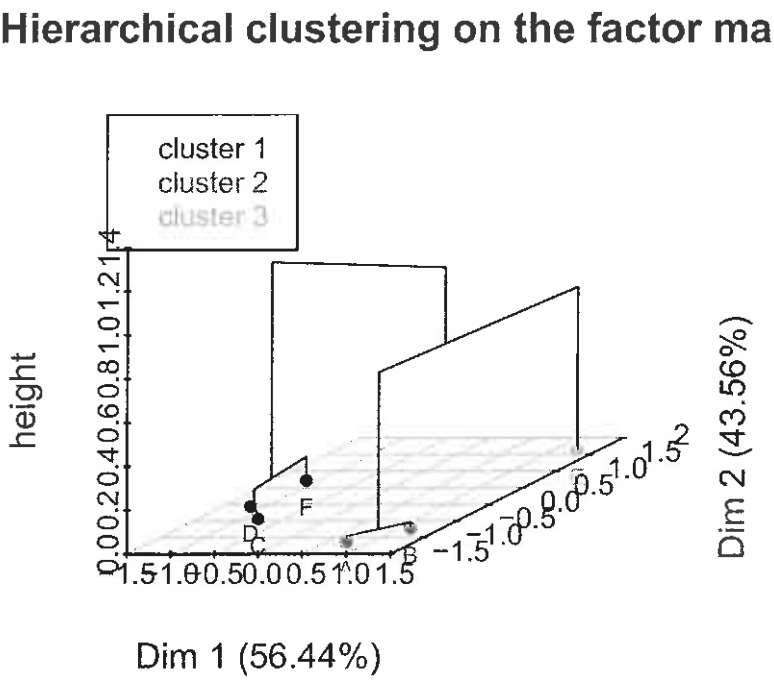
2.4 CAH avec FactoMineR

La fonction `HCPC()` de FactoMineR permet de réaliser la CAH complétement : calcul des distances, de l'arbre et découpe en classes. En revanche, elle ne réalise cette CAH que sur les axes d'une analyse factorielle (ACP, AFC, ACM). On peut produire plusieurs graphiques avec la fonction `plot.HCPC()` :

```
> pca <- PCA(Cpoints,graph=FALSE)
> hp <- HCPC(pca,graph=FALSE)
> plot.HCPC(hp,choice="tree")

> plot.HCPC(hp,choice="map",draw.tree=FALSE)
```




```
> plot.HCPC(hp,choice="3D.map",angle=30)
```



On sait représenter l'arbre et les classes mais comment récupérer la variable qualitative de partition ? Le fonctionnement de `HCPC()` est caractéristique des fonctions de : l'objet `hp` créé contient plusieurs sous objets. On peut explorer cette liste d'objets avec `names()`.

```
> names(hp)

[1] "data.clust" "desc.var"   "desc.axes"  "call"       "desc.ind"

> hp$data.clust

  x y clust
D 5 2     1
F 6 3     1
C 4 2     1
A 1 3     2
B 1 4     2
E 4 7     3
```

`hp$data.clust` contient les données de départ et la variable de partition.

2.5 Description des classes

L'objet créé par `HCPC()` contient également des informations de description de classes. Il s'agit d'identifier les variables caractéristiques des classes. Seules les variables avec une valeur test inférieure à -2 ou supérieure à 2 sont citées. Dans le cas présent les variables n'ont pas de sens.

```
> hp$desc.var$quanti

$`1`
NULL

$`2`
     v.test Mean in category Overall mean sd in category Overall sd    p.value
x -2.088173                1          3.5              0   1.892969 0.03678226

$`3`
   v.test Mean in category Overall mean sd in category Overall sd    p.value
y 2.04939                7          3.5             NA   1.707825 0.04042398
```

Plus généralement, comment lire ces sorties ? Pour chaque groupe, on trouve les variables caractéristiques de la classe, c'est-à-dire celles pour lesquelles la classe est différente de l'ensemble des données. Il y a deux façons d'être différents pour une variable quantitative : plus petit ou plus grand. L'information est donnée par le signe de la valeur-test (`v.test`).

Pour finir il faut nommer les classes. Ce travail de synthèse à partir de la description des classes par leurs individus et par les variables et de leur positionnement sur le plan factoriel est la partie intéressante – et difficile – de l'analyse.

