Acquisition et analyse de données (BING-F4002): travaux pratiques

Marius Gilbert & Marc Dufrêne

Année académique 2015-2016

I. Introduction

Les travaux pratiques seront effectués à l'aide du logiciel R, qui a été développé pour les statistiques et la modélisation. Bien qu'il existe de nombreux logiciels de statistiques, le choix de R s'est imposé pour ce cours pour plusieurs raisons.

En premier lieu, R est un logiciel extrêmement **versatile**. Il est composé d'un noyau formé par les fonctions de base qui permettent d'effectuer la plupart des statistiques et graphiques abordés au cours. A ce noyau s'ajoutent des centaines de modules (appelés « packages ») développés par des utilisateurs qui permettent de compléter les fonctionnalités de base. La diversité de ces modules est telle qu'il est d'ailleurs parfois difficile de s'y retrouver. Nous reviendrons sur l'utilisation de ces modules.

En second lieu, R est **libre et redistribuable** ce qui signifie que vous êtes libres d'installer le logiciel, de le partager, et même, si le cœur vous en dit, d'accéder au code source et de le modifier sous certaines conditions. En pratique, cela signifie surtout que si vous apprenez R, vous pouvez conserver cet outil informatique avec vous, quel que soit votre parcours professionnel ultérieur.

Enfin, R est **multi-plateforme** et des versions compilées existent pour Windows, Mac et Linux. Ici encore, apprenez R et vous disposez d'un outil qui fonctionnera partout.

En revanche, R est peu convivial et ne se pilote pas à l'aide d'une navigation par des menus et à l'aide de la souris, mais par des lignes de commande. Ceci ralentis la prise en main du logiciel par des utilisateurs débutants, mais i) plusieurs interfaces graphiques ont été développées pour R, ii) une fois l'étape de cette mise en main passée, la productivité que l'on peut atteindre est largement supérieure que par des analyses effectuées à l'aide de menus déroulants. En outre, cela empêche l'utilisation en « boite noire » de l'outil statistique puisque les analyses doivent être explicitées sous la forme de code.

Ce document présente les fonctions de base de R qui seront utilisées aux travaux pratiques. En complément, il existe des dizaines de manuels et ouvrages (libres et commerciaux) en différentes langues qui traitent de la prise en main de R. Nous recommandons en particulier le manuel « R pour les débutants » de E. Paradis qui peut être téléchargé à l'adresse :

http://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf

Une liste de manuels et documents disponibles en ligne est accessible sur :

http://cran.r-project.org/other-docs.html

Il existe un blog extrêmement bien documenté sur les usages de R, qui peut être consulté à l'adresse:

http://www.r-bloggers.com

Enfin, de nombreuses fonctions (avec codes exemples) sont illustrées de manière très conviviale sur la page *Quick-R*:

http://www.statmethods.net

R fournis un noyau de fonctions qui sont utilisables à l'aide d'une console, mais avec peu de support à l'utilisateur. L'application RStudio a été développée pour fournir une interface plus conviviale aux utilisateurs. Si, son utilisation n'est pas indispensable, elle est néanmoins fortement recommandée.

De nombreuses vidéo explicant la manière d'installer et d'utiliser R et Rstudio sont également disponibles ici: http://www.statslectures.com

Les travaux pratiques démarreront par une séance d'introduction à R, à sa syntaxe, et à son organisation. Les séances suivantes seront réalisées en lien direct avec le cours théorique qui précède. **Il est fortement recommandé d'assister au cours qui précède chaque séance de TP**. En effet, le cours théorique et les TP sont prévus pour se compléter l'un l'autre, et les explications données aux TP ne pourront pas être aussi détaillées que celles données au cours.

II. Installation et démarrage de R et RStudio

R est déjà installé sur les différents postes des salles informatiques du NO. Si vous disposez déjà d'une adresse de courrier électronique de l'ULB, vous pouvez directement démarrer une session avec votre login et votre mot de passe (selon les circonstances, données en séance, vous devrez peut-être réactiver votre mot de passe). L'installation de R dans la salle informatique du NO étant modifiée régulièrement, les instructions pour accéder à R seront données en séance .

Pour installer R et RStudio chez vous, ou sur un portable, vous pouvez en premier lieu installer R depuis CRAN (http://cran.r-project.org/), et ensuite RStudio depuis le site http://www.rstudio.com/, en choisissant bien la version gratuite pour Dekstop.

Pour démarrer R et RStudio, il vous suffira de démarrer RStudio (R y est inclus s'il a été préalablement installé). Les instructions sur l'utilisation de RStudio seront données lors de la première séance de travauc pratiques.

III Introduction à R et aux statistiques descriptives

III.1. Fonctions et commandes de base

III.1.1. La console

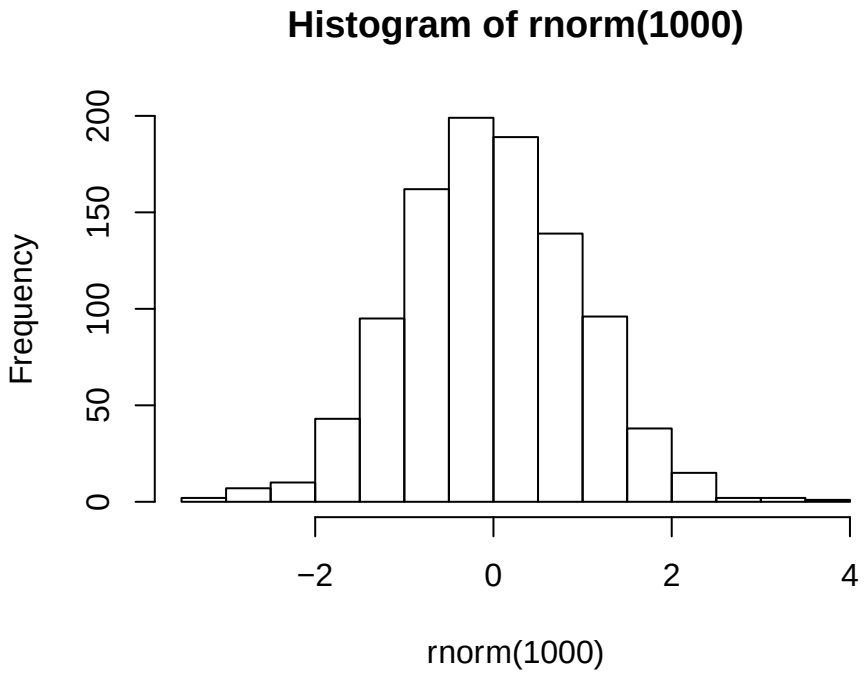
Lorsque vous démarrez RStudio, votre zone de travail est divisée en différentes zones. La plus importante est la console, dans laquelle les instructions seront transmises depuis votre code. C'est dans la console que nous allons entrer des lignes de commande, et qu'en retour, R nous affichera le résultat de ces commandes. De manière à conserver la trace de son code, il est directement conseillé d'écrire son code depuis un nouveau code R. Pour cela, aller dans le menu Fichier | Nouveau fichier | R script. C'est dans cette fenêtre que sera tapé le code, et celui-ci sera transmis à la console, soit en tapant Ctrl-Enter à la fin de chaque ligne, soit en exécutant les lignes de codes à l'aide du bouton approprié.Nous pouvons par exemple réaliser une première opération mathématique :

```
4+5
```

