Analyses statistiques avec 

Introduction et éléments de base

M. Bailly-Bechet,
adapté de J. R. Lobry, adapté de Deepayan Sarkar

Biostatistiques & Bioinformatique (L2)

Table des matières

R en trois lignes

- R est un environnement d'analyse statistique qui a vu le jour en 1995, suite à un projet néo-zélandais lancé en 1990. Il est basé sur le langage de programmation S.
- R est un logiciel libre : il est accessible à tous gratuitement, et les utilisateurs peuvent librement modifier son code source – et faire partager leurs modifications aux autres – à condition de ne pas les commercialiser par la suite. Il existe également des implémentations commerciales, telles que `Rpro` ou `R+`.
- R est un logiciel professionnel, employé de manière courante par des scientifiques et des entreprises. Il est fréquemment cité comme référence dans des analyses statistiques.

Sommaire

Qu'est ce que R ?

- R est un environnement permettant de faire des analyses statistiques et de produire des graphiques. C'est également un langage de programmation complet.
- Nous allons utiliser ici R comme une boîte à outils pour faire des analyses statistiques standard, telles que celles que vous avez étudiées ce semestre.
- Cependant, il faut bien comprendre que R est un langage de programmation : il est implémenté pour permettre à chacun de personnaliser ses techniques d'analyse.

Les informations sur R sont disponibles sur la page Web du projet R : `http://www.r-project.org`, c'est le premier résultat pour la recherche de la lettre "R" avec le moteur de recherche google.

Lancer et quitter 

- *Unix/Linux* : taper "R" dans un terminal
- *Mac OS X* : double-click sur l'icône 
- *Windows* : double-click sur l'icône 

Pour quitter  , entrer `q()` ou `quit()` sur la ligne de commande.

Quelques exemples simples

Dans les exemples suivants, ce qui est entré par l'utilisateur figure en rouge, et la réponse de  est en bleu. Par exemple :

```
2 + 2
```

```
[1] 4
```

Quelques exemples simples

```
exp(-2)
```

```
[1] 0.1353353
```

```
cos(17 * pi/4)
```

```
[1] 0.7071068
```

```
rnorm(10, mean = 0, sd = 3)
```

```
 [1]  0.2397997 -3.1966877 -0.7084357  3.9569580 -1.2746693 -3.9656568
 [7]  3.5075520  2.4327539  0.7470257  1.8902481
```

La dernière commande produit 10 nombres pseudo-aléatoires d'après une loi normale de moyenne 0 et d'écart-type 3. Le résultat affiché est un vecteur de 10 nombres.
Les nombres entre crochets au début de chaque ligne donnent l'indice du premier nombre de la ligne.

Les fonctions

- `exp()`, `log()` et `rnorm()` sont des fonctions.
- Les appels aux fonctions sont indiqués par la présence de parenthèses.
- Le contenu des parenthèses est appelé "arguments" de la fonction.
- La plupart des choses utiles sous  sont faites par des fonctions.

Sommaire

Variables et Affectations

Comme la plupart des langages de programmation, R a des variables auxquelles on peut affecter une valeur. Pour cela on utilise l'opérateur <- ou ->. L'opérateur classique = marche aussi, et est un équivalent de <-.

```
x <- 2
y = x + 3
string <- "ceci est une chaîne de caractères"
x
```

```
[1] 2
```

```
y
```

```
[1] 5
```

```
string
```

```
[1] "ceci est une chaîne de caractères"
```

```
x + x
```

```
[1] 4
```

```
x^y
```

```
[1] 32
```

Noms des variables

Les noms de variables sont très flexibles. Cependant, il faut savoir que :

- Les noms de variables ne peuvent pas commencer par un chiffre ou un caractère spécial
- Les noms sont sensibles à la casse des caractères (un caractère minuscule comme `x` est différent d'un caractère majuscule comme `X`)
- Il est très fortement recommandé d'utiliser des noms explicites pour vos variables.

Vecteurs

- Les types élémentaires dans R sont tous des vecteurs
- Même un simple nombre est un vecteur de longueur 1

La fonction `c(...)` peut être utilisée pour générer un nouveau vecteur :

```
poids <- c(60, 72, 57, 90, 95, 72)
poids
```

```
[1] 60 72 57 90 95 72
```

"poids" est donc un vecteur de 5 éléments.
Pour accéder à un élément particulier d'un vecteur, il vous suffit d'utiliser l'opérateur d'indexation `[]` :

```
poids[3]
```

```
[1] 57
```

Arithmétique vectorielle

Les opérations arithmétiques usuelles,

- + pour faire des additions
- - pour faire des soustractions
- * pour faire des multiplications
- / pour faire des divisions
- ^ pour élever à la puissance

et les fonctions mathématiques travaillent élément par élément sur les vecteurs et produisent un autre vecteur :

```
taille <- c(1.75, 1.8, 1.65, 1.9, 1.74, 1.91)
taille^2
```

```
[1] 3.0625 3.2400 2.7225 3.6100 3.0276 3.6481
```

```
imc <- poids/taille^2
imc
```

```
[1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
```

```
log(imc)
```

```
[1] 2.975113 3.101093 3.041501 3.216102 3.446107 2.982460
```