3 Exercice : `olympic` le retour

1. Vous devez réaliser une CAH sur les données `olympic` et sur les axes de l'ACP de `olympic`.
2. Identifiez des classes pertinentes et nommez-les. Exportez l'information (tableau de départ et classes) sur un tableur.
3. Vous devez aussi écrire un script commenté et facilement réutilisable qui reprend les éléments pour réaliser les étapes suivantes :
   - Import de données
   - Analyses uni- et bivariées
   - ACP
   - CAH
   - Export de données

Boot os

ID

Bureau

TD - 3.3
Étude de cas
Subventions agricoles

Vincent PAYET

« J'écoute et j'oublie; je lis et je retiens ; je fais et je comprends. »
— Proverbe chinois, peut-être Confucius.

Objectif : Réaliser une étude de cas sur des données décrivant l'agriculture française par département. Le travail demandé couple l'analyse multivariée et une représentation cartographique.

1 Les données

Vous disposez d'un jeu de données nommé `Subve.xls` et d'un fond de carte des départements français au format `shp`. Les données décrivent l'agriculture de départements français par quelques indicateurs. La question pourrait être simplement :

Proposez une analyse de la structure de l'agriculture française en vous basant sur les données subvention et en utilisant les outils de statistique et de cartographie dont vous disposez.

Nous vous proposons quelques pistes pour démarrer et des consignes de rendus de document.

2 Pour démarrer

2.1 Importer les données

Le fichier de tableur contient deux feuilles : les données sur la première et une rapide description des variables sur la seconde.

Comment importer ces données dans ? Il faut produire un fichier texte (format `*.txt` ou `*.csv`) avant de l'importer avec la commande `read.table()`. Plusieurs paramètres peuvent vous servir comme `header`, `sep` ou `dec` que vous connaissez déjà, ou encore `row.names` qui permet de préciser les noms des lignes.

Que faire de la seconde feuille ? Ne pas oublier de la lire posément pour l'interprétation.

```
> subv <- read.table("subve.csv",header=TRUE, ...)
> ?read.table # Si besoin
```

2.2 Contexte et description du jeu de données

N'oubliez pas de traiter la question 0 de toute étude statistique, c'est-à-dire de définir le contexte : population, individus, variables.

Pour les variables, prenez le temps de les comprendre. Vous ne disposez pas toujours des unités clairement définies mais vous devez néanmoins identifier les informations portées par chacune d'elles. Une attention particulière sera portée aux deux variables économiques.

L'analyse uni- et bivariée doit être votre entrée dans le jeu de données. Une variable et un individu posent problème. La variable par sa distribution très particulière, l'individu par une valeur extrême pour une des variables. Identifiez-les puis utilisez-les comme de l'information supplémentaire, hors de l'analyse multivariée notamment. Vos résultats en seront plus lisibles. Proposez une explication pour l'individu (l'hypothèse de l'erreur de frappe ne nous intéresse pas) et si possible pour la variable.

2.3 Traitements

Identifier les structures majeures de l'information contenues dans ce tableau. En particulier, que nous disent les différentes variables sur la répartition des subventions agricoles dont il est question dans cette étude ?

Pour ce faire vous réaliserez une typologie des départements avec les outils de l'analyse multivariée, approche factorielle et classification. Vous utiliserez l'outil cartographique afin de visualiser les résultats de la classification.

3 Document à rendre

Vous devez produire un rapport par binôme. Le document sera synthétique, structuré et illustré. Fournissez un script reproductible de votre analyse en annexe. Le corps du texte ne doit pas contenir de lignes de code. Les figures doivent être numérotées et légendées.

Le document, au format pdf et nommé d'après vos deux noms : `JULES-JIM-TDR33.pdf`, sera déposé sur e-campus, rubrique travaux, avant 10 jours.

"Je suis fort aise"

Analyse Factorielle & ses Correspondances

NA → Non Available (pas dispo)

Lien entre 2 variables qualitative → pas de corrélation mais des correspondances (cf. le χ^2)
à plusieurs modalités

Tableau un peu connu now

	1 ... j ... p	
1		
⋮		
i	n_{ij}	Σ
⋮		
n		
	Σ	

"si je peux faire une somme en ligne et une somme en colone, et que ça a un sens, on peut faire une AFC"

Décrire un ensemble de lignes caractérisées par un grand nombre de colonnes
→ Description des liens entre les lignes
→ Description des liens entre les colonnes
→ Description des proximités entre lignes et colonnes

[illustration : Nom de cigarette]

La contribution d'un point à l'axe dépend de la distance de son projeté sur l'axe et de son poids → oui, pour l'instant on a considéré que tous les poids valaient 1.