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

Comme on peut le constater, R nous affiche simplement le résultat de l'opération que nous venons de réaliser. Nous pouvons également générer un graphique (ici un histogramme de fréquence de 1000 valeurs aléatoires qui suivent une distribution normale centrée réduite) par une commande comme :

```
hist(rnorm(1000))
```



Cette commande ouvre affiche un graphique dans un des panels de RStudio.

Attention !!! La source d'erreur la plus fréquente dans R est le non-respect de la casse. En effet, R est sensible au respect des majuscules et minuscules. La fonction `hist()` n'est pas équivalente à la fonction `Hist()` (qui n'existe pas et renverra donc un message d'erreur). De même, une variable appelée `monNom` n'est pas la même qu'une autre appelée `monnom`. Lorsqu'il y a une erreur que vous ne comprenez pas, commencez donc toujours par vérifier la casse de vos lignes de commande.

Attention, n'oubliez pas de sauver régulièrement le contenu de votre code. Par ailleurs, vous pouvez ajouter des commentaires à votre code, R ignorera toutes les commandes qui se trouvent après le signe `#` ce qui vous permet d'ajouter des commentaires. Par exemple, le code suivant, renverra exactement le même résultat que le précédent, puisque les 3 premières lignes seront ignorées.

```
######################################################################
# génère un histogramme de fréquence de 1000 valeurs aléatoires
######################################################################
hist(rnorm(1000))
```

III.1.2. Les objets

Nous pourrions réaliser un grand nombre d'opérations sur des nombres, mais R ne serait alors guère plus qu'une calculette. Au lieu d'afficher directement les résultat d'une opération, on peut assigner un contenu à un objet, ainsi la commande :

```
myVar = 4 + 5
```

va simplement assigner le résultat de l'opération 4 + 5 à l'objet `myVar` (attention, il ne faut pas confondre le signe =, qui est un opérateur d'assignation, avec le == qui est l'opérateur de comparaison) Pour afficher le contenu de l'objet myVar, il suffit d'entrer son nom et R va alors afficher son contenu :

```
myVar
```

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

Les objets peuvent être de différents types. Il peut s'agir d'une variable avec une seule valeur comme illustré ci-dessus, d'un tableau de données, d'une fonction, ou du résultat d'une analyse. L'objet le plus simple est une variable qui stocke une valeur numérique, logique ou une chaîne de caractère. Par exemple, en tapant la suite de commandes suivantes, on génère quatre objets différents :

```
A = 67
B = 34.78
C = TRUE
D = "Welcome to R"
```

On peut ensuite afficher le contenu de ces objets, afficher le résultat d'opérations, ou assigner le résultat d'une opération à un nouvel objet :

```
A*B
```

```
## [1] 2330
```

```
D
```

```
## [1] "Welcome to R"
```

On peut déjà constater que le fait que le type d'objet a une influence sur le résultat. R autorise et affiche le produit de A par B, en revanche, lorsque l'on multiplie la chaîne de caractères par un nombre (par exemple en multipliant A par D), R ne sachant comment interpréter cette multiplication, affiche un message d'erreur. Ayant créé un certain nombre d'objets, on peut demander à R d'afficher la liste des objets existants par la commande :

```
ls()
```

```
## [1] "A"     "B"     "C"     "D"     "myVar"
```

Notons au passage que pour la première fois, nous utilisons une fonction, en l'occurrence la fonction ls (). Comme pour toutes les fonctions, nous pouvons demander de l'aide sur cette fonction en tapant `? ls()` qui nous renverra une aide sur cette fonction et les arguments éventuels. Il y a de très nombreux types d'objets dans R. les principaux types d'objets que nous allons utiliser dans le cadre de ces travaux pratiques sont :

- les scalaires : une seule valeur stockée dans un objet ;
- les vecteurs : un ensemble de valeurs de même type (numérique, logiques, caractères) ;
- les matrices : un ensemble de valeurs de même type stockées dans une matrice à n x m lignes et colonnes (numériques, logique, caractéres) ;
- les "dataframe" : tableau de données dans lequel les observations sont disposées en lignes, et les variables sont disposées en colonnes ;
- les listes : ensemble d'objets pouvant être de types différents ;

III.1.3. Vecteurs et matrices

Pour construire un vecteur, on peut simplement combiner des valeurs en utilisant la fonction `c()` :

```
V1 = c(0,34,87,26,97,566,20,15,6,70)
V1
```

```
##  [1]   0  34  87  26  97 566  20  15   6  70
```

Ou bien générer une séquence de nombres avec la fonction `seq()`

```
V2 = seq(from = 1.5, to = 2.4, by = 0.1)
V2
```

```
##  [1] 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4
```

Ou encore utiliser la fonction `rnorm(10)` pour générer 10 valeurs aléatoires qui suivent une distribution normale centrée réduite. R peut générer des séquences aléatoires correspondant à de nombreuses fonctions théoriques (`rnorm()` pour une normale, `runif()` pour une distribution uniforme, `rpois()` pour une distribution de Poisson).

```
V3 = rnorm(10)
V3
```

```
##  [1] -0.4442 -0.1472 -0.2684  0.1572 -0.6094 -1.7642  0.6333 -0.1026
##  [9]  0.1505  0.9307
```