Fonctions vectorisées

Beaucoup de fonctions résument un vecteur de données en produisant un nombre à partir d'un vecteur. Par exemple :

```
sum(poids)
```

```
[1] 446
```

```
length(poids)
```

```
[1] 6
```

```
poids_moyen <- sum(poids)/length(poids)
poids_moyen
```

```
[1] 74.33333
```

```
mean(poids)
```

```
[1] 74.33333
```

```
summary(poids)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  57.00   63.00   72.00   74.33   85.50   95.00
```

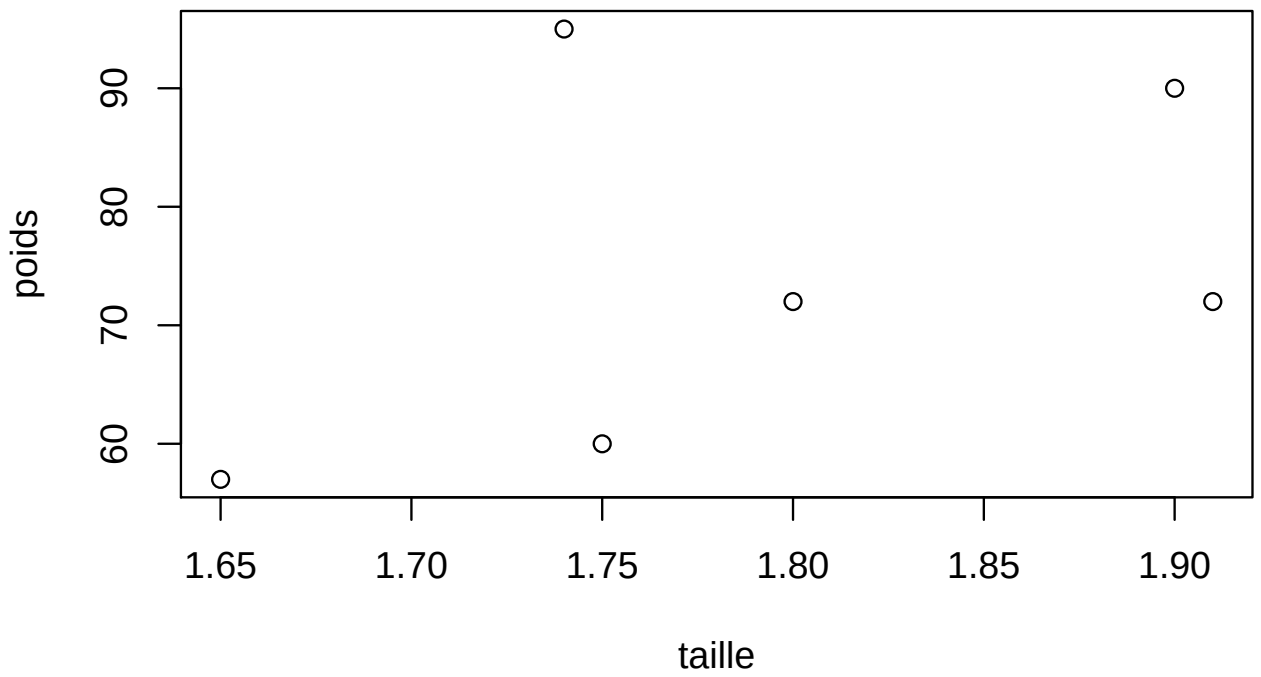
La dernière commande, très utile, donne un résumé statistique des caractéristiques d'un vecteur.

Sommaire

plot()

La manière la plus simple de produire des graphiques sous R est d'utiliser la fonction `plot()`, qui veut dire graphe en anglais :