On a vu que les variables sont aussi liées et inversement

→ si des points en ligne sont les mêmes, les points en colonne sont les mêmes

Distance χ^2 : $d^2 = \sum \frac{\left(\frac{k_{ij}}{k_i} - \frac{k_{ij'}}{k_{i'}}\right)^2}{k_j / k}$

Analyse (factorielle) des correspondances multiples

A.C.M
A.F.C.M

Tableau de Burt

→ éviter les modalités trop différentes (4 critères à 2 modalités et 1 critère à 17 et 1 critère à 23 ça l'fait moyen)

Commandes usuelles de R

J. Chiquet, janvier 2010

Clear console : Ctrl+L

Cette liste de commandes (non exhaustive) est une adaptation de la ref-card R de Tom Short. Elle est un relais entre ce que cherche l'utilisateur et l'aide de R : les nombreuses options disponibles ne sont que rarement spécifiées ici.

Aide et fonctions de base

`help(topic)`, `?topic` affiche l'aide relative à `topic`
`apropos("topic")`, `??topic` recherche par mot-clé (version courte)
`help.search("topic")`, `???topic` recherche par mot-clé (version longue)
`help.start()` lance la version HTML de l'aide
`str(a)` affiche la structure de l'objet `a`
`head(a)` affiche les premiers éléments de l'objet `a` (selon son type: vecteur, matrice, tableau, etc.)
`summary(a)` propose un «résumé» de `a`, la plupart du temps un résumé statistique
`search()` affiche l'itinéraire de recherche
`ls()` affiche tous les objets présents dans l'itinéraire de recherche
`ls.str()` applique `str()` à variable présente dans l'itinéraire de recherche
`dir()` affiche les fichiers présents dans le répertoire courant
`library(x)` charge la bibliothèque `x`
`attach(x)` place l'objet `x` dans l'itinéraire de recherche; `x` peut être une liste, un tableau de données ou un objet créé à l'aide de la fonction `save`
`detach(x)` ôte l'objet `x` de l'itinéraire de recherche
`rm(x)`, `remove(x)` détruit l'objet `x`
`setwd(dir)`, `getwd(dir)` affecte ou récupère le chemin du répertoire de travail courant
`function( arglist ) { expr return(result)}` définition de fonction
`if`, `while`, `repeat`, `etc.` voir `help(if")`

Entrées / Sorties

`save(file,x,y)` enregistre les objets `x,y` dans le fichier binaire `file`
`save.image(file)` enregistre tous les objets de la session
`load(file)` charge un objet précédemment enregistré à l'aide de `save`
`data(x)` charge le jeu de données `x`
`read.table(file)`, `read.csv`, `read.delim` lit un fichier stocké sous la forme d'un tableau et crée un objet `data.frame`; le séparateur par défaut est le caractère espace pour `read.table`, la virgule ou le point virgule pour `read.csv`, la tabulation pour `read.delim`; utiliser l'option `header=TRUE` pour que la première ligne soit considérée comme définissant le nom des colonnes
`cat(...)` fonction d'impression bas niveau
`print(a)` fonction d'affichage d'un objet `a` s'adaptant au type de l'objet
`format(x)` formatage d'un objet
`write.table(x)` imprime `x` après conversion en type `data.frame`

Création de données

`c(...)` fonction générique combinant une suite d'éléments en un vecteur
`from:to` génère une séquence; priorité de l'opérateur «:» `1:4 + 1` vaut `2,3,4,5`
`seq(from,to)` génère une séquence; `by=` et `length=` spécifient l'incrément et/ou la longueur
`seq(along=x)` génère `1, 2, ..., length(x)`; utile lors de boucle `for`
`rep(x,times)` répète `x` un nombre `times` de fois; utiliser `each=` pour répéter chaque élément `each` fois; `each` peut être un vecteur
`data.frame(...)` crée un tableau de données; les vecteurs courts sont répétés jusqu'à correspondre à la taille des vecteurs les plus longs
`list(...)` crée une liste
`array(x,dim=)` crée un tableau multidimensionnel `x`; les éléments de `x` sont répétés si la taille ne correspond pas aux dimensions spécifiées
`matrix(x,nrow=,ncol=)` crée une matrice; les éléments de `x` sont répétés si la taille ne convient pas
`factor(x,levels=)` crée un vecteur de facteurs
`expand.grid()` génère un tableau de données contenant les combinaisons des vecteurs spécifiés en arguments
`rbind()`, `cbind()` pour combiner les éléments d'un objet par ligne et par colonne