Ici encore, on trouvera de l'aide sur ces fonctions à l'aide de la commande `? seq`, ou `? rnorm`. Les opérations sur les vecteurs sont permises pour autant que les vecteurs aient des longueurs identiques (attention, dans le cas contraire, R n'affiche pas de message d'erreur mais recycle les valeurs du vecteur le plus court).

```
V1 * V2
```

```
##  [1]    0.0   54.4  147.9   46.8  184.3 1132.0   42.0   33.0   13.8  168.0
```

```
V1 + c(3,10)
```

```
##  [1]   3  44  90  36 100 576  23  25   9  80
```

On peut également faire une opération entre un vecteur et un scalaire, l'opération s'applique entre le scalaire et chaque élément du vecteur pour fournir un vecteur de même longueur:

```
3.3 * V1
```

```
##  [1]    0.0  112.2  287.1   85.8  320.1 1867.8   66.0   49.5   19.8  231.0
```

Pour accèder à la nème valeur d'un vecteur, on utilise un système d'indexation qui est symbolisé par les signes `[ ]`. Pour accèder à la 5ème valeur du vecteur V1, on entrera donc :

```
V1[5]
```

```
## [1] 97
```

De la même manière, on peut accèder à plusieurs valeurs, par exemple, `V1[c(2,5)]` nous renverra la 2ème et la 5ème valeur du vecteur V1. Ou encore `V1[2:5]` nous renverras toutes les valeurs depuis la position 2 à la position 5 du vecteur V1.

Des matrices peuvent également être crées à l'aide de la fonction `matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)`. Dans un premier temps, nous ne les utiliserons pas. Sachez cependant que les opérateurs peuvent s'appliquer aux matrices (par défaut, ceux-ci s'appliquent aux élements i,j de la matrice deux à deux), et que des fonctions spécifiques de calcul matriciel existent (produit matriciel, transposition, calcul de déterminant, etc....), mais celles-ci seront introduite au fur et à mesure des besoins.

III.1.4. Opérateurs

Les principaux opérateurs de R sont :

- Arithmétiques
  - addition: `a + b`
  - soustraction: `a - b`
  - division: `a / b`
  - multiplication: `a * b`
  - puissance: `a^b`
  - modulo (reste de la division): `a %% b`
  - division entière: `a %/% b`
- Comparaisons logiques
  - inférieur à: `a < b`
  - supérieur à: `a > b`
  - inférieur ou égal à: `a <= b`
  - supérieur ou égal à: `a >= b`
  - égal:`a == b`
  - différent: `a <> b`
  - NON logique (NON a): `!a`
  - ET logique (a ET b): `a & b`
  - OU logique (a OU b): `a | b`

A ces opérateurs s'ajoutent des fonctions trop nombreuses pour être citées ici. On trouve toutes les fonctions mathématiques de base (`log(a)`, `exp(a)`, `log10(a)`, `log2(a)`, `sin(a)`, `cos(a)`, `tan(a)`, `asin(a)`, `acos(a)`, `atan(a)`, `abs(a)`, `sqrt(a)`,...), ainsi que diverses fonctions utiles en statistiques sur lesquelles nous reviendrons.

Exercice : construisez un vecteur 8 nombres et appelez ce vecteur myVal. La fonction sum(vecteur) calcule la somme des valeurs comprises dans le vecteur. Utilisez cette fonction et des opérations les vecteurs pour calculer la moyenne de ces valeurs que vous appelerez myMean. Calculez ensuite la variance de ces valeurs (rappel, l'estimateur de la variance est la somme des carré des écarts à la moyenne, le tout divisé par le nombre d'observations – 1).

III.1.5. Importer ou créer un jeux de données

Créer un jeux de données Nous allons dans un premier temps créer un jeux de données en entrant directement les valeurs dans la console. Nous commençons par créer deux vecteurs. Le premier vecteur représente une variable qualitative qui indique le traitement effectué (ici 1 ou 2). Le second vecteur représente la variable quantitative mesurée sur chaque individus (ici un temps de vol d'insectes mesuré en secondes) :

```
Traitement = c(1,1,1,1,1,2,2,2,2,2)
DureeVol = c(423,235,400,368,410,700,753,689,750,677)
Traitement
```

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

```
DureeVol
```

```
##  [1] 423 235 400 368 410 700 753 689 750 677
```

Nous pouvons ensuite joindre ces deux vecteurs en un tableau, grâce à la fonction `data.frame()` qui va convertir l'ensemble des deux vecteurs en un tableau de données :

```
myTable = data.frame(Traitement,DureeVol)
myTable
```

```
##    Traitement DureeVol
## 1           1      423
## 2           1      235
## 3           1      400
## 4           1      368
## 5           1      410
## 6           2      700
## 7           2      753
## 8           2      689
## 9           2      750
## 10          2      677
```

Exercice : Créez un jeux de données comprenant votre moyenne annuelle de points dans une variable, et l'année dans la seconde variable (selon vos souvenirs, ou bien inventez !).

Charger un jeu de données R possède un certain nombres de jeux de données par défaut qui sont fourni à titre d'exemple. Pour charger ces jeux de données, il suffit d'utiliser la fonction `data()`. Nous pouvons d'abord obtenir la liste des jeux de données existants en tapant la commande :

```
data()
```

Il suffit ensuite d'entrer le nom du jeux de données que l'on souhaite charger dans la fonction data ()

```
data(islands)
islands
```

```
##           Africa       Antarctica             Asia        Australia
##            11506             5500            16988             2968
##     Axel Heiberg           Baffin            Banks           Borneo
##               16              184               23              280
##          Britain          Celebes            Celon             Cuba
##               84               73               25               43
##            Devon        Ellesmere           Europe        Greenland
##               21               82             3745              840
##           Hainan       Hispaniola         Hokkaido          Honshu
##               13               30               30               89
##          Iceland          Ireland             Java          Kyushu
##               40               33               49               14
##            Luzon       Madagascar         Melville         Mindanao
##               42              227               16               36
##         Moluccas      New Britain       New Guinea  New Zealand (N)
##               29               15              306               44
##  New Zealand (S)     Newfoundland    North America    Novaya Zemlya
##               58               43             9390               32
##  Prince of Wales         Sakhalin    South America      Southampton
##               13               29             6795               16
```

```
##      Spitsbergen          Sumatra           Taiwan        Tasmania
##               15              183               14              26
## Tierra del Fuego            Timor        Vancouver        Victoria
##               19               13               12              82
```

