Introduction à R... en 1h45

Ecole de Bioinformatique AVIESAN-IFB, Roscoff 2017

Hugo Varet (Institut Pasteur)

2017-11-14

Contents

Quelques liens utiles

- Projet R: http://www.r-project.org
- Télécharger R + des bibliothèques de fonctions: http://cran.r-project.org
- L'environnement de travail RStudio: https://www.rstudio.com

- Bioconductor, des bibliothèques de fonction pour la biologie: http://www.bioconductor.org
- R cheat sheeet (antisèche): https://www.rstudio.com/wp-content/uploads/2016/10/r-cheat-sheet-3.pdf
- R basics and easy: http://www.sthda.com/english/wiki/r-basics-quick-and-easy
- Google R style guide: https://google.github.io/styleguide/Rguide.xml#identifiers

R vu comme une calculatrice

Le symbole ">" est l'invite de commande ("prompt") de **R**. Ilsignifie que R est prêt à recevoir une nouvelle instruction. Exemple:

```
## Quelques calculs
2 + 3
```

```
## [1] 5
```

```
2 * 3
```

```
## [1] 6
```

```
2 / 3
```

```
## [1] 0.6666667
```

Terminez votre commande !

Quand vous entrez une commande incomplète, R affiche une invite `+`, qui indique qu'il attend que vous complétiez votre commande

```
2 *
```

Si vous êtes coincés avec une invite `+`, vous devez terminer la commande pour vous tirer d'affaire. Dans ce cas-ci, tapez par exemple taper `7` puis ENTER.

Pendant le calcul, pas d'invite de commande

Si on envoie à R un calcul qui dure un certain temps, pendant le calcul il n'affiche aucun symbole d'invite de commande.

Un exemple de commande qui prend quelques secondes (nous expliquerons plus bas en quoi consistent ces fonctions).

Note: le symbole `#` permet d'écrire des commentaires, i.e. du texte non interprété par R.

```
## Calculer la moyenne de 100 millions de nombres
## tirés au hasard selon une distribution normale
mean(rnorm(100000000))
```

```
## [1] -4.529952e-05
```

Pendant quelques secondes rien ne se passe ... puis la réponse s'affiche.

Utiliser des variables

Le symbole `<-` assigne une valeur à une variable. Si cette variable n'a pas encore déclarée, elle est créée au passage (un espace mémoire lui est réservé).

```
a <- 2  ## Assignation d'une valeur à une variable
b = 3   ## Notation équivalente acceptée mais pas recommandée
```

Nommer ses variables avec des noms informatifs

Plutôt que a,b,c , on recommande de nommer les varables de façon explicite

```
resultat <- a * b + a + b  ## Le résultat est stocké dans une variable nommée resultat
print(resultat)
```

```
## [1] 11
```

Attention ! Les noms de variables sont sensibles à la casse ! La commande suivante retourne une erreur.

```
print(Resultat)
```

Error in print(Resultat) : object 'Resultat' not found

Ne pas exagérer dans la longueur des noms de variables sinon cela nuit à la lisibilité du code, comme quand on fait des titres à rallonge qui n'aident pas non plus

```
ma_nouvelle_variable_avec_un_nom_a_rallonge <- 10
```

Noms réservés

Attention: certains noms sont déjà utilisés pour des fonctions/objets **R**. Il ne faut donc pas les utiliser comme noms de variables. On les qualifie de **noms réservés**.

```
c # créer un vecteur
t # transposer une matrice
sum # fonction somme
mean # fonction moyenne

TRUE # booléen
FALSE # booléen
T # booléen
F # booléen

NA # not available
NaN # not a number (par exemple log(-1))
Inf # infini (par exemple 1/0)
NULL # objet nul

pi # 3.14...
letters # 26 lettres minuscules
LETTERS # 26 lettres majuscules
```

Types de données

Données numériques

Par défaut les nombres sont considérés comme des variables numériques de type "floating point" (variables réelles), même si leur valeur particulière est entière.

```
# "double precision floating point numbers"
x <- 3
print(x)
```

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

```
mode(x)
```

```
## [1] "numeric"
```

```
typeof(x)
```

```
## [1] "double"
```

On peut cependant explicitement déclarer une variable entière avec `as.integer()`

```
y <- as.integer(2)
print(y)
```

```
## [1] 2
```

```
mode(y)
```

```
## [1] "numeric"
```

```
typeof(y)
```

```
## [1] "integer"
```

Intérêt: les variables de type "Integer" utilisent moins d'espace mémoire que les "Floating point".

Chaînes de charactères

Les variables de type "character" permettent de stocker des chaînes de caractères.

```
x <- "chaîne de caractères, toujours entre guillemets"
print(x)
```

```
## [1] "chaîne de caractères, toujours entre guillemets"
```

```
y <- 'ou avec des guillemets simples'
print(y)
```

```
## [1] "ou avec des guillemets simples"
```

```
mode(x)                 ## type de contenu de la variable x
```

```
## [1] "character"
```

```
nchar(x)                ## lontgueur d'une chaîne de caractères
```

```
## [1] 47
```

```
paste(x, y, sep=" / ")   ## concaténation de variables de type character
```

```
## [1] "chaîne de caractères, toujours entre guillemets / ou avec des guillemets simples"
```

Variables Booléennes (logiques, binaires)

```
x <- TRUE # ou T ou FALSE ou F
print(x)
```

```
## [1] TRUE
```

```
mode(x)
```

```
## [1] "logical"
```

```
# sera utile pour tester des choses :
# if (condition) {
#   faire ça
# } else {
#   faire autre chose
# }
```

Structures de données

Vecteurs

Un vecteur permet de stocker dans une seule variable une liste d'élements du même type (numeric, character...).

Un vecteur peut contenir des valeurs de différents types: entiers, réells, Booléens, char, ...

```
u <- c(2, 4, 5, 1)
print(u)
```

```
## [1] 2 4 5 1
```

```
v <- c(10, 5, 2, 2)
print(v)
```

```
## [1] 10  5  2  2
```

```
w <- c("Pierre", "Paul", "Jacques", "Henri")
print(w)
```

```
## [1] "Pierre"  "Paul"    "Jacques" "Henri"
```

```
l <- c(TRUE, FALSE, FALSE, TRUE)
print(l)
```

```
## [1]  TRUE FALSE FALSE  TRUE
```

Caractéristiques d'un vecteur

```
length(u)
```

```
## [1] 4
```

```
mode(u)
```

```
## [1] "numeric"
```

```
mode(w)
```

```
## [1] "character"
```

```
mode(l)
```

```
## [1] "logical"
```

Sélection/suppression d'éléments d'un vecteur

```
u[2]
```

```
## [1] 4
```

```
u[-2]
```

```
## [1] 2 5 1
```

```
u[c(1, 3)]
```

```
## [1] 2 5
```

```
u[l]
```

```
## [1] 2 1
```

```
u >= 3
```

```
## [1] FALSE  TRUE  TRUE FALSE
```

```
u[which(u >= 3)]
```

```
## [1] 4 5
```

```
v %in% c(2, 5)
```

```
## [1] FALSE  TRUE  TRUE  TRUE
```

```
v[which(v %in% c(2, 5))]
```

```
## [1] 5 2 2
```

Opérations sur des vecteurs

```
sort(u)
```

```
## [1] 1 2 4 5
```

```
summary(u)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.75    3.00    3.00    4.25    5.00
```

```
sum(u)
```

```
## [1] 12
```

```
mean(u)
```

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

```
u + v
```

```
## [1] 12  9  7  3
```

```
u * v
```

```
## [1] 20 20 10  2
```

```
u / v
```

```
## [1] 0.2 0.8 2.5 0.5
```

```
cbind(u, v)
```

```
##      u  v
## [1,] 2 10
## [2,] 4  5
## [3,] 5  2
## [4,] 1  2
```

```
rbind(u, v)
```

```
##   [,1] [,2] [,3] [,4]
## u    2    4    5    1
## v   10    5    2    2
```

Création de séquences de nombres

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(from=1, to=100, by=10)
```

```
##  [1]  1 11 21 31 41 51 61 71 81 91
```

```
seq(from=1, to=100, length=4)
```

```
## [1]   1  34  67 100
```

```
rep(x=c(1, 2), times=3)
```

```
## [1] 1 2 1 2 1 2
```

```
rep(x=c(1, 2), each=3)
```

```
## [1] 1 1 1 2 2 2
```

Valeurs non définies (NA)

```
u <- c(4, NA, 5, 2, NA, 3)
mean(u)
```

```
## [1] NA
```

```r
mean(u, na.rm=TRUE)
```

```
## [1] 3.5
```

Facteurs

Les structures de données de type `factor` permettent de traiter des listes d'éléments dont les valeurs appartiennent à un ensemble défini. Ces valeurs sont indexées pour assurer une efficacité maximale du traitement informatique.