Extraction de données

Indexation des listes

`x[n]`	une liste avec les éléments de `n`
`x[[n]]`	le n^e élément de la liste
`x$name`, `x[["name"]]`	l'élément `"name"`

Indexation des vecteurs

`x[n]`	n^e élément du vecteur
`x[-n]`	tous les éléments sauf le n^e
`x[1:n]`	`n` premiers éléments
`x[-(1:n)]`	tous les éléments sauf les `n` premiers
`x[c(1,4,2)]`	éléments 1,4 et 2
`x["name"]`	élément(s) de nom `"name"`
`x[x > 3]`	tous les éléments plus grands que 3
`x[x > 3 & x < 5]`	tous les éléments compris entre 3 et 5

Indexation des matrices

`x[i,j]`	élément de la i^e ligne et j^e colonne
`x[i,]`	i^e ligne
`x[,j]`	j^e colonne
`x[,c(1,3)]`	colonnes 1 et 3
`x["name",]`	lignes intitulées `"name"`
`x[rowSums(x)>10,]`	lignes dont la somme est supérieure à 10

Variables et attributs

`as.array(x)`, `as.data.frame(x)`, `as.numeric(x)`, `as.logical(x)`, `as.character(x)`, ... conversion de type
`is.na(x)`, `is.null(x)`, `is.array(x)`, `is.data.frame(x)`, `is.numeric(x)`, `is.character(x)`, ... teste le type d'un objet

Les fonctions suivantes s'utilisent pour récupérer ou spécifier un attribut.

`length(x)` nombre d'éléments de `x`
`dim(x)` nombre de dimensions d'un objet
`dimnames(x)` noms des dimensions d'un objet
`nrow(x)`, `ncol(x)` nombre de lignes et de colonnes
`class(x)` classe de l'objet `x`
`unclass(x)` supprime l'attribut `class` de la variable `x`
`attr(x,which)` récupère ou spécifie les attributs de `x` décrits par `which`
`attributes(x)` récupère ou spécifie tous les attributs de `x`

Manipulation et sélection de données

`which.max(x)`, `which.min(x)` retourne l'indice du plus grand (resp. plus petit) élément de `x`

`rev(x)` inverse l'ordre des éléments `x`

`sort(x)` ordonne les éléments de `x` par ordre croissant

`cut(x,breaks)` découpe `x` en intervalles définis par `breaks`

`match(x, y)` renvoie un vecteur de la même taille que `x` dont l'élément i vaut `x[i]` si `x[i]` appartient à `y` et `NA` sinon

`which(x==a)` renvoie les indices des éléments de `x` vérifiant `x==a`

`choose(n, k)` calcule les combinaisons de k éléments parmi n

`na.omit(x)` supprime les observations manquantes (notées `NA`)

`na.fail(x)` renvoie une erreur si `x` contient au moins un `NA`

`unique(x)` supprime les doublons d'un vecteur ou d'un tableau

`table(x)` renvoie un tableau avec le nombre des différentes valeurs

`subset(x, ...)` renvoie un sous ensemble de `x` défini par `...`

`sample(x, size)` crée un échantillon aléatoire de taille `size` parmi les éléments de `x`

Mathématiques

`abs`, `sqrt`, `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`, `log`, `log10`, `exp`, ... fonctions mathématiques élémentaires

`max(x)`, `min(x)`, `range(x)`, `sum(x)`, `diff(x)`, `prod(x)`, `mean(x)`, `median(x)`, `sd(x)` maximum, minimum, amplitude, somme, différences, produit, moyenne, médiane, écart-type

`quantile(x,probs=)` fractiles des éléments de `x`

`weighted.mean(x, w)` moyenne de `x` pondérée par `w`

`var(x)`, `cov(x)` variance empirique corrigée; si `x` est une matrice, renvoie la matrice de variance-covariance