Lorsque le jeux de données est important (en nombre d'observations ou de variables), il devient très difficile d'afficher son contenu à l'écran. On utilise alors la fonction `str()` qui nous permet d'afficher la structure d'un objet. Nous pouvons par exemple charger le jeu de données « iris » et afficher sa structure :

```
data(iris)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Cette commande nous renseigne alors sur la nature de l'objet `iris`, ici en l'occurrence un « data.frame » (un jeux de données) qui comprends 150 observations et 5 variables : 4 variables numériques (« num » : Sepal.Length, Sepal.Width, Petal.Length, Petal.Width) et une variable qualitative non ordonnées (« Factor » : Species).

Exercice : chargez les jeux de données Nile et Orange, et inspectez leur structure à l'aide de la fonction `str()`. N'oubliez pas que vous pouvez toujours obtenir de l'aide sur un objet en tapant un ? avant celui-ci. En tapant `? Nile`, par exemple, vous aurez une aide sur ce jeux de données (ce ci ne s'applique qu'aux jeux de données fournis avec R).

Importer un jeux de données Pour importer un jeu de données existant, il y a plusieurs options possibles, selon le format des données d'origine. Si les données ont été encodées à l'aide d'un tableur, il faut tout d'abord exporter ces données au format texte, puis utiliser la fonction `read.table()`, `read.delim()` ou `read.csv()`. Nous illustrerons les fonctions d'importation de données par des exemples pratiques dans une séance ultérieure.

III.1.6. Manipulations sur les tableaux

Pour illustrer les opérations sur les tableaux, nous allons commencer par charger le jeu de données ToothGrowth, et afficher sa structure :

```
data(ToothGrowth)
str(ToothGrowth)
```

```
## 'data.frame':    60 obs. of  3 variables:
##  $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
##  $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
##  $ dose: num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
```

Changer le nom d'une variable Pour changer le nom de variables nous allons, par exemple, change les noms des trois variables du jeu de données « ToothGrowth ». Nous pouvons d'abord afficher le nom des trois variables :

```
names(ToothGrowth)
```

```
## [1] "len"  "supp" "dose"
```

Si nous voulons accéder au 2ème nom de variable, nous pouvons utiliser l'indexation, nous avons donc :

```
names(ToothGrowth)[2]
```

```
## [1] "supp"
```

Qui nous affiche le nom de la seconde variable. En utilisant l'assignation, nous pouvons remplacer ce nom par « Supplement », et vérifier par la commande `str(ToothGrowth)` que le changement a bien été effectué :

```
names(ToothGrowth)[2] = "Supplement"
str(ToothGrowth)
```

```
## 'data.frame':    60 obs. of  3 variables:
##  $ len       : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
##  $ Supplement: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
##  $ dose      : num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
```

Recoder une variable Il peut être utile de convertir une variable (entre les différents types : numérique, facteur, booléen). Dans le jeux de données « ToothGrowth », par exemple, seules 3 doses de vitamine C ont été testées dans l'expérience, et nous pouvons ici recoder cette variable numérique *quantitative* en variable *qualitative* :

```
ToothGrowth$dose = as.factor(ToothGrowth$dose)
str(ToothGrowth)
```

```
## 'data.frame':    60 obs. of  3 variables:
##  $ len       : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
##  $ Supplement: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
##  $ dose      : Factor w/ 3 levels "0.5","1","2": 1 1 1 1 1 1 1 1 1 1 ...
```

Nous voyons ici que le séparateur $ permet d'accéder à une variable à l'intérieur d'un tableau de données. Ainsi, `ToothGrowth$len` est le vecteur de 60 observations de la variable len dans le jeu de données ToothGrowth.

Ajouter une variable On peut utiliser cet accès aux variables d'un tableau pour créer de nouvelles variables, par exemple, les commandes :

```
ToothGrowth$Test = 1
ToothGrowth$Loglen = log(ToothGrowth$len)
```

ont créé deux nouvelles variables Test et Loglen qui contiennent la value 1 pour la première, et le logarithme en base e de la variable len pour la seconde, comme on peut le vérifier avec la fonction `str()`.

```
str(ToothGrowth)
```

```
## 'data.frame':    60 obs. of  5 variables:
##  $ len       : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
##  $ Supplement: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
##  $ dose      : Factor w/ 3 levels "0.5","1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Test      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Loglen    : num  1.44 2.44 1.99 1.76 1.86 ...
```

Système d'indexation On a vu que le système d'indexation permet d'accéder aux valeurs à l'intérieur d'un vecteur. Il en va de même pour accéder aux valeurs d'un tableau, sauf que l'indexation se fait avec deux dimensions, les lignes et les colonnes. De manière générale, l'indexation accède aux données par [no de ligne,no de colonne], par exemple :

```
ToothGrowth[2,3]
```

```
## [1] 0.5
## Levels: 0.5 1 2
```

Nous renvoie la valeur de la 2ème ligne, et 3ème colonne du jeu de données ToothGrowth. On peut de la même manière, accéder à toute une ligne :

```
ToothGrowth[2,]
```

```
##    len Supplement dose Test Loglen
## 2 11.5         VC  0.5    1  2.442
```

Ou à toute une colonne :

```
ToothGrowth[,3]
```

```
##  [1] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 1   1   1   1   1   1   1
## [18] 1   1   1   2   2   2   2   2   2   2   2   2   2   0.5 0.5 0.5 0.5
## [35] 0.5 0.5 0.5 0.5 0.5 0.5 1   1   1   1   1   1   1   1   1   1   2
## [52] 2   2   2   2   2   2   2   2   2
## Levels: 0.5 1 2
```

Enfin, l'indexation peut même être utilisée pour supprimer une ligne ou une colonne dans un tableau. Par exemple, la commande `ToothGrowth[,-4]` permet de supprimer la 4ème colonne (la variable « Test ») du jeux de données:

```
ToothGrowth = ToothGrowth[,-4]
str(ToothGrowth)
```

```
## 'data.frame':    60 obs. of  4 variables:
##  $ len       : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
##  $ Supplement: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
##  $ dose      : Factor w/ 3 levels "0.5","1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Loglen    : num  1.44 2.44 1.99 1.76 1.86 ...
```

Extraire un sous jeux de données

La fonction subset() permet d'accéder à une partie du jeux de données, définie par une, ou une combinaison d'opérateurs logiques. Dans l'exemple ci-dessous :