```r
## Un vecteur
mentions <- c("Passable", "AB", "AB", "B", "TB", "TB", "Passable", "B", "Passable", "TB")
print(mentions)
```

```
##  [1] "Passable" "AB"       "AB"       "B"        "TB"       "TB"
##  [7] "Passable" "B"        "Passable" "TB"
```

```r
# vecteur avec des catégories
f <- factor(mentions)
print(f)    # Afficher le contenu du facteur
```

```
##  [1] Passable AB       AB       B        TB       TB       Passable
##  [8] B        Passable TB
## Levels: AB B Passable TB
```

```r
levels(f)   # Enumérer les catégories du facteur
```

```
## [1] "AB"       "B"        "Passable" "TB"
```

```r
table(f)    # Dénombrer les éléments par catégorie
```

```
## f
##       AB        B Passable       TB
##        2        2        3        3
```

Définir un ordre sur les niveaux d'un facteur

```r
# notion d'ordre
f <- factor(mentions, levels=c("Passable", "AB", "B", "TB"))
print(f)    # Afficher le contenu du facteur
```

```
##  [1] Passable AB       AB       B        TB       TB       Passable
##  [8] B        Passable TB
## Levels: Passable AB B TB
```

```r
levels(f)   # Enumérer les catégories du facteur
```

```
## [1] "Passable" "AB"       "B"        "TB"
```

```r
table(f)    # Dénombrer les éléments par catégorie
```

```
## f
## Passable       AB        B       TB
##        3        2        2        3
```

Traitement des valeurs non-assignées (NA)

```
# NA et catégories non représentées
f <- factor(c("Passable", NA, "AB", "AB", NA, "B"), levels=c("Passable", "AB", "B", "TB"))
print(f)
```

```
## [1] Passable <NA>     AB       AB       <NA>     B
## Levels: Passable AB B TB
```

```
levels(f)
```

```
## [1] "Passable" "AB"       "B"        "TB"
```

```
table(f)   # Où sont passées les valeurs NA ?
```

```
## f
## Passable       AB        B       TB
##        1        2        1        0
```

```
## Imprimer les valeurs NA seulement si elles existent
table(f, useNA="ifany")
```

```
## f
## Passable       AB        B       TB     <NA>
##        1        2        1        0        2
```

Matrices

```
# tableau à N lignes et P colonnes avec des éléments du même type (numeric, character...)
m <- matrix(rnorm(30), nrow=6, ncol=5)
print(m)
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] -1.4370079 -1.1747119 -0.7151555 -1.0229298 -0.6813862
## [2,] -1.5765645 -0.7937732 -0.1152495 -1.0040834 -0.8254008
## [3,]  1.3343920 -2.4725081  1.5455722 -0.2632661 -1.2389841
## [4,]  1.5845029  2.2426697 -1.0135324  1.1583605 -0.3448616
## [5,] -0.4044907  0.4908001  0.7691316 -1.5360332 -0.5070376
## [6,]  0.4434907  0.9738929  0.4354720 -1.3661237 -0.5010609
```

```
n <- matrix(c("a","b","c","d","e","f"), ncol=3, byrow=TRUE)
print(n)
```

```
##      [,1] [,2] [,3]
## [1,] "a"  "b"  "c"
## [2,] "d"  "e"  "f"
```

Caractéristiques d'une matrice

```
# caractéristiques d'une matrice
ncol(m)
```

```
## [1] 5
```

```r
nrow(m)
```

```
## [1] 6
```

```r
dim(m)
```

```
## [1] 6 5
```

```r
length(m)
```

```
## [1] 30
```

```r
mode(m)
```

```
## [1] "numeric"
```

```r
mode(n)
```

```
## [1] "character"
```

Opérations sur des matrices

```r
# opérations sur des matrices
mat1 <- matrix(1:6, nrow=2, ncol=3)
mat2 <- matrix(rnorm(6), nrow=2, ncol=3)
print(mat1)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```r
print(mat2)
```

```
##            [,1]        [,2]       [,3]
## [1,]  0.3284217  0.04969546 0.69113822
## [2,] -0.8709496 -1.93531514 0.04584997
```

Elément par élément

```r
# élément par élément
mat1 + mat2
```

```
##          [,1]     [,2]     [,3]
## [1,] 1.328422 3.049695 5.691138
## [2,] 1.129050 2.064685 6.045850
```

```r
mat1 * mat2
```

```
##            [,1]       [,2]      [,3]
## [1,]  0.3284217  0.1490864 3.4556911
## [2,] -1.7418992 -7.7412606 0.2750998
```

Produit matriciel (attention aux dimensions)

```r
# produit matriciel (attention aux dimensions)
mat1 %*% t(mat2)
```

```
##          [,1]      [,2]
## [1,] 3.933199 -6.447645
## [2,] 5.002455 -9.208060
```

```
t(mat1) %*% mat2
```

```
##           [,1]       [,2]      [,3]
## [1,] -1.413477  -3.820935 0.7828382
## [2,] -2.498533  -7.592174 2.2568146
## [3,] -3.583589 -11.363414 3.7307909
```

Somme, moyenne, ...

```
# somme, moyenne, ...
sum(mat1) # de tous les éléments
```

```
## [1] 21
```

```
rowSums(mat1) # des éléments de chaque ligne
```

```
## [1]  9 12
```

```
colSums(mat1) # des éléments de chaque colonne
```

```
## [1]  3  7 11
```

```
mean(mat1)
```

```
## [1] 3.5
```

```
rowMeans(mat1)
```

```
## [1] 3 4
```

```
colMeans(mat1)
```

```
## [1] 1.5 3.5 5.5
```

Sélection/suppression d'éléments d'une matrice

```
# sélection/suppression d'éléments d'une matrice
mat1[, c(2, 3)]
```

```
##      [,1] [,2]
## [1,]    3    5
## [2,]    4    6
```

```
mat1[1,]
```

```
## [1] 1 3 5
```

```
mat1[1, c(2, 3)]
```

```
## [1] 3 5
```

Structures de données de type liste

```
# permet de stocker des objets de types/longueurs différents
l1 <- list(n = c(TRUE, FALSE),
           v = c(3, 4),
           r = c("toto","plop", "tutu"),
           mat1)
print(l1)
```

```
## $n
## [1]  TRUE FALSE
##
## $v
## [1] 3 4
##
## $r
## [1] "toto" "plop" "tutu"
##
## [[4]]
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
length(l1)
```

```
## [1] 4
```

```
names(l1)
```

```
## [1] "n" "v" "r" ""
```

Liste contenant une liste

```
# liste contenant une liste
l2 <- list(a="chaîne de caractères", l1=l1)
print(l2)
```

```
## $a
## [1] "chaîne de caractères"
##
## $l1
## $l1$n
## [1]  TRUE FALSE
##
## $l1$v
## [1] 3 4
##
## $l1$r
## [1] "toto" "plop" "tutu"
##
## $l1[[4]]
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Extraction d'éléments d'une liste

```
# extraction d'éléments d'une liste
l1$n
```

```
## [1]  TRUE FALSE
```

```
l2$l1
```

```
## $n
## [1]  TRUE FALSE
##
## $v
## [1] 3 4
##
## $r
## [1] "toto" "plop" "tutu"
##
## [[4]]
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
l2$l1$v
```

```
## [1] 3 4
```

```
l2[1]
```

```
## $a
## [1] "chaîne de caractères"
```

```
l2[[1]]
```

```
## [1] "chaîne de caractères"
```

```
is.list(l2[1])
```

```
## [1] TRUE
```

```
is.list(l2[[1]])
```

```
## [1] FALSE
```

Strctures de données de type data.frame

```
# tableau dont les colonnes ne sont pas nécessairement du même type (numeric, character...)
d <- data.frame(nom=c("Pierre", "Paul", "Henri", "Mathieu"),
                taille=c(165, 168, 163, 170),
                poids=c(58, 60, 62, 68))
print(d)
```

```
##       nom taille poids
## 1  Pierre    165    58
## 2    Paul    168    60
## 3   Henri    163    62
## 4 Mathieu    170    68
```

```r
class(d)
```

```
## [1] "data.frame"
```

```r
typeof(d) # une data.frame est un cas particulier d'une liste
```

```
## [1] "list"
```

Caractéristiques d'une data.frame

```r
# caractéristiques d'une data.frame
ncol(d)
```

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

```r
nrow(d)
```

```
## [1] 4
```

```r
names(d)
```

```
## [1] "nom"    "taille" "poids"
```

```r
length(d)
```

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

Sélection de colonnes

```r
# sélection de colonnes
d$taille
```

```
## [1] 165 168 163 170
```

```r
d[, "taille"]
```

```
## [1] 165 168 163 170
```

```r
d[, c("nom", "taille")]
```

```
##       nom taille
## 1  Pierre    165
## 2    Paul    168
## 3   Henri    163
## 4 Mathieu    170
```

Sélection de lignes

```r
# sélection de lignes
d[c(1, 3:4), ]
```

```
##       nom taille poids
## 1  Pierre    165    58
## 3   Henri    163    62
## 4 Mathieu    170    68
```

```r
d$nom == "Pierre"
```

```
## [1]  TRUE FALSE FALSE FALSE
```

```r
d[which(d$nom == "Pierre"),]
```

```
##      nom taille poids
## 1 Pierre    165    58
```

Résumé statistique

```r
# résumé statistique
summary(d)
```

```
##       nom        taille          poids
##  Henri  :1   Min.   :163.0   Min.   :58.0
##  Mathieu:1   1st Qu.:164.5   1st Qu.:59.5
##  Paul   :1   Median :166.5   Median :61.0
##  Pierre :1   Mean   :166.5   Mean   :62.0
##              3rd Qu.:168.5   3rd Qu.:63.5
##              Max.   :170.0   Max.   :68.0
```