`cor(x)` matrice de corrélations de `x`

`var(x, y)`, `cov(x, y)` covariance entre `x` et `y`, ou entre les colonnes de `x` et de `y` si ce sont des matrices ou des tableaux

`cor(x, y)` idem pour la corrélation linéaire

`round(x, n)` arrondit les éléments de `x` à `n` décimales

`scale(x)` centre et réduit les données `x`; pour centrer et/ou réduire uniquement, utiliser les `scale` et/ou `center`

`pmin(x,y,...)`, `pmax(x,y,...)` un vecteur dont le i^e élément est le minimum (resp. maximum) entre `x[i]` et `y[i]`

`cumsum(x)` un vecteur dont le i^e élément est la somme des i premiers éléments de `x`

`cumprod(x)`, `cummin(x)`, `cummax(x)` idem pour le produit, le min, le max

`union(x,y)`, `intersect(x,y)`, `setdiff(x,y)`, `setequal(x,y)` et `is.element(el,set)` fonctions de définition d'ensembles

`Re(x)`, `Im(x)`, `Mod(x)`, `Arg(x)`, `Conj(x)` partie réelle, partie imaginaire, module, argument et conjugué d'un nombre complexe

`convolve(x,y)` calcule de convolution entre deux séquences

`fft(x)`, `mvfft(x)` transformation de Fourier d'une matrice, resp des colonnes d'une matrice

`filter(x,filter)` application d'un filtre linéaire à chaque élément d'une suite `x`

Matrices

`rowSums(x)`, `colSums(x)`, `rowMeans(x)`, `colMeans(x)` somme et moyenne de chaque ligne, resp. chaque colonne de `x`

`t(x)` transposée de `x`

`diag(x)` renvoie ou spécifie la diagonale de `x`

`%*%` multiplication matricielle

`crossprod(x,y)`, `t(x)%*%y` produit scalaire de `x` par `y`

`svd(x)` décomposition en valeurs singulières

`eigen(x)` diagonalisation d'une matrice

`chol(x)` décomposition de Cholesky

`qr(x)` décomposition QR

`solve(a,b)` résout `a %*% x = b`

`solve(a)` calcule l'inverse de `a`

Traitements avancés

`apply(x,INDEX,FUN=)` renvoie un vecteur ou une liste de valeurs obtenues en appliquant la fonction `FUN` aux éléments de la dimension `INDEX` de `x`

`lapply(x,FUN)` applique `FUN` aux éléments d'une liste

`tapply(x,INDEX,FUN=)` applique `FUN` à chaque groupe du tableau `X` défini par les indices `INDEX`

`by(data,INDEX,FUN)` applique `FUN` au tableau de données `data` découpé via `INDEX`

`ave(x,...,FUN)` applique `FUN` à chaque sous-ensemble de `x` définis par des facteurs

`merge(a,b)` fusion de deux tableaux de données portant les mêmes noms de ligne ou de colonne

`aggregate(x,by,FUN)` découpe le tableau `x` en sous-ensembles auxquels sont appliqués la fonction `FUN` et renvoie le résultat; `by` est une liste définissant les sous-ensembles de `x`

`stack(x, ...)`, `unstack(x, ...)` transforme un tableau ou une liste `x` en un vecteur colonne, et réciproquement

Chaînes de caractères

`paste(...)` concaténation de vecteurs après conversion en caractères

`substr(x,start,stop)` extraction ou spécification d'une sous-chaîne de `x`

`strsplit(x,split)` découpe `x` selon la sous-chaîne `split`

`grep(pattern,x)` recherche le motif `pattern` dans la chaîne `x`

`tolower(x)`, `toupper(x)` conversion en minuscules, resp. en majuscules

`match(x,table)` un vecteur renvoyant les positions où les éléments de `x` ont été pour la première fois rencontrés dans `table`

`x %in% table` identique, mais renvoie un vecteur de booléens

`pmatch(x,table)` appariement partiel des éléments de `x` parmi `table`

`nchar(x)` nombre de caractères de `x`

Graphiques et figures

`x11()`, `windows()` ouvre une nouvelle fenêtre graphique

`pdf()`, `png()`, `jpeg()`, `bitmap()`, `xfig()`, `pictex()`, `postscript()` pilote graphique produisant des sorties dans des fichiers plutôt qu'à l'écran

`dev.off()` ferme le pilote de sortie graphique pour clore le fichier de sortie

Commandes graphiques haut niveau

`plot(x)` trace les valeurs contenues dans `x` sur l'axe des y; s'adapte à la classe de l'objet `x`