```
TG1 = subset(ToothGrowth, dose == 1)
TG2 = subset(ToothGrowth, len >= 10)
```

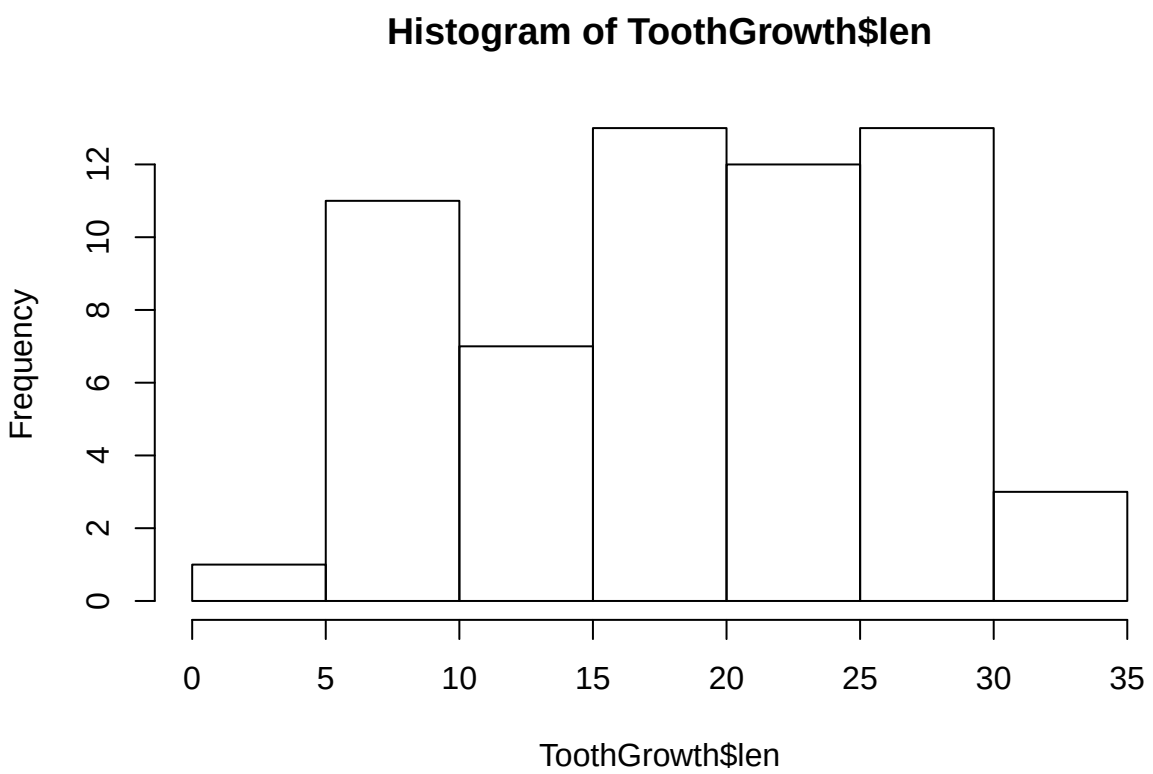
Crée respectivement deux nouveaux jeux de données, `TG1` et `TG2`. Le premier comprends toutes les observations ayant une dose de 1 ; le second comprends toutes les observations dont la variable len est supérieure ou égale à 10, comme on pourra le vérifier avec les fonctions `str(TG1)` et `str(TG2)`.

Exercice : chargez le jeux de données « Orange ». Créez une nouvelle variable appelée LogCirc comprenant le logarithme de la circonférence. Affichez les lignes 5 à 12 du jeu de données. Créez un jeu de données appelé OrangeYoung ne comprenant que les arbres ayant un âge inférieur à 1000 jours. Affichez la circonférence de la 4ème observation de ce jeu de données.

III.1.7. Créer et modifier un graphique

Nous avons déjà vu que la fonction `hist()` nous permettait de faire un graphique de l'histogramme d'un vecteur. Il existe de très nombreuses fonctions graphiques dans R. Les principales fonctions que nous verrons aux travaux pratiques sont les fonctions `hist()`, `boxplot()` et `plot()`. Ces graphiques seront présentés individuellement dans les sections qui suivent. Par exemple :

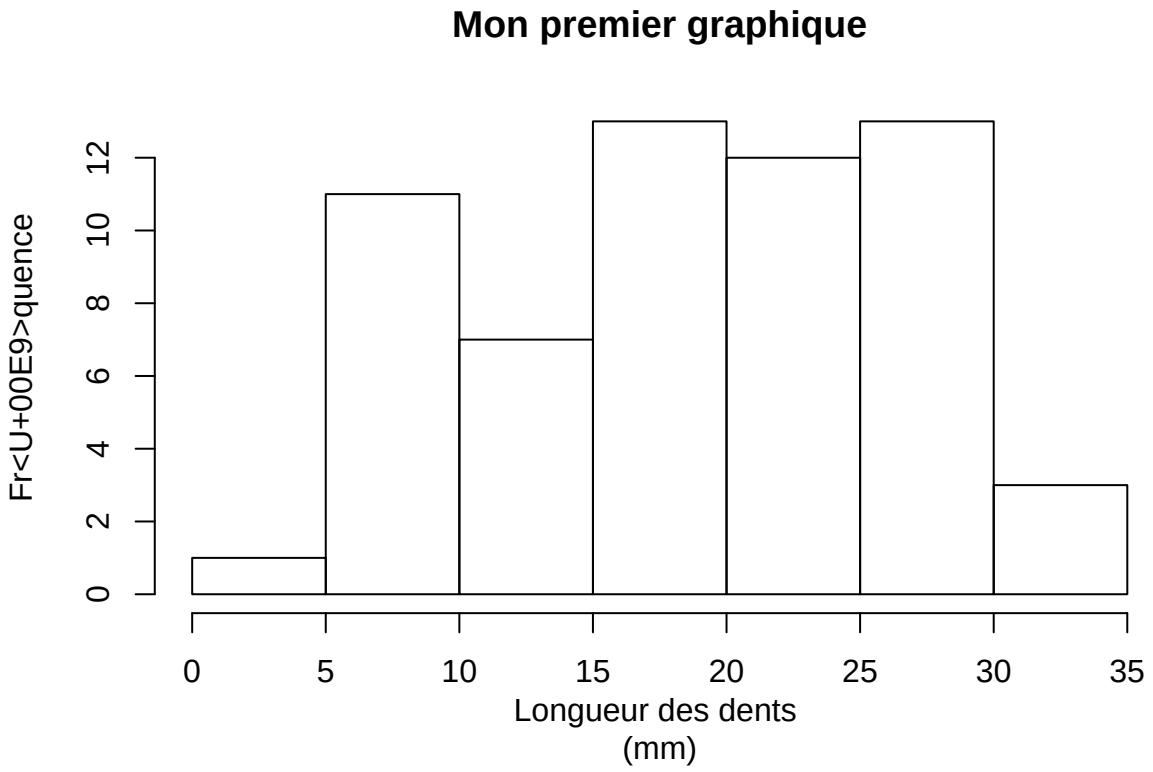
```
hist(ToothGrowth$len)
```



Nous présente l'histogramme de fréquence de la variable ToothGrowth$len. L'aide de la fonction hist (accessible en tapant `? hist()`) donnera les différentes options qui peuvent être utilisées dans l'affichage de ce graphique.

Il existe un certain nombre d'options génériques qui s'appliquent à tous les graphiques. Ces options sont i) `main = «titre»` qui permet de changer le titre du graphique ; `xlab = «nom axe des x»` qui permet de changer la légende de l'axe des abscisses ; `ylab = « nom axe y»` qui permet de changer la légende de l'axe des ordonnées. Par exemple, pour avoir un graphique personnalisé, on écrira :