Fusion de deux data frames

```r
# fusion de deux data frames
d2 <- data.frame(nom=c("Paul","Henri","Louis"), age=c(34, 29, 47))
print(d2)
```

```
##     nom age
## 1  Paul  34
## 2 Henri  29
## 3 Louis  47
```

```r
merge(x=d, y=d2, by="nom")
```

```
##     nom taille poids age
## 1 Henri    163    62  29
## 2  Paul    168    60  34
```

```r
merge(x=d, y=d2, by="nom", all=TRUE)
```

```
##       nom taille poids age
## 1   Henri    163    62  29
## 2 Mathieu    170    68  NA
## 3    Paul    168    60  34
## 4  Pierre    165    58  NA
## 5   Louis     NA    NA  47
```

```r
merge(x=d, y=d2, by="nom", all.x=TRUE)
```

```
##       nom taille poids age
## 1   Henri    163    62  29
## 2 Mathieu    170    68  NA
## 3    Paul    168    60  34
## 4  Pierre    165    58  NA
```

```
merge(x=d, y=d2, by="nom", all.y=TRUE)
```

```
##     nom taille poids age
## 1 Henri    163    62  29
## 2  Paul    168    60  34
## 3 Louis     NA    NA  47
```

Création d'une nouvelle variable

```
# création d'une nouvelle variable
print(d)
```

```
##       nom taille poids
## 1  Pierre    165    58
## 2    Paul    168    60
## 3   Henri    163    62
## 4 Mathieu    170    68
```

```
d$age <- c(35, 42, 31, 28)
d$classe_poids <- ifelse(test=d$poids >= 60, yes=">=60", no="<60")
print(d)
```

```
##       nom taille poids age classe_poids
## 1  Pierre    165    58  35          <60
## 2    Paul    168    60  42         >=60
## 3   Henri    163    62  31         >=60
## 4 Mathieu    170    68  28         >=60
```

A l'aide

Aide dans R

```
help(read.table)
?read.table
```

Aide sur le web

- Google !!!
- plein de forums dédiés: par exemple https://stackoverflow.com/
- mailing list: https://www.r-project.org/mail.html
- spéficique Bioconductor: https://support.bioconductor.org/

Aide ici

- les formateurs !

Espace de travail

Où suis-je ?

La commande **R** `getwd()` est équivalente de la commande Unix `pwd` vue hier.

```
# où suis-je ? équivalent de la commande Unix 'pwd'
getwd()
```

```
## [1] "/Users/jvanheld/Documents/enseignement/AVIESAN_ecole_bioinformatique/stats_avec_RStudio_EBA/prac
```

R	Unix
`getwd()`	`pwd`

Où vais-je ?

La commande **R** `setwd()` est équivalente de la commande Unix `cd` vue hier.

```
# changement de répertoire courant: équivalent de la commande unix 'cd <path>'

# Chemin relativ
setwd("chemin/acces/au/nouveau/repertoire/")

# Chemin absolu
setwd("/home/hugo/chemin/acces/au/nouveau/repertoire/")
```

R	Unix
`setwd()`	`cd`

Lire et écrire des données

Quuelques manipulations de fichiers

R	Unix
`list.files()`	`ls`
`file.copy()`	`cp`

```
# quelques manipulations de fichiers
f <- list.files(path="/projet/sbr/ggb/intro_R/", full.names=TRUE)
print(f)
```

```
## character(0)
```

```
file.copy(from=f, to=getwd())
```

```
## logical(0)
```

Vérifier ensuite que les fichiers ed données ont bien été copiés dans l'espace de travail.

```
list.files(path=getwd())
```

```
## [1] "intro_R_Roscoff.html"
## [2] "intro_R_Roscoff.pdf"
## [3] "intro_R_Roscoff.r"
## [4] "intro_R_Roscoff.Rmd"
## [5] "rnaseq_data.csv"
## [6] "rnaseq_data.txt"
## [7] "rnaseq_export.txt"
## [8] "Saccharomyces_cerevisiae.R64-1-1.90.gtf"
## [9] "slides_intro_R_roscoff_2017.pdf"
```

```
list.files() ## Commande équivalente, plus légère
```

```
## [1] "intro_R_Roscoff.html"
## [2] "intro_R_Roscoff.pdf"
## [3] "intro_R_Roscoff.r"
## [4] "intro_R_Roscoff.Rmd"
## [5] "rnaseq_data.csv"
## [6] "rnaseq_data.txt"
## [7] "rnaseq_export.txt"
## [8] "Saccharomyces_cerevisiae.R64-1-1.90.gtf"
## [9] "slides_intro_R_roscoff_2017.pdf"
```

Charger les données à partir d'un fichier texte

La commande `read.table()` permet de charger le contenu d'un fichier dans une variable.

Le fichier `rna_seq.txt` contient des valeurs séparées par des tabulations (symbole `\t`).

```
# Charger le fichier rna_seq
rna <- read.table("rnaseq_data.txt", sep="\t", header=TRUE)
print(rna)        # Vérifier le contenu
```

```
##             geneid     name   WT1   WT2   WT3   KO1   KO2   KO3
## 1  ENSG00000000003   TSPAN6    64    55    37    62    45    50
## 2  ENSG00000000005     TNMD     0     0     0     0     0     0
## 3  ENSG00000000419     DPM1  8370  5420  6154  7823  5283  5849
## 4  ENSG00000000457    SCYL3   970   811   567   950   669   545
## 5  ENSG00000000460 C1orf112  1689  1113   981  2065  1082  1264
## 6  ENSG00000000938      FGR    44    25    10    24     5     1
## 7  ENSG00000000971      CFH   205    27    32    20     5     7
## 8  ENSG00000001036    FUCA2  1735   688   661  1543   491   414
## 9  ENSG00000001084     GCLC  3526  5007  3802  4207  4583  3514
## 10 ENSG00000001167     NFYA  4730  5398  3725  4556  5408  4173
## 11 ENSG00000001460    STPG1   360   368   265   422   348   302
## 12 ENSG00000001461   NIPAL3  2260  2509  1851  2527  2283  1884
## 13 ENSG00000001497    LAS1L  6007  9238  5271  5469 10413  7290
## 14 ENSG00000001561    ENPP4  1507   795  1240  1427   734  1045
## 15 ENSG00000001617   SEMA3F     1     3     0     5     8     3
## 16 ENSG00000001626     CFTR     0     0     0     0     0     0
## 17 ENSG00000001629   ANKIB1  1855  3210  1515  2182  3073  1567
## 18 ENSG00000001630  CYP51A1  7933  2837  3119  8635  3484  3192
## 19 ENSG00000001631    KRIT1  1200  2225  1115  1491  1872  1183
## 20 ENSG00000002016    RAD52   159   207    76   196   211    93
```

```
## 21 ENSG00000002079    MYH16     0     0     0     0     1     0
## 22 ENSG00000002330      BAD  1905  1854  1785  1322  1900  1979
## 23 ENSG00000002549     LAP3 22969 14296 15085 19888 14524 16968
## 24 ENSG00000002586     CD99 16856 27742 22743 14489 21501 18203
## 25 ENSG00000002587   HS3ST1    16    13     6    19     4     1
## 26 ENSG00000002726     AOC1     8     0     0     3     0     0
## 27 ENSG00000002745    WNT16    58    25     0    28     9     2
## 28 ENSG00000002746    HECW1     0     0     0     0     0     2
## 29 ENSG00000002822   MAD1L1   753  1989   776   738  2396  1248
## 30 ENSG00000002834    LASP1 10527 21095 11236 11746 22994 13958
## 31 ENSG00000002919    SNX11  4362  4239  3703  3724  3887  3795
## 32 ENSG00000002933 TMEM176A     1     0     5     6     2     3
## 33 ENSG00000003056     M6PR 11037  7346  7685 10150  7153  8069
## 34 ENSG00000003096   KLHL13    66    53    26    97   134    87
## 35 ENSG00000003137  CYP26B1     4     0     4     1     0     2
## 36 ENSG00000003147     ICA1   156    24    26   145    33    56
## 37 ENSG00000003249   DBNDD1    27    32    17     8    49    11
## 38 ENSG00000003393     ALS2  1617  1664  1210  1427  1567  1328
## 39 ENSG00000003400   CASP10   904   764   409  1215  1004   532
## 40 ENSG00000003402    CFLAR 39344 56308 41971 30152 37386 27284
## 41 ENSG00000003436     TFPI     5     1     8    45    16    12
## 42 ENSG00000003509  NDUFAF7   877  1106   716   821   977   570
## 43 ENSG00000003756     RBM5  2502  7437  2398  2770  6499  2745
## 44 ENSG00000003987    MTMR7     0     0     2     0     1     2
## 45 ENSG00000003989   SLC7A2     5     4     0     0     2     0
## 46 ENSG00000004059     ARF5 12018  8963  9746  9462  9168  9353
## 47 ENSG00000004139    SARM1  1154  2913  1247  1041  2420  1262
## 48 ENSG00000004142  POLDIP2 15498 15945 12886 12785 18109 16509
## 49 ENSG00000004399   PLXND1    32   137    55    28   128   102
## 50 ENSG00000004455      AK2 19857 17803 16815 17128 18649 17445
```

```r
print(head(rna))  # N'imprimer que le début du fichier
```

```
##            geneid     name  WT1  WT2  WT3  KO1  KO2  KO3
## 1 ENSG00000000003   TSPAN6   64   55   37   62   45   50
## 2 ENSG00000000005     TNMD    0    0    0    0    0    0
## 3 ENSG00000000419     DPM1 8370 5420 6154 7823 5283 5849
## 4 ENSG00000000457    SCYL3  970  811  567  950  669  545
## 5 ENSG00000000460 C1orf112 1689 1113  981 2065 1082 1264
## 6 ENSG00000000938      FGR   44   25   10   24    5    1
```

Charger les données à partir d'un fichier type CSV

Le fichier `rna_seq.csv` contient des valeurs séparées par des points-virgules (en dépit de son nom "comma-separated values").