`plot(x, y)` graphe bivarié (`x` sur l'axe des x, `y` sur l'axe des y)

`hist(x)` histogramme des fréquences de `x`

`curve(expr)` trace la fonction définie par l'expression `expr`

`barplot(x)` histogramme des valeurs de `x`

`dotchart(x)` si `x` est un tableau de données, trace les données par nuages de points groupés par ligne en ordonnées puis par colonne en abscisses

`pie(x)` graphe en camembert

`boxplot(x)` boîte à moustaches

`interaction.plot (f1, f2, y)` si `f1` et `f2` sont des facteurs, trace les moyennes de `y` en fonction des valeurs de `f1` et `f2` sur deux courbes différentes

`matplot(x,y)` graphe bivarié traçant la première colonne de `x` *vs.* la première colonne de `y`, puis la deuxième colonne de `x` *vs.* la deuxième colonne de `y`, etc.

`assocplot(x)` graphe d'association indiquant à quelle point les colonnes et lignes du tableau de contingence `x` dévient de l'hypothèse d'indépendance
`mosaicplot` graphe mosaïque des résidus d'un modèle log-linéaire
`pairs(x)` trace tous les graphes bivariés possibles entre les colonnes du tableau de données `x`
`plot.ts(x)` si `x` est de classe `"ts"` (time-serie), trace `x` en fonction du temps
`qqnorm(x)` fractiles de `x` en fonction des valeurs attendues sous l'hypothèse gaussienne
`qqplot(x, y)` fractiles de `y` en fonction des fractiles de `x`
`contour(x, y, z)`, `image(x, y, z)`, `persp(x, y, z)` variantes pour tracer les données de la matrice `z` en fonction des vecteurs `x` et `y`
`symbols(x, y, ...)` trace aux coordonnées spécifiées par `x` et `y` des cercles, carrés, rectangles, étoiles, boîtés à moustaches, etc.
`termplot(mod.obj)` trace les termes d'un modèle de régression en fonction des prédicteurs

Paramètres récurrents des fonctions graphiques

`add=FALSE` si `TRUE` superpose le graphe au précédent
`axes=TRUE` si `FALSE` ne trace pas d'axes
`type="p"` spécifie le type de tracé: `"p"` pour points, `"l"` pour lignes, `"b"` pour points liés par des lignes, `"o"` pour lignes superposées aux points, `"h"` pour lignes verticales, `"s"` ou `"S"` pour fonction en escaliers
`xlim=`, `ylim=` spécifie les limites des axes x et y
`xlab=`, `ylab=` annotation des axes x et y
`main=` titre du graphe en cours
`sub=` sous-titre du graphe en cours

Commandes graphiques bas-niveau

`points(x, y)` ajoute des points aux coordonnées `x` et `y`
`lines(x, y)` trace `y` en fonction de `x`
`text(x, y, labels, ...)` ajoute le texte `labels` aux coordonnées `(x,y)`
`mtext(text, side=3, ...)` ajoute le texte `text` dans la marge `side`
`segments(x0, y0, x1, y1)` trace des lignes des points `(x0,y0)` aux points `(x1,y1)`
`arrows(x0, y0, x1, y1)` identique mais avec des flèches
`abline(a,b)` trace une droite de pente `b` et de décalage `a` par rapport à l'axe des x
`abline(h=y)` trace une ligne horizontale à l'ordonnée `y`
`abline(v=x)` trace une ligne verticale à l'abscisse `x`
`rect(x1, y1, x2, y2)` trace un rectangle défini par `x1`, `x2`, `y1` et `y2`
`polygon(x, y)` trace un polygone en liant les points de coordonnées définies dans les vecteurs `x` et `y`
`legend(x, y, legend)` ajoute une légende au point `(x,y)` spécifié par `legend`
`title()` ajoute un titre et éventuellement un sous-titre
`axis(side)` fonction de bas niveau pour gérer les axes de la figure
`box()` trace un cadre autour de la figure courante

Paramètres graphiques de bas de niveau

Ces paramètres sont définis à l'aide de la commande `par(...)` ou directement par passage à la fonction graphique d'appel