```
hist(ToothGrowth$len, main = "Mon premier graphique", xlab = "Longueur des dents
(mm)", ylab = "Fréquence")
```



III.2. Statistiques descriptives

III.2.1. La fonction « summary »

La fonction `summary()` nous donne un certain nombre de statistiques descriptives sur l'objet auquel elle est appliquée. Par exemple :

```
data(iris)
summary(iris)
```

```
##   Sepal.Length   Sepal.Width    Petal.Length   Petal.Width
##  Min.   :4.30   Min.   :2.00   Min.   :1.00   Min.   :0.1
##  1st Qu.:5.10   1st Qu.:2.80   1st Qu.:1.60   1st Qu.:0.3
##  Median :5.80   Median :3.00   Median :4.35   Median :1.3
##  Mean   :5.84   Mean   :3.06   Mean   :3.76   Mean   :1.2
##  3rd Qu.:6.40   3rd Qu.:3.30   3rd Qu.:5.10   3rd Qu.:1.8
##  Max.   :7.90   Max.   :4.40   Max.   :6.90   Max.   :2.5
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Affiche des statistiques descriptives sur le jeu de données iris (chargé à la première ligne). On peut voir que pour les variables quantitatives, la fonction nous renvoie la valeur minimum, le premier quartile (valeur qui

sépare le jeux de données avec 25% des observations à gauche), la médiane (valeur qui sépare le jeux de données en deux jeux de données avec le même nombre d'observations), la moyenne arithmétique, le 3ème quartile (valeur qui sépare le jeux de données avec 75% des observations à gauche), et la valeur maximale. Pour les variables qualitatives (ici par exemple la variable « Species »), la fonction renvoie la fréquence des observations dans chacun des niveaux de la variables qualitative.

III.2.2. Fonctions statistiques de base

Les mesures principales de position sont les fonctions `mean()` et `median()` qui renvoient respectivement la moyenne et la médiane du vecteur transmis comme argument. La principale mesure dispersion est l'écart type que l'on obtient à l'aide de la fonctions `sd()`. Par exemple, la moyenne, médiane, et l'écart type de la variable len du jeu de données ToothGrowth s'obtiennent donc de la manière suivante :

```
mean(ToothGrowth$len)
```

```
## [1] 18.81
```

```
median(ToothGrowth$len)
```

```
## [1] 19.25
```

```
sd(ToothGrowth$len)
```

```
## [1] 7.649
```

On peut aussi obtenir ces même statistique, mais groupées par une second variable qui définit les niveaux à l'aide de la fonction `by` dans laquelle le premier argument est le vecteur sur lequel on veut calculer une statistique, le second argument est le vecteur de groupement (ici la variable Supplement), et le troisième argument est la statistique que l'on veut calculer:

```
by(ToothGrowth$len, ToothGrowth$Supplement, mean)
```

```
## ToothGrowth$Supplement: OJ
## [1] 20.66
## ------------------------------------------------------------
## ToothGrowth$Supplement: VC
## [1] 16.96
```

```
by(ToothGrowth$len, ToothGrowth$Supplement, sd)
```

```
## ToothGrowth$Supplement: OJ
## [1] 6.606
## ------------------------------------------------------------
## ToothGrowth$Supplement: VC
## [1] 8.266
```

III.2.3. Corrélation et matrice de corrélation

Pour illustrer les fonctions de mesure de corrélation, nous allons utiliser le jeu de données iris et la fonction `cor()`. Nous pouvons obtenir soit un cœfficient de corrélation entre 2 variables, soit entre un ensemble de variables :

```r
# Corrélation entre 2 variables
cor(iris$Sepal.Length,iris$Petal.Length)
```

```
## [1] 0.8718
```

```r
# Corrélation entre les variables 1 à 4 du jeux de données iris
cor(iris[,1:4])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length       1.0000     -0.1176       0.8718      0.8179
## Sepal.Width       -0.1176      1.0000      -0.4284     -0.3661
## Petal.Length       0.8718     -0.4284       1.0000      0.9629
## Petal.Width        0.8179     -0.3661       0.9629      1.0000
```

La première ligne nous donne le cœfficient de corrélation entre les variables Sepal.Length et Petal.Length. A la deuxième ligne, nous demandons une matrice de cœfficients de corrélation entre les 4 première variables du jeu de données iris (rappelez vous de l'indexation qui nous permet ici de passer directement les 4 variables à la fonction). Notez l'utilisation de commentaires précédé du charactère `#`. Celui-ci peut **toujours** être ajouté à votre code pour le commenter, noter des information, des résultats, ds interprétations, etc.... Par défaut, la fonction `cor()` renvoie la valeur du cœfficient de corrélation de Pearson, mais un cœfficient de Spearman peut être obtenu à l'aide de l'argument method :

```r
cor(iris$Sepal.Length,iris$Petal.Length, method = "spearman")
```