- L'option `sep` permet de spécifier un séparateur de colonnes.
- L'option `col.names` permet d'assigner des noms aux colonnes pendant la lecture des données.

```r
# charger le fichier rnaseq_data.csv
rna2 <- read.table(
  "rnaseq_data.csv", sep=";", header=FALSE,
  col.names=c("geneid","name","WT1","WT2","WT3","KO1","KO2","KO3"))
print(head(rna2))
```

```
##            geneid     name  WT1  WT2  WT3  KO1  KO2  KO3
## 1 ENSG00000000003   TSPAN6   64   55   37   62   45   50
## 2 ENSG00000000005     TNMD    0    0    0    0    0    0
## 3 ENSG00000000419     DPM1 8370 5420 6154 7823 5283 5849
## 4 ENSG00000000457    SCYL3  970  811  567  950  669  545
## 5 ENSG00000000460 C1orf112 1689 1113  981 2065 1082 1264
## 6 ENSG00000000938      FGR   44   25   10   24    5    1
```

Ecriture d'une table de données dans un fichier texte

```
# exporter des données dans un fichier (qui sera écrasé s'il existe déjà !)
write.table(rna, file="rnaseq_export.txt", sep=" ", col.names=FALSE, row.names=FALSE)
```

Environnement R

Liste des objets présents dans l'environnement R

```
# Liste des objets présents dans l'environnement R
ls()
```

```
##  [1] "a"
##  [2] "b"
##  [3] "d"
##  [4] "d2"
##  [5] "f"
##  [6] "l"
##  [7] "l1"
##  [8] "l2"
##  [9] "m"
## [10] "ma_nouvelle_variable_avec_un_nom_a_rallonge"
## [11] "mat1"
## [12] "mat2"
## [13] "mentions"
## [14] "n"
## [15] "resultat"
## [16] "rna"
## [17] "rna2"
## [18] "u"
## [19] "v"
## [20] "w"
## [21] "x"
## [22] "y"
```

Suppression d'objets de la mémoire

```
# Suppression d'un objet (utile si particulièrement lourd)
a <- 3
print(a)
```

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

```
rm(a)
print(a)
```

Error in print(a) : object 'a' not found

```
a <- 3
print(a)
```

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

Sauvegarde de tout l'environnement

La commande `save.image()` enregistre sur disque une copie de l'environnement complet, c'est-à-dire tous les objets existant dans l'espace mémoire de **R**.

```
# sauvegarde de tout l'environnement, i.e. tous les objets existants
save.image(file="mon_environnement.RData")
```

Suppression de tous les éléments présents dans la session R

Attention, en faisant ceci on perd tout le résultat du travail précédent (chargement des tables de comptage RNA-seq, ...)

```
# suppression de tous les éléments présents dans la session R
rm(list=ls())
ls()                              # Vérification: tout a disparu :-(
```

```
## character(0)
```

```
load("mon_environnement.RData")   # On recharge l'environnement
ls()                              # Vérification: tout est revenu :-)
```

```
##  [1] "a"
##  [2] "b"
##  [3] "d"
##  [4] "d2"
##  [5] "f"
##  [6] "l"
##  [7] "l1"
##  [8] "l2"
##  [9] "m"
## [10] "ma_nouvelle_variable_avec_un_nom_a_rallonge"
## [11] "mat1"
## [12] "mat2"
## [13] "mentions"
## [14] "n"
## [15] "resultat"
## [16] "rna"
## [17] "rna2"
## [18] "u"
## [19] "v"
## [20] "w"
## [21] "x"
## [22] "y"
```

Sauvegarde d'un objet spécifique

```
# Vérifions le contenu de la variable a
print(a)
```

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

```
# sauvegarde d'un objet spécifique
save(a, file="objet_a.RData")
```

Chargement d'un objet/environnement

```
# Effacçons le contenu de la variable a
rm(a)
print(a)
```

Error in print(a) : object 'a' not found

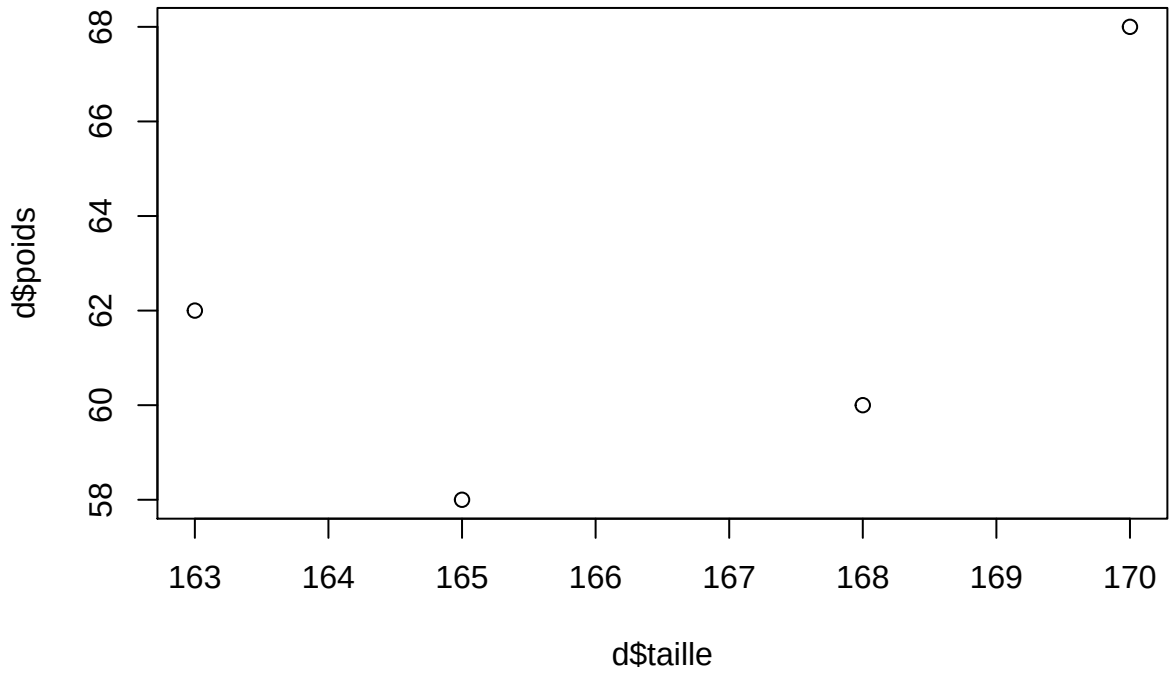
```
# chargement d'un objet/environnement
load("objet_a.RData")
print(a)
```

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

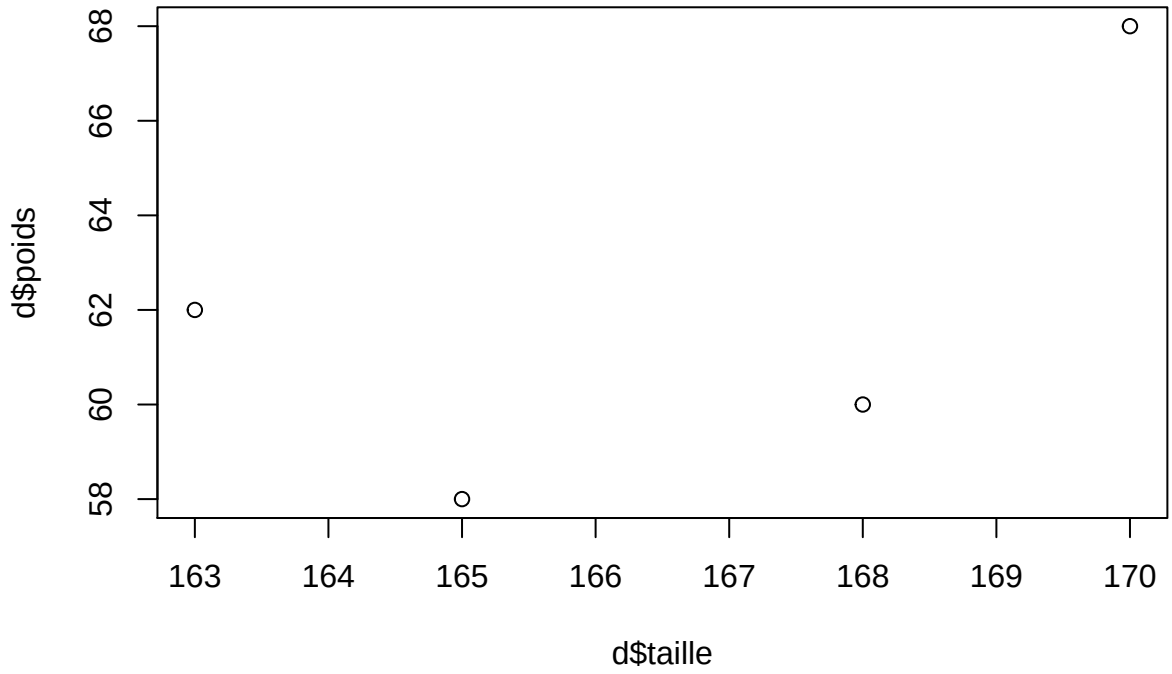
Graphiques

Nuage de points (XY plot)