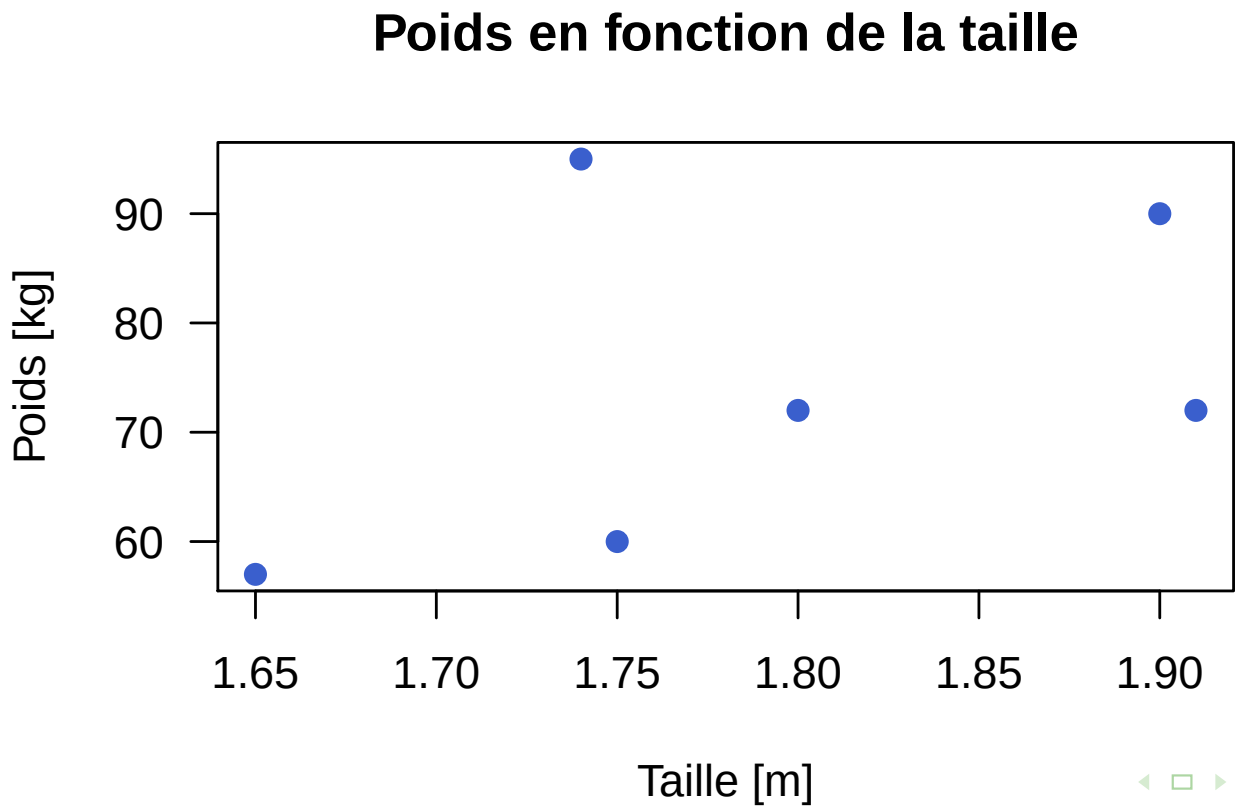
```
plot(x = taille, y = poids)
```



Options et retouche

Un exemple de graphique utilisant quelques options :

```
plot(x = taille, y = poids, pch = 19, col = "royalblue3",
    las = 1, main = "Poids en fonction de la taille", xlab = "Taille [m]",
    ylab = "Poids [kg]")
```



Options et retouche

Dans le graphe précédent :

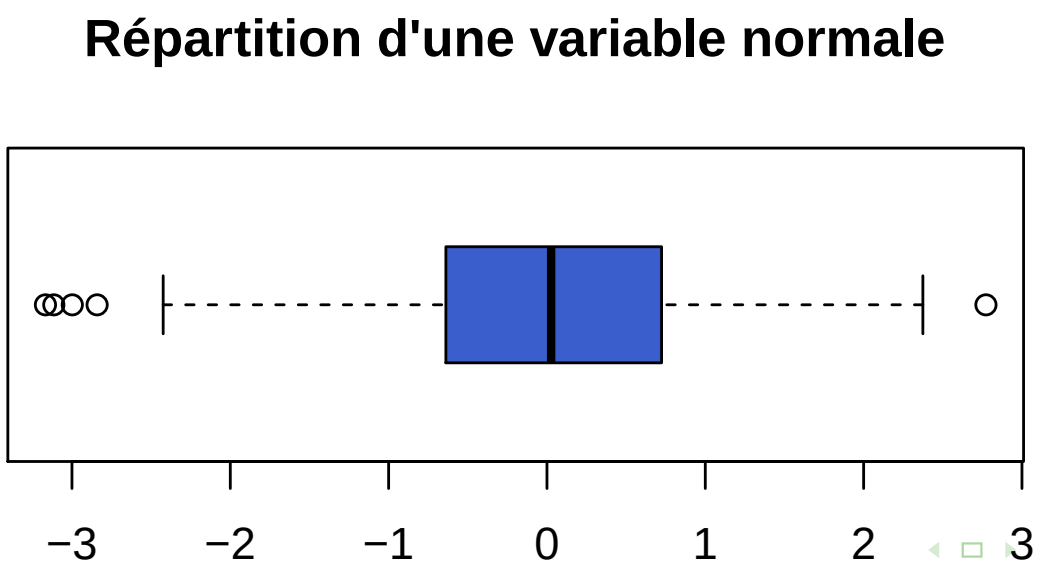
- `pch` permet de choisir le type de point avec lequel on fait le graphe (rond, carré, triangle...)
- `col` permet de choisir la couleur des points
- `las=1` permet d'écrire les poids et les tailles sur les axes à l'horizontale
- `main` définit le titre du graphique
- `xlab` définit le nom de l'axe des abcisses
- `ylab` définit le nom de l'axe des ordonnées

Pour savoir quelles sont les options disponibles pour une fonction, le plus efficace est de regarder la page d'aide de la fonction `plot`, en tapant `help(plot)`.

boxplot()

Ici on génère 500 nombres aléatoires tirés avec une loi normale centrée réduite (commande `rnorm`), et on trace leur répartition avec la commande `boxplot` : 50% des valeurs ont été tirées dans les limites du carré bleu, et 95% entre les bords extrêmes.

```
valeurs<-rnorm(500)
boxplot(valeurs, col = "royalblue3", horizontal = TRUE,
main="Répartition d'une variable normale")
```



Sommaire

help.search()

Quand vous voulez obtenir de l'aide sur un sujet donné, mais que vous ne savez pas quelle est la bonne page d'aide, la fonction `help.search()` est très utile. Elle vous renvoie toutes les fonctions dont la description contient le mot que vous cherchez :

```
help.search("logarithm")
```

```
Help files with alias or concept or title matching fuzzy matching:


VGAM::Log                   Logarithmic Distribution
VGAM::logff                 Logarithmic Distribution
base::log                   Logarithms and Exponentials
nlme::logDet                Extract the Logarithm of the Determinant
```

help(sujet) ou ?sujet

`help(sujet)` que l'on peut aussi écrire `?sujet` affiche la page d'aide pour le sujet ou la fonction `sujet`. Toutes les fonctions de R ont une page d'aide. Quand on connaît le nom de la fonction ou du sujet qui nous intéresse, c'est en général le meilleur moyen d'apprendre à l'utiliser.

```
help(log)
```

```
Description
log computes logarithms, by default natural logarithms. The general form
log(x, base) computes logarithms with base base.

Usage
log(x, base = exp(1))

Arguments
x       a numeric or complex vector.
base    a positive or complex number: the base with respect to which
        logarithms are computed. Defaults to e=exp(1)
```

Que trouve t-on dans les pages d'aide ?