`adj` contrôle la justification du texte
`bg` spécifie la couleur de fond
`bty` contrôle le type de cadre tracé autour de la figure
`cex` contrôle la taille du texte et des symboles
`col` contrôle la couleur des symboles et des courbes (entier ou chaîne de caractères)
`font` un entier contrôlant le style de la police
`las` un entier contrôlant l'orientation des annotations des axes
`lty` contrôle le type de ligne (entier ou chaîne de caractère)
`lwd` contrôle l'épaisseur des lignes
`mar` contrôle l'espace entre le tracé et les bordures de la fenêtre
`mfc` un vecteur de la forme `c(nr,nc)` qui partitionne la fenêtre graphique en `nr` lignes et `nc` colonnes, les graphes étant tracés par colonne.
`mfrow` identique mais les graphes sont tracés par ligne
`pch` contrôle le type de symbole:

1 ○ 2 △ 3 + 4 × 5 ◇ 6 ▽ 7 ⊠ 8 ✳ 9 ⊕ 10 ⊛ 11 ⊠ 12 ⊞ 13 ⊠ 14 ⊡ 15 ■
16 ● 17 ▲ 18 ◆ 19 ● 20 ● 21 ○ 22 □ 23 ◇ 24 △ 25 ▽ * • . · X X a a ? ?

`ps` un entier contrôlant la taille du texte et des symboles

Optimisation, ajustement, statistique

`optimize(fn,interval)` méthode d'optimisation pour les fonctions unidimensionnelles
`optim(par, fn)` méthode d'optimisation générique minimisant la fonction `fn` en partant de la valeur `par` des coefficients
`nlm(f,p)` minimise la fonction `f` à l'aide d'un algorithme type Newton, partant de la valeur `p` pour les coefficients
`lm(formula)` ajuste un modèle linéaire; `formula` est typiquement de la forme `response ~ termA + termB + ...`
`glm(formula,family=)` ajuste un modèle linéaire généralisé
`nls(formula)` estimateur non-linéaire des moindres carrés des paramètres d'un modèle non-linéaire
`approx(x,y=)` interpolation linéaire
`spline(x,y=)` interpolation par splines cubiques
`aov`, `anova` fonction d'analyse de la variance
`density(x)` estimateur à noyaux de la densité de `x`
`t.test()`, `pairwise.t.test()`, `chisq.test()`, `binom.test()`, `prop.test()`, `power.t.test()`, ... utiliser `help.search("test")` pour voir l'ensemble des tests statistiques disponibles

Attributs d'un modèle

Les fonctions ci-dessus renvoient un objet modèle dont l'ajustement dépend de la méthode utilisée. Certains des attributs de cet objet sont évalués à l'aide des commandes suivantes.

`df.residual(fit)` renvoie le nombre de degrés de liberté résiduels de `fit`
`coef(fit)` renvoie les coefficients estimés de `fit`
`residuals(fit)` renvoie les résidus
`fitted(fit)` renvoie les valeurs ajustées
`logLik(fit)` calcule la log-vraisemblance du modèle et le nombre de paramètres
`AIC(fit)` calcule le critère AIC (Akaike information criterion)

Distributions

Toutes les fonctions suivantes peuvent s'utiliser en remplaçant la lettre `r` avec `d`, `p` ou `q` pour obtenir, respectivement, un tirage de n réalisations d'une variable aléatoire, la densité de probabilité, la fonction de répartition, et la valeur des fractiles.

`rnorm(n, mean=0, sd=1)` gaussienne
`rexp(n, rate=1)` exponentielle
`rgamma(n, shape, scale=1)` Gamma
`rpois(n, lambda)` Poisson
`rweibull(n, shape, scale=1)` Weibull
`rcauchy(n, location=0, scale=1)` Cauchy
`rbeta(n, shape1, shape2)` Beta
`rt(n, df)` Student
`rf(n, df1, df2)` Fisher–Snedecor (F)
`rchisq(n, df)` Pearson (χ^2)
`rbinom(n, size, prob)` binomiale
`rgeom(n, prob)` geometrique
`rhyper(nn, m, n, k)` hypergeometrique
`rlogis(n, location=0, scale=1)` logistique
`rlnorm(n, meanlog=0, sdlog=1)` lognormale
`rnbinom(n, size, prob)` binomiale negative
`runif(n, min=0, max=1)` uniforme
`rwilcox(nn, m, n)`, `rsignrank(nn, n)` Statistique de Wilcoxon

barplot(rev(sort(table(xi))))