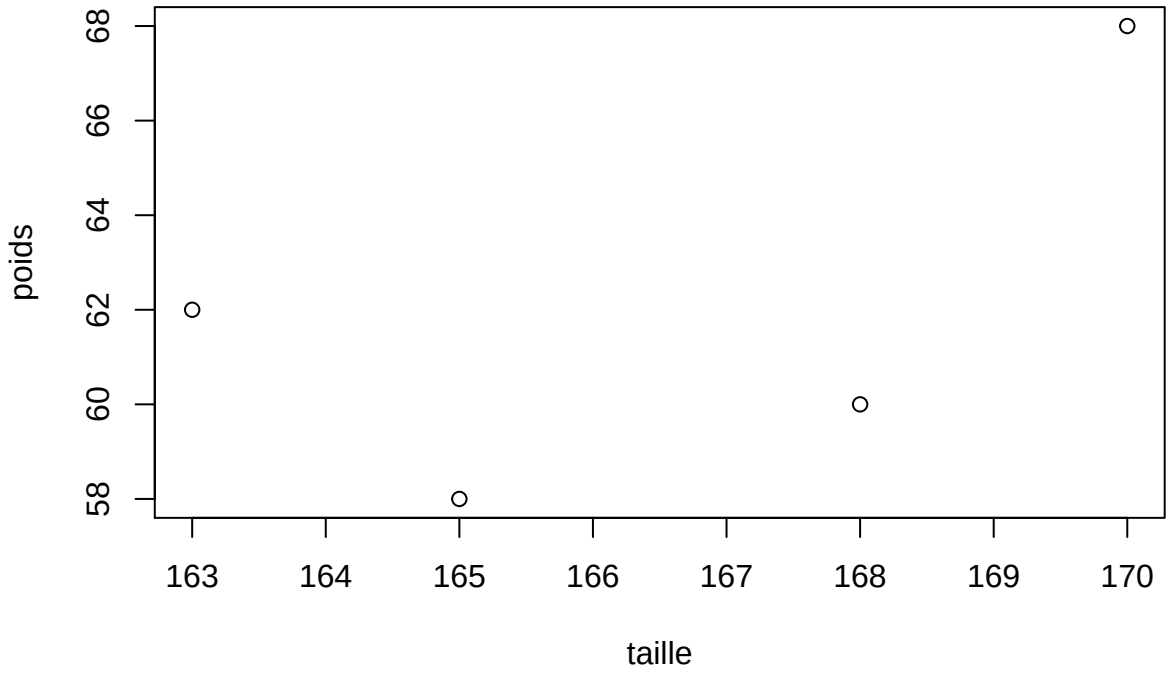
```
# plot y vs x
plot(x=d$taille, y=d$poids)
```



```
plot(formula=d$poids ~ d$taille)
```

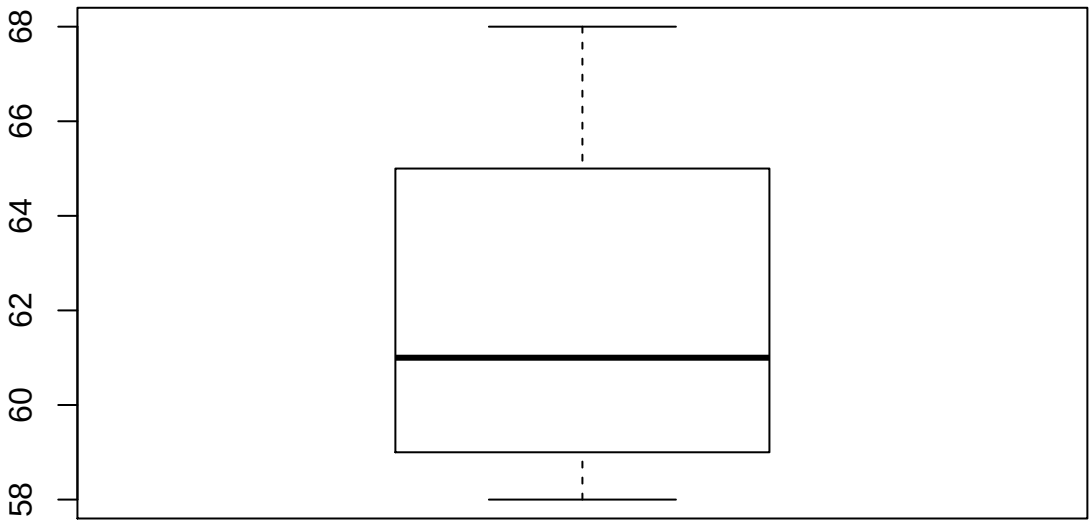



```
plot(formula=poids ~ taille, data=d)
```



Boîte à moustaches (boxplot)

```
# boite à moustache d'une série de valeurs
boxplot(d$poids)
```



Histogramme

```
# histogramme d'une série de valeurs
hist(d$poids)
```

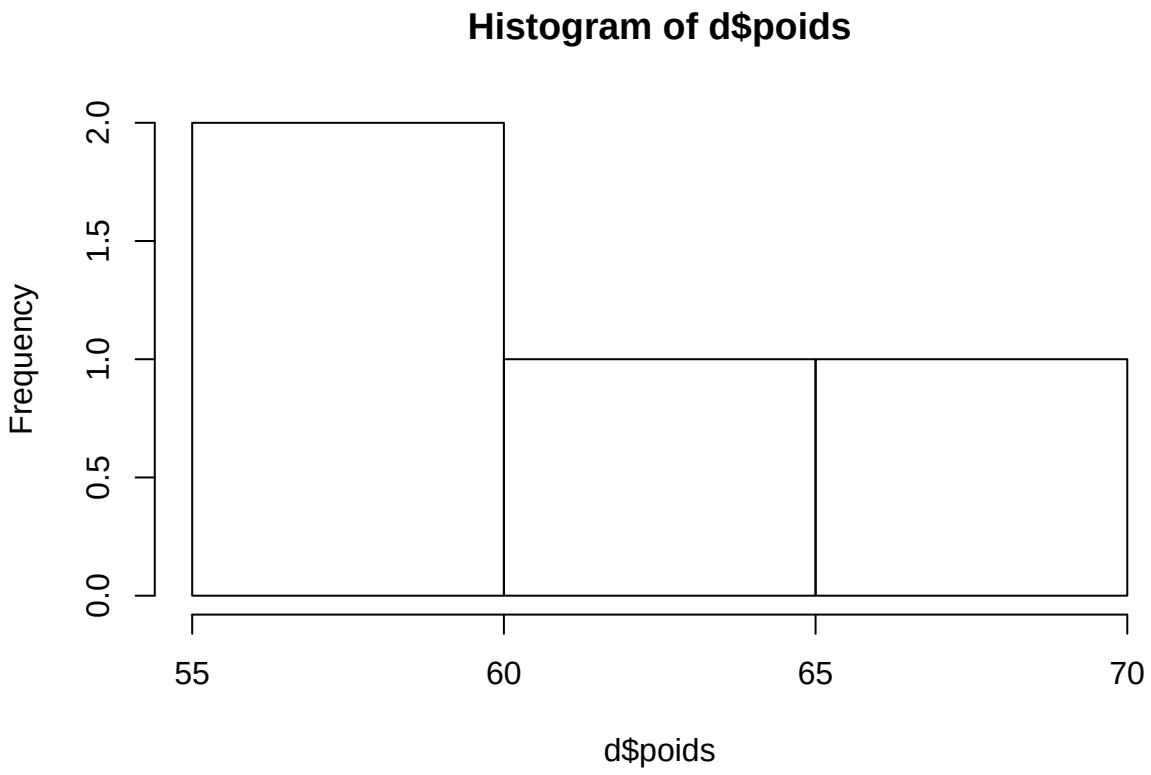
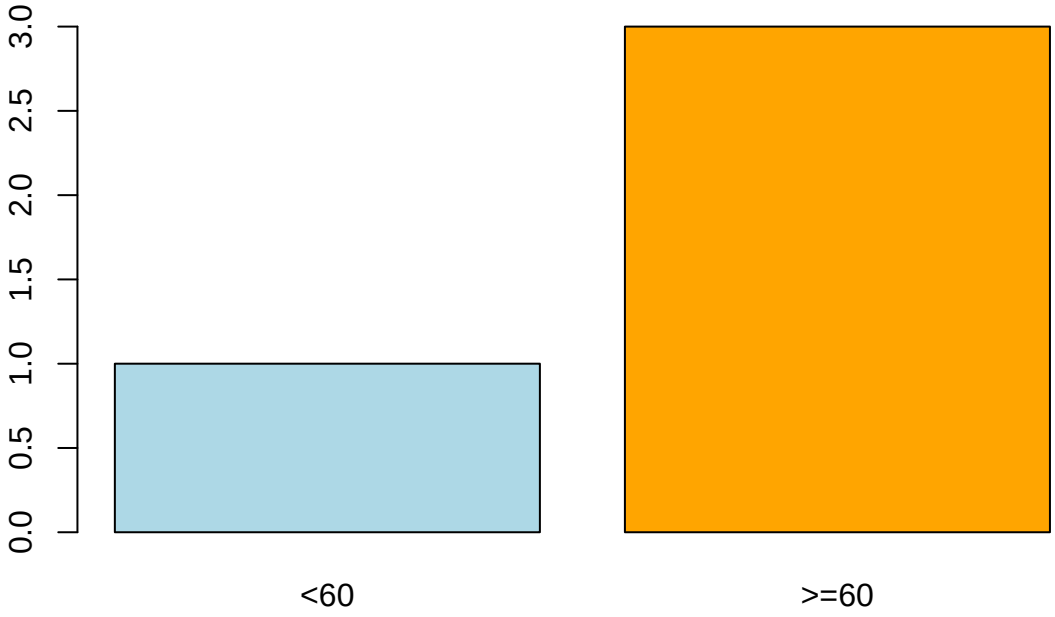