Les pages d'aide sont généralement très détaillées. Elles contiennent souvent, entre autres :

- Une section "See Also" qui donne les pages d'aide sur des sujets apparentés
- Une section "Description" de ce que fait la fonction
- Une section "Examples" avec du code illustrant ce que fait la fonction documentée. Ces exemples sont souvent la meilleure façon de comprendre comment marche la fonction.

Pour en savoir plus

- Pour un public francophone, un très bon point de départ est le manuel d'Emmanuel Paradis, R pour les débutants, qui a la particularité d'exister également en version internationale (*R for Beginners*). Les deux sont disponibles (`http://www.r-project.org/`) dans la rubrique **Documentation**, sous-rubrique **Contributed**.
- Plusieurs milliers de pages d'enseignement en français de statistiques sous  sont disponibles ici : `http://pbil.univ-lyon1.fr/R/`. Les niveaux vont de l'initiation au niveau post-doctoral, à vous d'explorer.

Sommaire

Présentation

Figure: *I.setosa, I.versicolor, I.Virginica*

Les données utilisées ici sont inspirées de données célèbres. Elles ont été collectées par Edgar Anderson. Ce sont les mesures en centimètres des variables suivantes : longueur du sépale, largeur du sépale, longueur du pétale et largeur du pétale pour trois espèces d'iris : *Iris setosa, I. versicolor et I. virginica*.

Chargement du jeu de données

Pour charger un jeu de données à partir d'un fichier texte organisé en colonnes, on utilise la fonction `read.table()` :

```
iris <- read.table("iris_color.dat", header = T)
iris[1:10, ]
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species  Color
1           5.8         4.0          1.2         0.2  setosa purple
2           5.7         4.4          1.5         0.4  setosa    red
3           5.7         3.8          1.7         0.3  setosa    red
4           5.5         4.2          1.4         0.2  setosa purple
5           5.5         3.5          1.3         0.2  setosa purple
6           5.4         3.9          1.7         0.4  setosa   blue
7           5.4         3.7          1.5         0.2  setosa   blue
8           5.4         3.9          1.3         0.4  setosa purple
9           5.4         3.4          1.7         0.2  setosa   blue
10          5.4         3.4          1.5         0.4  setosa orange
```

Extraction de colonnes

On peut ensuite facilement extraire les colonnes qui nous intéressent dans le jeu de données, en utilisant l'opérateur $ pour choisir une colonne dans un jeu de données. Par exemple :

```
 iris[1:30, 3]
```

```
 [1] 1.2 1.5 1.7 1.4 1.3 1.7 1.5 1.3 1.7 1.5 1.5 1.5 1.4 1.5 1.4 1.4 1.5
[18] 1.5 1.7 1.5 1.9 1.6 1.4 1.5 1.6 1.6 1.2 1.3 1.6 1.4
```

```
 length(iris$Petal.Length)
```

```
[1] 150
```

```
 iris$Petal.Length[1:30]
```

```
 [1] 1.2 1.5 1.7 1.4 1.3 1.7 1.5 1.3 1.7 1.5 1.5 1.5 1.4 1.5 1.4 1.4 1.5
[18] 1.5 1.7 1.5 1.9 1.6 1.4 1.5 1.6 1.6 1.2 1.3 1.6 1.4
```

Analyse statistique d'un vecteur

Les analyses statistiques classiques sont pré-implémentées sous  .
Il suffit de parler un peu anglais pour obtenir :

- La moyenne :

```
mean(iris$Petal.Length)
```

```
[1] 3.758
```

- La variance :

```
var(iris$Petal.Length)
```

```
[1] 3.116278
```

- L'écart-type :

```
sd(iris$Petal.Length)
```

```
[1] 1.765298
```

Analyse statistique d'un vecteur (2)

On peut aussi faire des choses plus complexes :

- Une moyenne conditionnelle, celle de la longueur des pétales des iris *setosa* :

```
mean(iris$Petal.Length[iris$Species == "setosa"])
```

```
[1] 1.462
```

```
mean(iris$Petal.Length[1:50])
```

```
[1] 1.462
```

- Le nombre d'iris ayant des pétales de longueur supérieure à 3.5 cm :

```
length(iris$Petal.Length[iris$Petal.Length > 3.5])
```

```
[1] 95
```