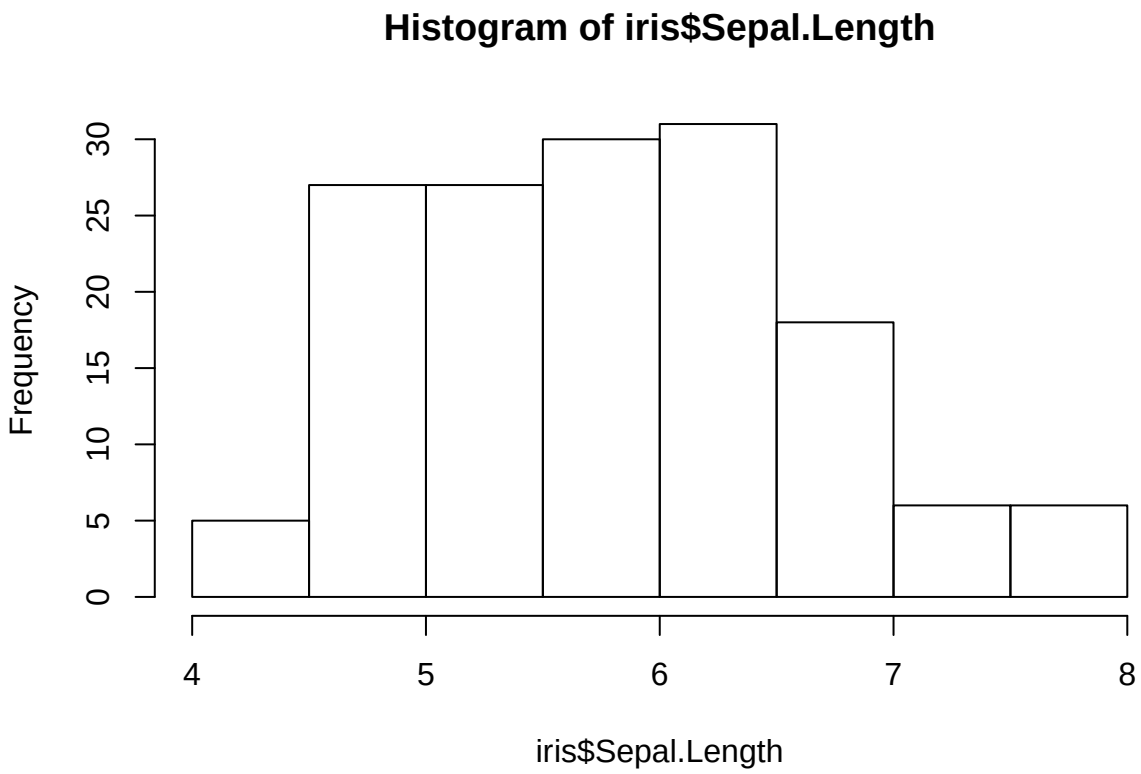
```
## [1] 0.8819
```

III.3. Graphiques univariés

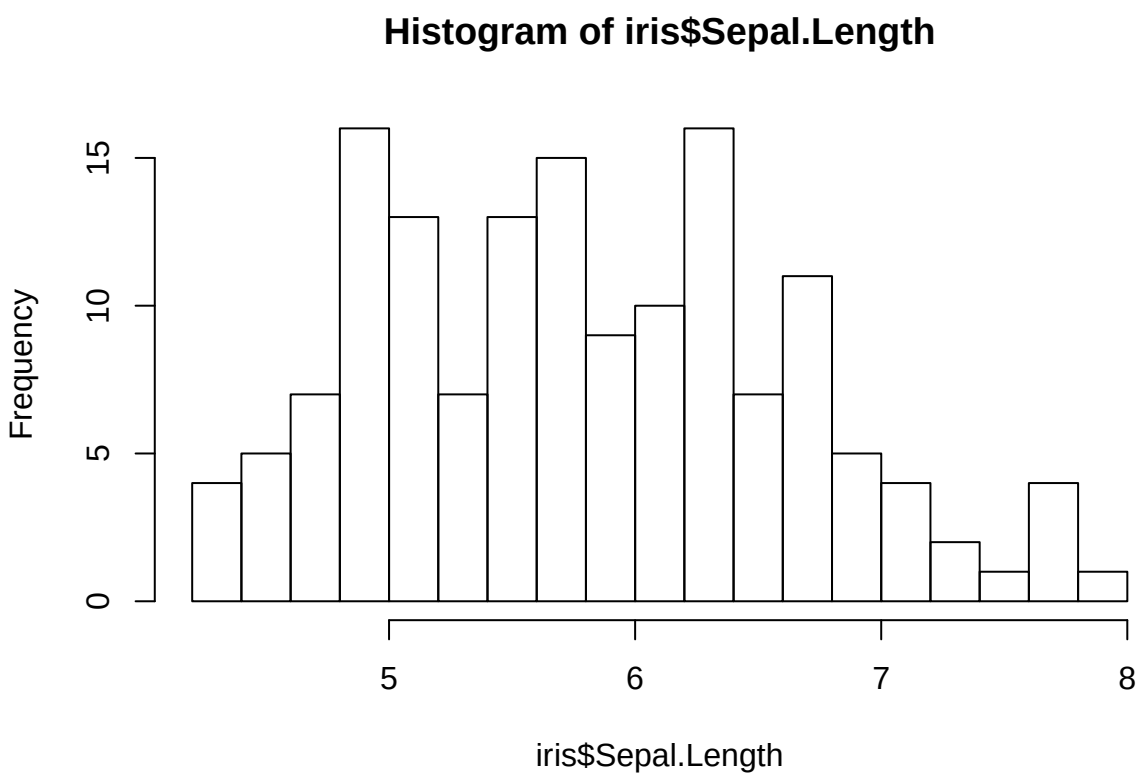
III.3.1. Histogramme de fréquence

Comme nous l'avons déjà vu, l'histogramme de fréquence d'une variable s'obtient à l'aide de la fonction `hist()`. La fonction calcule un nombre de classes équidistantes. Le nombre de classes, ou l'intervalle entre les classes peut néanmoins être modifié à l'aide de l'argument `breaks`. Par exemple, les fonctions suivantes renvoient deux histogrammes avec des nombres de classes différents :

```r
hist(iris$Sepal.Length)
```