Diagramme en bâtons (barplot)

```
# diagramme en bâtons
table(d$classe_poids)
```

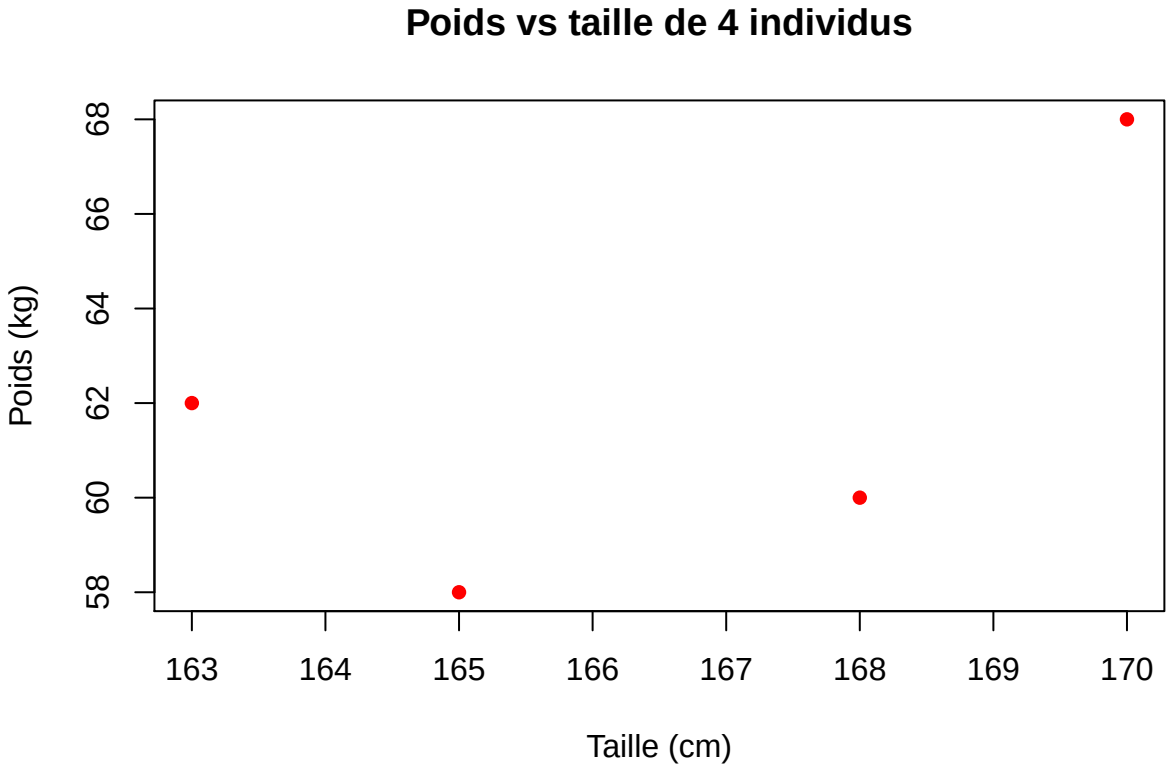
```
##
##  <60 >=60
##    1    3
```

```r
barplot(table(d$classe_poids), col=c("lightblue","orange"))
```



Personnalisation des graphiques

```r
# customisation rapide
plot(formula=poids ~ taille, data=d, pch=16, col="red",
     main="Poids vs taille de 4 individus", xlab="Taille (cm)", ylab="Poids (kg)")
```