Test de Student

On veut comparer la moyenne de la largueur des sépales chez deux espèces, *I. virginica* et *I. versicolor* :

```
 mean(iris$Sepal.Width[iris$Species == "versicolor"])
```

```
[1] 2.77
```

```
 mean(iris$Sepal.Width[iris$Species == "virginica"])
```

```
[1] 2.974
```

```
 t.test(x = iris$Sepal.Width[iris$Species == "versicolor"],
     y = iris$Sepal.Width[iris$Species == "virginica"])
```

```
Welch Two Sample t-test

data:  iris$Sepal.Width[iris$Species == "versicolor"]
       and iris$Sepal.Width[iris$Species == "virginica"]
t = -3.2058, df = 97.927, p-value = 0.001819
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.33028364 -0.07771636
sample estimates:
mean of x mean of y
    2.770     2.974
```

Test du χ^2

On veut vérifier l'indépendance de deux variables qualitatives : l'espèce et la couleur.

```
table(iris$Species)
```

```
    setosa versicolor  virginica
        50         50         50
```

```
table(iris$Color, iris$Species)
```

```
         setosa versicolor virginica
 blue        15         12         7
 orange      12         10        11
 purple      13         17        17
 red         10         11        15
```

```
chisq.test(table(iris$Color, iris$Species))
```

```
Pearson's Chi-squared test

data:  table(iris$Color, iris$Species)
X-squared = 4.9117, df = 6, p-value = 0.5552
```

ANOVA1

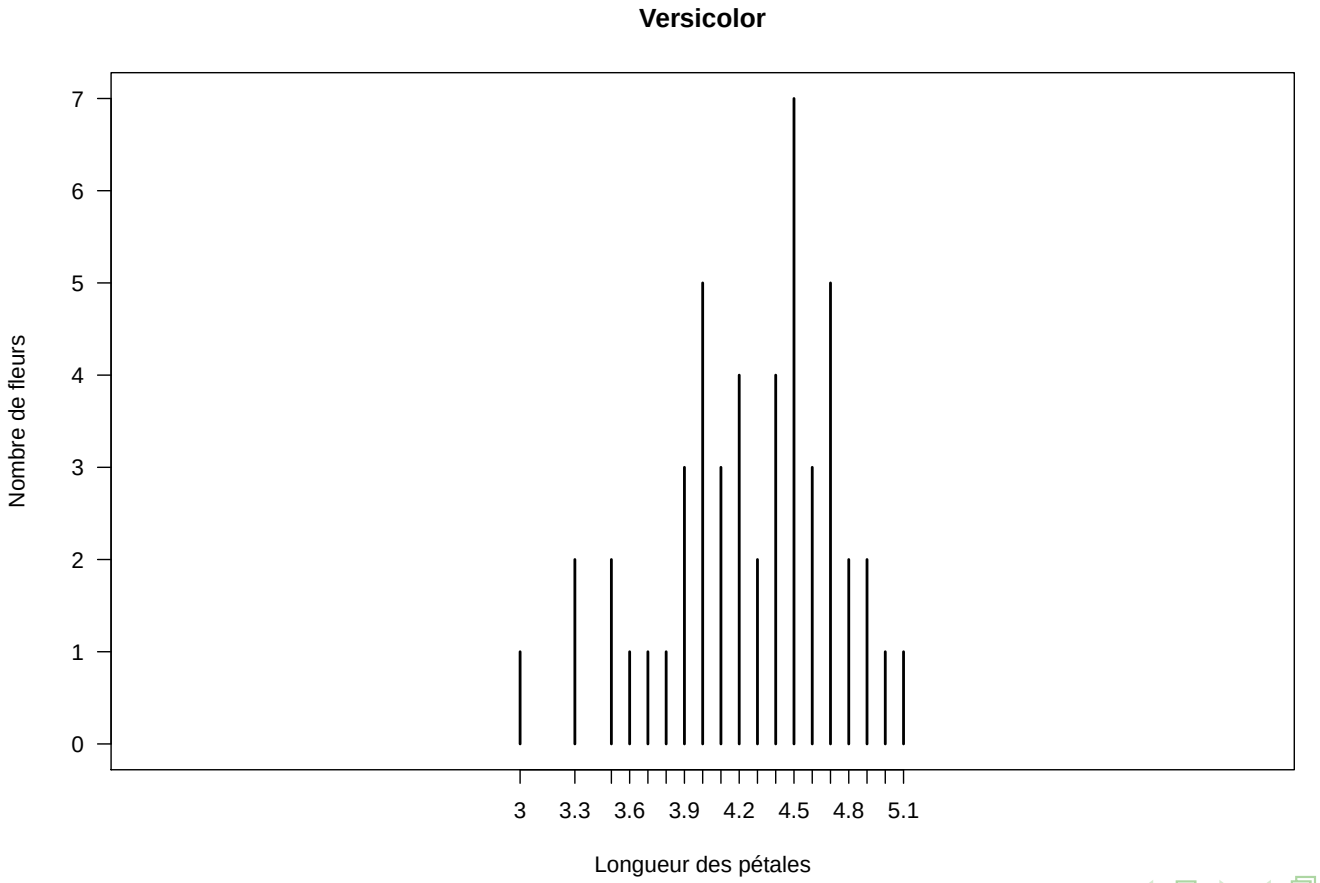
Sous R, les ANOVA deviennent faciles, avec la commande aov.
Ici on fait une ANOVA 1 où l'on étudie la longueur des pétales en fonction de l'espèce (on a donc 3 catégories) :

```
result <- aov(iris$Petal.Length ~ iris$Species)
summary.aov(result)
```

```
              Df Sum Sq Mean Sq F value    Pr(>F)
iris$Species   2 437.10 218.551  1180.2 < 2.2e-16 ***
Residuals    147  27.22   0.185
---
Signif. codes:  0
```