Couleurs disponibles

```
# liste des couleurs disponibles
colors()
```

```
##   [1] "white"              "aliceblue"          "antiquewhite"
##   [4] "antiquewhite1"      "antiquewhite2"      "antiquewhite3"
##   [7] "antiquewhite4"      "aquamarine"         "aquamarine1"
##  [10] "aquamarine2"        "aquamarine3"        "aquamarine4"
##  [13] "azure"              "azure1"             "azure2"
##  [16] "azure3"             "azure4"             "beige"
##  [19] "bisque"             "bisque1"            "bisque2"
##  [22] "bisque3"            "bisque4"            "black"
##  [25] "blanchedalmond"     "blue"               "blue1"
##  [28] "blue2"              "blue3"              "blue4"
##  [31] "blueviolet"         "brown"              "brown1"
##  [34] "brown2"             "brown3"             "brown4"
##  [37] "burlywood"          "burlywood1"         "burlywood2"
##  [40] "burlywood3"         "burlywood4"         "cadetblue"
##  [43] "cadetblue1"         "cadetblue2"         "cadetblue3"
##  [46] "cadetblue4"         "chartreuse"         "chartreuse1"
##  [49] "chartreuse2"        "chartreuse3"        "chartreuse4"
##  [52] "chocolate"          "chocolate1"         "chocolate2"
##  [55] "chocolate3"         "chocolate4"         "coral"
##  [58] "coral1"             "coral2"             "coral3"
##  [61] "coral4"             "cornflowerblue"     "cornsilk"
##  [64] "cornsilk1"          "cornsilk2"          "cornsilk3"
##  [67] "cornsilk4"          "cyan"               "cyan1"
```

```
##  [70] "cyan2"              "cyan3"              "cyan4"
##  [73] "darkblue"           "darkcyan"           "darkgoldenrod"
##  [76] "darkgoldenrod1"     "darkgoldenrod2"     "darkgoldenrod3"
##  [79] "darkgoldenrod4"     "darkgray"           "darkgreen"
##  [82] "darkgrey"           "darkkhaki"          "darkmagenta"
##  [85] "darkolivegreen"     "darkolivegreen1"    "darkolivegreen2"
##  [88] "darkolivegreen3"    "darkolivegreen4"    "darkorange"
##  [91] "darkorange1"        "darkorange2"        "darkorange3"
##  [94] "darkorange4"        "darkorchid"         "darkorchid1"
##  [97] "darkorchid2"        "darkorchid3"        "darkorchid4"
## [100] "darkred"            "darksalmon"         "darkseagreen"
## [103] "darkseagreen1"      "darkseagreen2"      "darkseagreen3"
## [106] "darkseagreen4"      "darkslateblue"      "darkslategray"
## [109] "darkslategray1"     "darkslategray2"     "darkslategray3"
## [112] "darkslategray4"     "darkslategrey"      "darkturquoise"
## [115] "darkviolet"         "deeppink"           "deeppink1"
## [118] "deeppink2"          "deeppink3"          "deeppink4"
## [121] "deepskyblue"        "deepskyblue1"       "deepskyblue2"
## [124] "deepskyblue3"       "deepskyblue4"       "dimgray"
## [127] "dimgrey"            "dodgerblue"         "dodgerblue1"
## [130] "dodgerblue2"        "dodgerblue3"        "dodgerblue4"
## [133] "firebrick"          "firebrick1"         "firebrick2"
## [136] "firebrick3"         "firebrick4"         "floralwhite"
## [139] "forestgreen"        "gainsboro"          "ghostwhite"
## [142] "gold"               "gold1"              "gold2"
## [145] "gold3"              "gold4"              "goldenrod"
## [148] "goldenrod1"         "goldenrod2"         "goldenrod3"
## [151] "goldenrod4"         "gray"               "gray0"
## [154] "gray1"              "gray2"              "gray3"
## [157] "gray4"              "gray5"              "gray6"
## [160] "gray7"              "gray8"              "gray9"
## [163] "gray10"             "gray11"             "gray12"
## [166] "gray13"             "gray14"             "gray15"
## [169] "gray16"             "gray17"             "gray18"
## [172] "gray19"             "gray20"             "gray21"
## [175] "gray22"             "gray23"             "gray24"
## [178] "gray25"             "gray26"             "gray27"
## [181] "gray28"             "gray29"             "gray30"
## [184] "gray31"             "gray32"             "gray33"
## [187] "gray34"             "gray35"             "gray36"
## [190] "gray37"             "gray38"             "gray39"
## [193] "gray40"             "gray41"             "gray42"
## [196] "gray43"             "gray44"             "gray45"
## [199] "gray46"             "gray47"             "gray48"
## [202] "gray49"             "gray50"             "gray51"
## [205] "gray52"             "gray53"             "gray54"
## [208] "gray55"             "gray56"             "gray57"
## [211] "gray58"             "gray59"             "gray60"
## [214] "gray61"             "gray62"             "gray63"
## [217] "gray64"             "gray65"             "gray66"
## [220] "gray67"             "gray68"             "gray69"
## [223] "gray70"             "gray71"             "gray72"
## [226] "gray73"             "gray74"             "gray75"
## [229] "gray76"             "gray77"             "gray78"
```

```
## [232] "gray79"             "gray80"             "gray81"
## [235] "gray82"             "gray83"             "gray84"
## [238] "gray85"             "gray86"             "gray87"
## [241] "gray88"             "gray89"             "gray90"
## [244] "gray91"             "gray92"             "gray93"
## [247] "gray94"             "gray95"             "gray96"
## [250] "gray97"             "gray98"             "gray99"
## [253] "gray100"            "green"              "green1"
## [256] "green2"             "green3"             "green4"
## [259] "greenyellow"        "grey"               "grey0"
## [262] "grey1"              "grey2"              "grey3"
## [265] "grey4"              "grey5"              "grey6"
## [268] "grey7"              "grey8"              "grey9"
## [271] "grey10"             "grey11"             "grey12"
## [274] "grey13"             "grey14"             "grey15"
## [277] "grey16"             "grey17"             "grey18"
## [280] "grey19"             "grey20"             "grey21"
## [283] "grey22"             "grey23"             "grey24"
## [286] "grey25"             "grey26"             "grey27"
## [289] "grey28"             "grey29"             "grey30"
## [292] "grey31"             "grey32"             "grey33"
## [295] "grey34"             "grey35"             "grey36"
## [298] "grey37"             "grey38"             "grey39"
## [301] "grey40"             "grey41"             "grey42"
## [304] "grey43"             "grey44"             "grey45"
## [307] "grey46"             "grey47"             "grey48"
## [310] "grey49"             "grey50"             "grey51"
## [313] "grey52"             "grey53"             "grey54"
## [316] "grey55"             "grey56"             "grey57"
## [319] "grey58"             "grey59"             "grey60"
## [322] "grey61"             "grey62"             "grey63"
## [325] "grey64"             "grey65"             "grey66"
## [328] "grey67"             "grey68"             "grey69"
## [331] "grey70"             "grey71"             "grey72"
## [334] "grey73"             "grey74"             "grey75"
## [337] "grey76"             "grey77"             "grey78"
## [340] "grey79"             "grey80"             "grey81"
## [343] "grey82"             "grey83"             "grey84"
## [346] "grey85"             "grey86"             "grey87"
## [349] "grey88"             "grey89"             "grey90"
## [352] "grey91"             "grey92"             "grey93"
## [355] "grey94"             "grey95"             "grey96"
## [358] "grey97"             "grey98"             "grey99"
## [361] "grey100"            "honeydew"           "honeydew1"
## [364] "honeydew2"          "honeydew3"          "honeydew4"
## [367] "hotpink"            "hotpink1"           "hotpink2"
## [370] "hotpink3"           "hotpink4"           "indianred"
## [373] "indianred1"         "indianred2"         "indianred3"
## [376] "indianred4"         "ivory"              "ivory1"
## [379] "ivory2"             "ivory3"             "ivory4"
## [382] "khaki"              "khaki1"             "khaki2"
## [385] "khaki3"             "khaki4"             "lavender"
## [388] "lavenderblush"      "lavenderblush1"     "lavenderblush2"
## [391] "lavenderblush3"     "lavenderblush4"     "lawngreen"
```

```
## [394] "lemonchiffon"          "lemonchiffon1"        "lemonchiffon2"
## [397] "lemonchiffon3"         "lemonchiffon4"        "lightblue"
## [400] "lightblue1"            "lightblue2"           "lightblue3"
## [403] "lightblue4"            "lightcoral"           "lightcyan"
## [406] "lightcyan1"            "lightcyan2"           "lightcyan3"
## [409] "lightcyan4"            "lightgoldenrod"       "lightgoldenrod1"
## [412] "lightgoldenrod2"       "lightgoldenrod3"      "lightgoldenrod4"
## [415] "lightgoldenrodyellow" "lightgray"            "lightgreen"
## [418] "lightgrey"             "lightpink"            "lightpink1"
## [421] "lightpink2"            "lightpink3"           "lightpink4"
## [424] "lightsalmon"           "lightsalmon1"         "lightsalmon2"
## [427] "lightsalmon3"          "lightsalmon4"         "lightseagreen"
## [430] "lightskyblue"          "lightskyblue1"        "lightskyblue2"
## [433] "lightskyblue3"         "lightskyblue4"        "lightslateblue"
## [436] "lightslategray"        "lightslategrey"       "lightsteelblue"
## [439] "lightsteelblue1"       "lightsteelblue2"      "lightsteelblue3"
## [442] "lightsteelblue4"       "lightyellow"          "lightyellow1"
## [445] "lightyellow2"          "lightyellow3"         "lightyellow4"
## [448] "limegreen"             "linen"                "magenta"
## [451] "magenta1"              "magenta2"             "magenta3"
## [454] "magenta4"              "maroon"               "maroon1"
## [457] "maroon2"               "maroon3"              "maroon4"
## [460] "mediumaquamarine"      "mediumblue"           "mediumorchid"
## [463] "mediumorchid1"         "mediumorchid2"        "mediumorchid3"
## [466] "mediumorchid4"         "mediumpurple"         "mediumpurple1"
## [469] "mediumpurple2"         "mediumpurple3"        "mediumpurple4"
## [472] "mediumseagreen"        "mediumslateblue"      "mediumspringgreen"
## [475] "mediumturquoise"       "mediumvioletred"      "midnightblue"
## [478] "mintcream"             "mistyrose"            "mistyrose1"
## [481] "mistyrose2"            "mistyrose3"           "mistyrose4"
## [484] "moccasin"              "navajowhite"          "navajowhite1"
## [487] "navajowhite2"          "navajowhite3"         "navajowhite4"
## [490] "navy"                  "navyblue"             "oldlace"
## [493] "olivedrab"             "olivedrab1"           "olivedrab2"
## [496] "olivedrab3"            "olivedrab4"           "orange"
## [499] "orange1"               "orange2"              "orange3"
## [502] "orange4"               "orangered"            "orangered1"
## [505] "orangered2"            "orangered3"           "orangered4"
## [508] "orchid"                "orchid1"              "orchid2"
## [511] "orchid3"               "orchid4"              "palegoldenrod"
## [514] "palegreen"             "palegreen1"           "palegreen2"
## [517] "palegreen3"            "palegreen4"           "paleturquoise"
## [520] "paleturquoise1"        "paleturquoise2"       "paleturquoise3"
## [523] "paleturquoise4"        "palevioletred"        "palevioletred1"
## [526] "palevioletred2"        "palevioletred3"       "palevioletred4"
## [529] "papayawhip"            "peachpuff"            "peachpuff1"
## [532] "peachpuff2"            "peachpuff3"           "peachpuff4"
## [535] "peru"                  "pink"                 "pink1"
## [538] "pink2"                 "pink3"                "pink4"
## [541] "plum"                  "plum1"                "plum2"
## [544] "plum3"                 "plum4"                "powderblue"
## [547] "purple"                "purple1"              "purple2"
## [550] "purple3"               "purple4"              "red"
## [553] "red1"                  "red2"                 "red3"
```

```
## [556] "red4"                "rosybrown"           "rosybrown1"
## [559] "rosybrown2"          "rosybrown3"          "rosybrown4"
## [562] "royalblue"           "royalblue1"          "royalblue2"
## [565] "royalblue3"          "royalblue4"          "saddlebrown"
## [568] "salmon"              "salmon1"             "salmon2"
## [571] "salmon3"             "salmon4"             "sandybrown"
## [574] "seagreen"            "seagreen1"           "seagreen2"
## [577] "seagreen3"           "seagreen4"           "seashell"
## [580] "seashell1"           "seashell2"           "seashell3"
## [583] "seashell4"           "sienna"              "sienna1"
## [586] "sienna2"             "sienna3"             "sienna4"
## [589] "skyblue"             "skyblue1"            "skyblue2"
## [592] "skyblue3"            "skyblue4"            "slateblue"
## [595] "slateblue1"          "slateblue2"          "slateblue3"
## [598] "slateblue4"          "slategray"           "slategray1"
## [601] "slategray2"          "slategray3"          "slategray4"
## [604] "slategrey"           "snow"                "snow1"
## [607] "snow2"               "snow3"               "snow4"
## [610] "springgreen"         "springgreen1"        "springgreen2"
## [613] "springgreen3"        "springgreen4"        "steelblue"
## [616] "steelblue1"          "steelblue2"          "steelblue3"
## [619] "steelblue4"          "tan"                 "tan1"
## [622] "tan2"                "tan3"                "tan4"
## [625] "thistle"             "thistle1"            "thistle2"
## [628] "thistle3"            "thistle4"            "tomato"
## [631] "tomato1"             "tomato2"             "tomato3"
## [634] "tomato4"             "turquoise"           "turquoise1"
## [637] "turquoise2"          "turquoise3"          "turquoise4"
## [640] "violet"              "violetred"           "violetred1"
## [643] "violetred2"          "violetred3"          "violetred4"
## [646] "wheat"               "wheat1"              "wheat2"
## [649] "wheat3"              "wheat4"              "whitesmoke"
## [652] "yellow"              "yellow1"             "yellow2"
## [655] "yellow3"             "yellow4"             "yellowgreen"
```