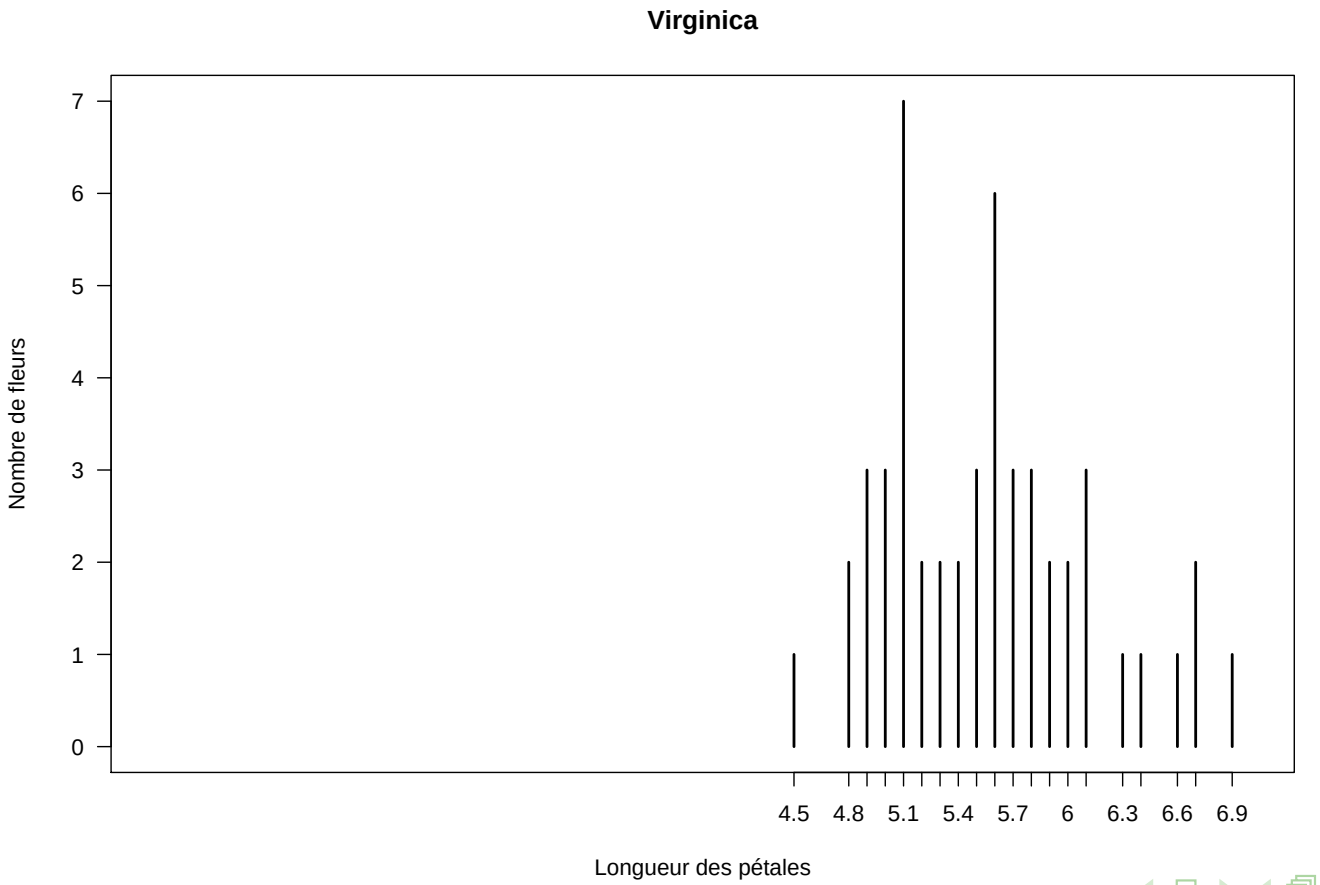
ANOVA1 (suite)

```
plot(table(iris$Petal.Length[iris$Species == "versicolor"]),
    ylab = "Nombre de fleurs", xlab = "Longueur des pétales",
    las = 1, xlim = c(1, 7), main = "Versicolor")
```



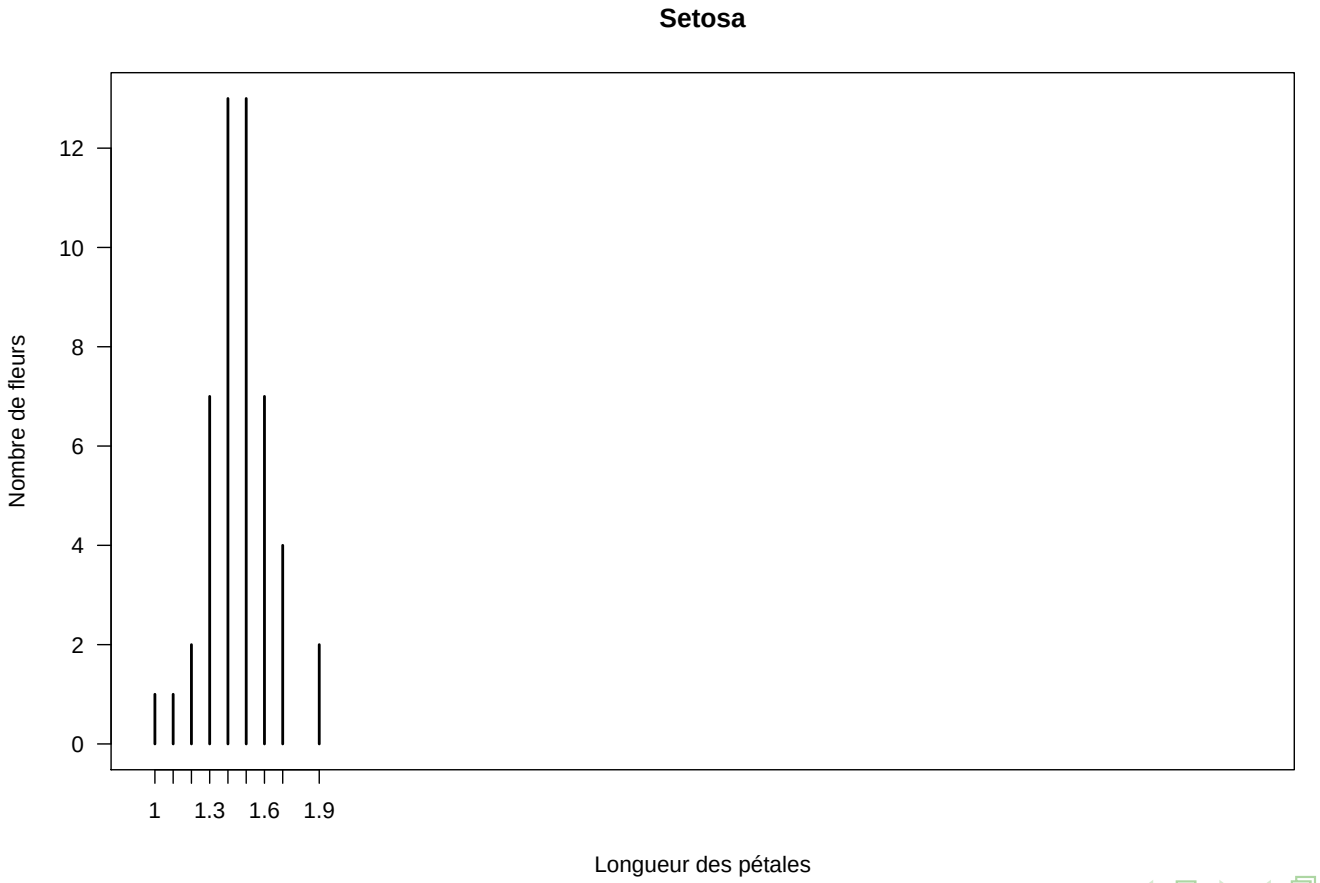
ANOVA1 (suite)

```
plot(table(iris$Petal.Length[iris$Species == "virginica"]),
    ylab = "Nombre de fleurs", xlab = "Longueur des pétales",
    las = 1, xlim = c(1, 7), main = "Virginica")
```



ANOVA1 (suite)

```
plot(table(iris$Petal.Length[iris$Species == "setosa"]),
    ylab = "Nombre de fleurs", xlab = "Longueur des pétales",
    las = 1, xlim = c(1, 7), main = "Setosa")
```



ANOVA2, le retour de la vengeance

Sous R, même les ANOVA 2 sont faciles...
Ici on fait une ANOVA 2 où l'on étudie la longueur des pétales en fonction de l'espèce (3 catégories) et de la couleur (4 catégories) :

```
result2 <- aov(iris$Petal.Length ~ iris$Species * iris$Color)
summary.aov(result2)
```

```
                       Df Sum Sq Mean Sq   F value Pr(>F)
iris$Species            2 437.10 218.551 1196.9440 <2e-16 ***
iris$Color              3   0.28   0.092    0.5040 0.6801
iris$Species:iris$Color 6   1.75   0.291    1.5964 0.1526
Residuals             138  25.20   0.183
---
Signif. codes:  0
```

Régression linéaire et test de linéarité

On peut également estimer et tester une corrélation. La longueur des pétales est-elle corrélée à celle des sépales ?

```
 cor(iris$Petal.Length, iris$Sepal.Length)
```

```
[1] 0.8717538
```

```
 cor.test(iris$Petal.Length, iris$Sepal.Length)
```

```
Pearson's product-moment correlation

data:  iris$Petal.Length and iris$Sepal.Length
t = 21.646, df = 148, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8270363 0.9055080
sample estimates:
      cor
0.8717538
```

Coefficients de la régression

On peut savoir quels sont les coefficients de la régression linéaire :

```
regression <- lm(iris$Sepal.Length ~ iris$Petal.Length)
names(regression)
```

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

```
regression$coefficients
```