Notions basiques de programmation

Fonctions

```
####################################
#            fonctions             #
####################################
ma_fonction <- function(x, y, z=0){
  resultat <- x + y + z
  return(resultat)
}

ma_fonction(x=100, y=10)
```

```
## [1] 110
```

```
ma_fonction(100, 10, 1000)
```

```
## [1] 1110
```

```
ma_fonction(z=10000, x=10, y=100)
```

```
## [1] 10110
```

Condition

```
#####################################
#             condition             #
#####################################
# quelques opérateurs
2 < 3
```

```
## [1] TRUE
```

```
2 >= 2
```

```
## [1] TRUE
```

```
2 != 3
```

```
## [1] TRUE
```

```
4 == 4
```

```
## [1] TRUE
```

```
2 == 3 | 3 == 3
```

```
## [1] TRUE
```

```
2 == 3 & 3 == 3
```

```
## [1] FALSE
```

```
a <- 1
if (a == 1){
  print("a est bien égal à 1")
} else{
  print("a n'est pas égale à 1")
}
```

```
## [1] "a est bien égal à 1"
```

Attention: une seule chose doit être testée

```
# attention: une seule chose doit être testée
v <- c(1, 3, 0)
v == 3
```

```
## [1] FALSE  TRUE FALSE
```

```
if (v == 3){
  print("v est égal à 3")
} else{
```

```
  print("v n'est pas égal à 3")
}
```

```
## Warning in if (v == 3) {: the condition has length > 1 and only the first
## element will be used
```

```
## [1] "v n'est pas égal à 3"
```

Boucles

```
for (i in 1:6){
  phrase <- paste("Voici le nombre", i)
  print(phrase)
}
```

```
## [1] "Voici le nombre 1"
## [1] "Voici le nombre 2"
## [1] "Voici le nombre 3"
## [1] "Voici le nombre 4"
## [1] "Voici le nombre 5"
## [1] "Voici le nombre 6"
```

```
prenoms <- c("pierre", "paul", "jacques", "louis")
for (i in 1:length(prenoms)){
  phrase <- paste("il ou elle s'appelle", prenoms[i])
  print(phrase)
}
```

```
## [1] "il ou elle s'appelle pierre"
## [1] "il ou elle s'appelle paul"
## [1] "il ou elle s'appelle jacques"
## [1] "il ou elle s'appelle louis"
```

```
for (p in prenoms){
  phrase <- paste("il ou elle s'appelle", p)
  print(phrase)
}
```

```
## [1] "il ou elle s'appelle pierre"
## [1] "il ou elle s'appelle paul"
## [1] "il ou elle s'appelle jacques"
## [1] "il ou elle s'appelle louis"
```

While (tant que)

```
# tant que
a <- 1
while (a < 4){
  print(paste("a est égal à", a))
  a <- a + 1
}
```

```
## [1] "a est égal à 1"
## [1] "a est égal à 2"
```

```
## [1] "a est égal à 3"
```

Packages

Chargement d'un package déjà installé

- CRAN: 11203 packages (août 2017) avec croissance exponentielle
- Bioconductor: 1381 "software" packages (août 2017)

```
# chargement d'un package déjà installé
library(survival)
# rend disponible des fonctions spécifiques aux modèles de survie, par exemple:
?coxph
```

Installation d'un package disponible sur le CRAN

```
# Installation d'un package disponible sur le CRAN
install.packages("packHV")
library(packHV)
?desc
```

installation d'un package disponible sur BioConductor

Note: la première installation d'un package BioConductor peut prendre du temps car il faut installer une série de packages de base.

```
# installation d'un package disponible sur Bioconductor
source("https://bioconductor.org/biocLite.R")
biocLite("rfPred")
# répondre "n" si R propose d'updater des packages
library(rfPred)
?rfPred_scores
```

Modélisation statistique

Modèle linéaire simple

```
# modèle linéaire simple
fit <- lm(formula=poids ~ taille, data=d)
fit
```

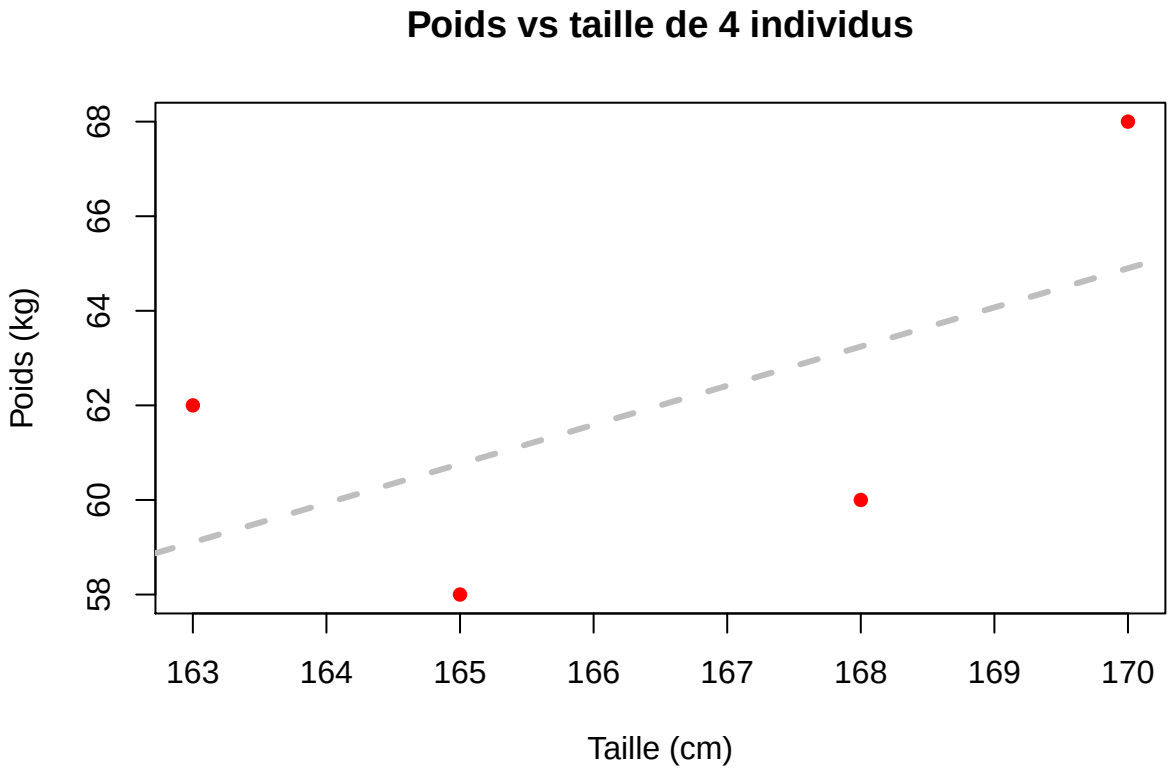
```
##
## Call:
## lm(formula = poids ~ taille, data = d)
##
## Coefficients:
## (Intercept)       taille
##    -75.7931       0.8276
```

```
summary(fit)
```

```
##
## Call:
## lm(formula = poids ~ taille, data = d)
##
## Residuals:
##      1      2      3      4
## -2.759 -3.241  2.897  3.103
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -75.7931   131.4434  -0.577    0.622
## taille        0.8276     0.7893   1.048    0.404
##
## Residual standard error: 4.251 on 2 degrees of freedom
## Multiple R-squared:  0.3547, Adjusted R-squared:  0.03202
## F-statistic: 1.099 on 1 and 2 DF,  p-value: 0.4044
```

Interprétation graphique

```
# interprétation graphique
plot(formula=poids ~ taille, data=d, pch=16, col="red",
     main="Poids vs taille de 4 individus", xlab="Taille (cm)", ylab="Poids (kg)")
abline(a=-75.7931, b=0.8276, lty=2, lwd=3, col="grey")
```



Et plein d'autres choses

- analyses exploratoires: ACP, clustering, etc
- tests statistiques
- génération automatique de rapports d'analyses en PDF/HTML via rmarkdown: mélange de texte, code et résultats
- applications web via R-Shiny (e.g. checkmyindex.pasteur.fr ou shaman.c3bi.pasteur.fr)