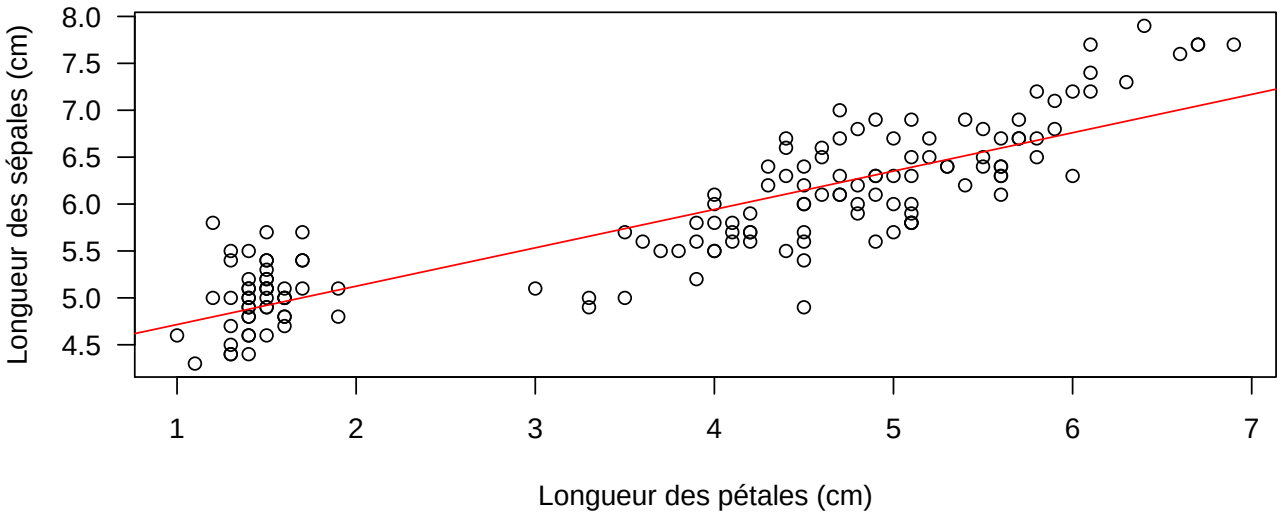
```
      (Intercept) iris$Petal.Length
        4.3066034         0.4089223
```

Un graphe avec régression

Méthode 1

On peut maintenant tracer à la fois le nuage de points et la droite de régression. Deux méthodes donnent le même résultat :

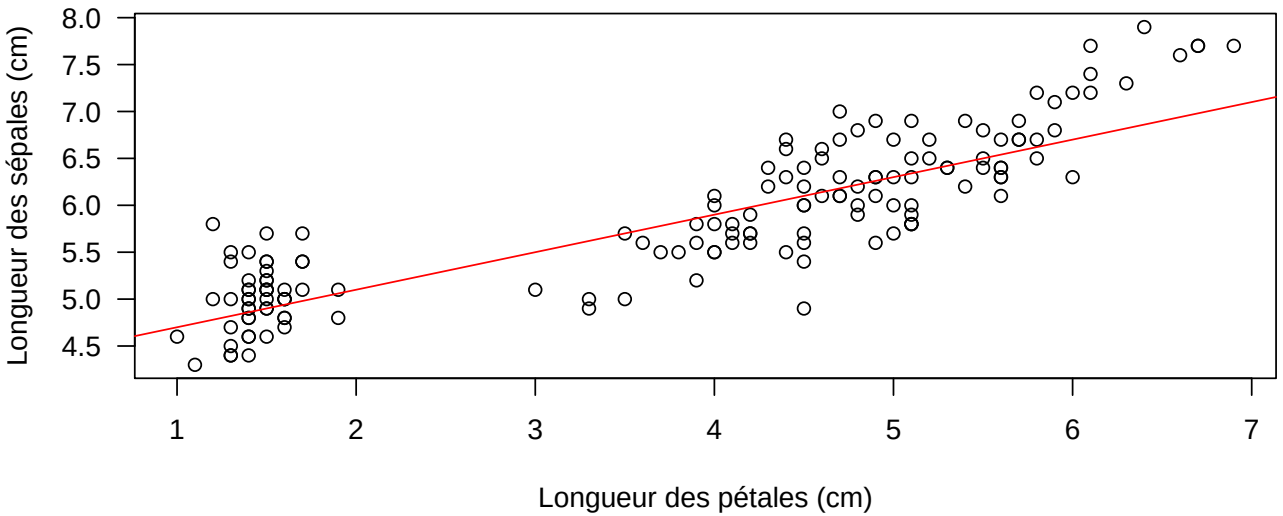
```
plot(iris$Petal.Length,iris$Sepal.Length, las=1,
ylab= "Longueur des sépales (cm)", xlab = "Longueur des pétales (cm)")
reg=lm(iris$Sepal.Length~iris$Petal.Length)
abline(reg,col="red")
```



Un graphe avec régression

Méthode 2

```
plot(iris$Petal.Length,iris$Sepal.Length, las=1,
xlab = "Longueur des pétales (cm)", ylab = "Longueur des sépales (cm)")
abline(a=4.30,b=0.4,col="red")
```



En guise de conclusion. . . une petite mise en garde :

 permet donc de faire de très nombreuses analyses statistiques, ainsi que les graphes correspondants, le tout pour un effort minime d'apprentissage initial.

Cependant, il ne faut pas tomber dans le piège consistant à dire que puisque  sait faire tout cela, il n'est pas besoin de savoir le faire soi-même. . .  ne vous dira pas comment vos données doivent être analysées, il vous aidera seulement à mener à bien l'analyse que vous aurez choisie d'effectuer, parce qu'elle correspond à ce que vous voulez étudier dans votre jeu de données.