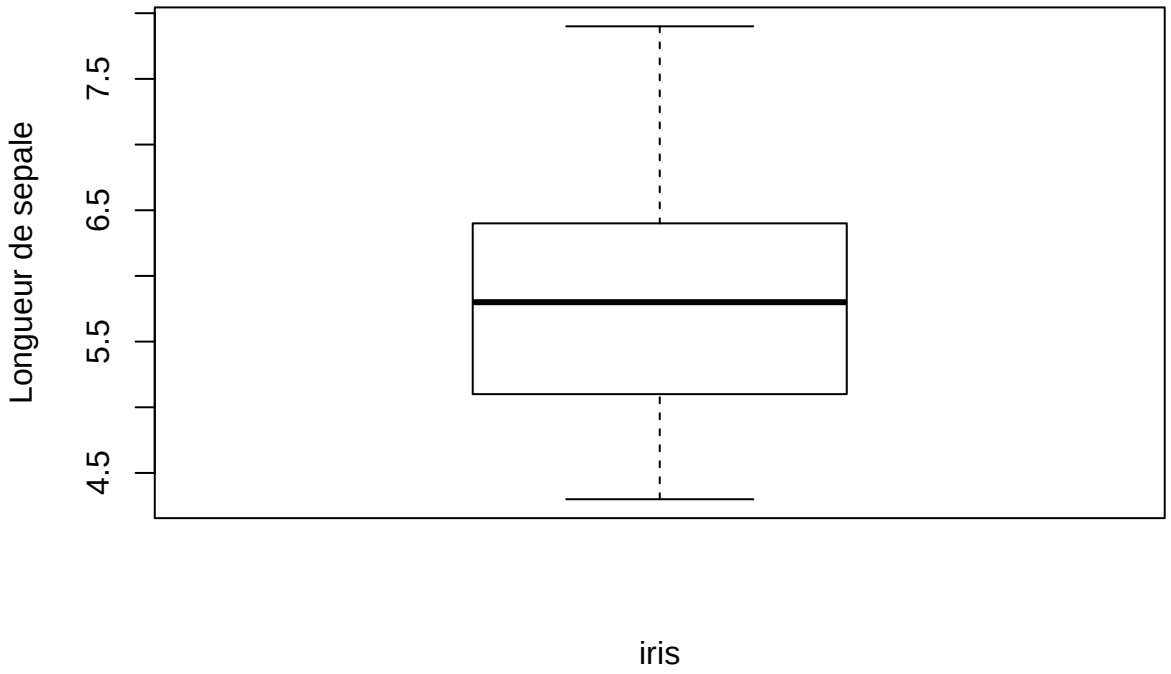

```
hist(iris$Sepal.Length, breaks = 20)
```



III.3.2. Boite de dispersion

La fonction `boxplot()` permet d'afficher le diagramme de dispersion pour une seule variable, ou pour une variable groupée par un facteur à n niveaux. Par exemple, la fonctions suivantes affichent la boite de dispersion de la variable Sepal.Length (jeu de données iris)

```
boxplot(iris$Sepal.Length, xlab = "iris", ylab = "Longueur de sepale")
```



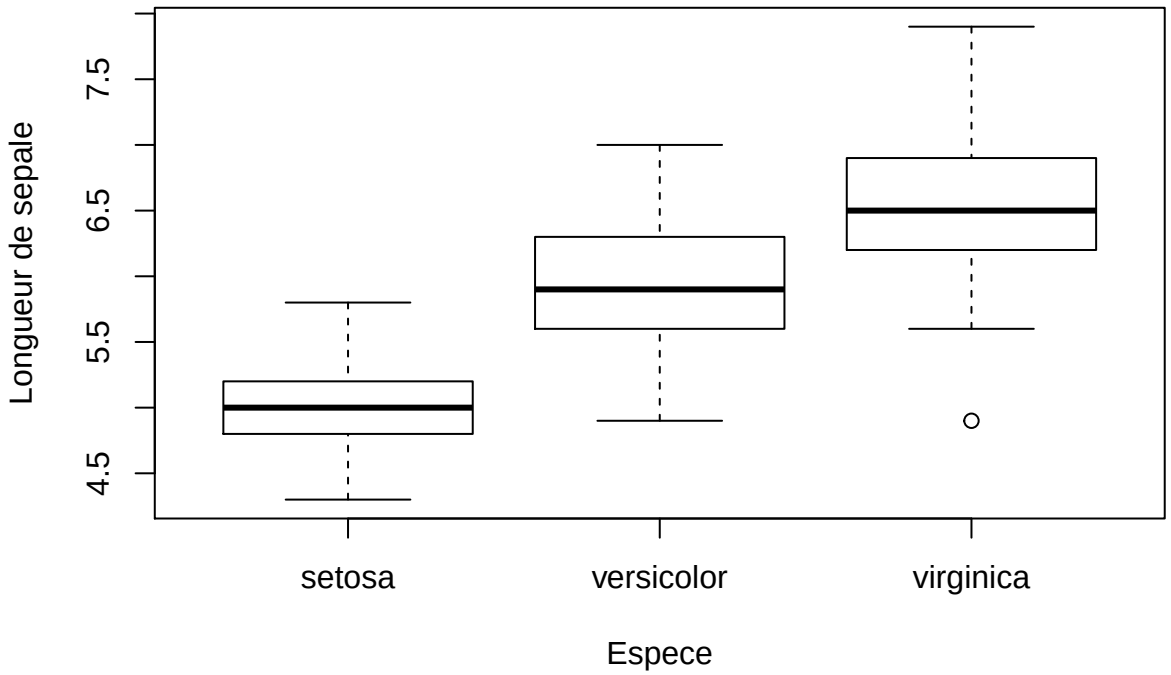
Pour rappel, le graphe de dispersion vous indique la médiane (ligne horizontale en gras au centre de la boite), les premiers et troisièmes quartiles (limites de la boîte). Les limites des barres présentent 1.57 fois l'espace interquartile / la racine carrée du nombre de points. Cette estimation a pour objectif de donner une estimation approximative d'un intervalle de confiance de 95% autour de la médiane. Les points présentés en dehors de cette boîte sont tous les points dont les valeurs s'écartent de cet intervalle. L'interprétation de la boîte de dispersion est très importante : elle permet de juger de la symétrie des distributions (les espaces entre la médiane et les quartiles sont-ils les mêmes de part et d'autre de la médiane ?), et de la dispersion autour de la médiane.

III.4. Graphiques bi-variés

III.4.1. Boite de dispersion

La boite de dispersion permet de représenter la distribution d'une variable quantitative en fonction d'une variable qualitative à plusieurs niveaux. Dans l'exemple présenté ci-dessous, nous affichons la boîte de dispersion de la variable `Sepal.Length` pour chaque espèce du jeu de données iris (variable de groupement Species) :

```
boxplot(Sepal.Length ~ Species, data = iris, xlab = "Espece", ylab = "Longueur de sepale")
```

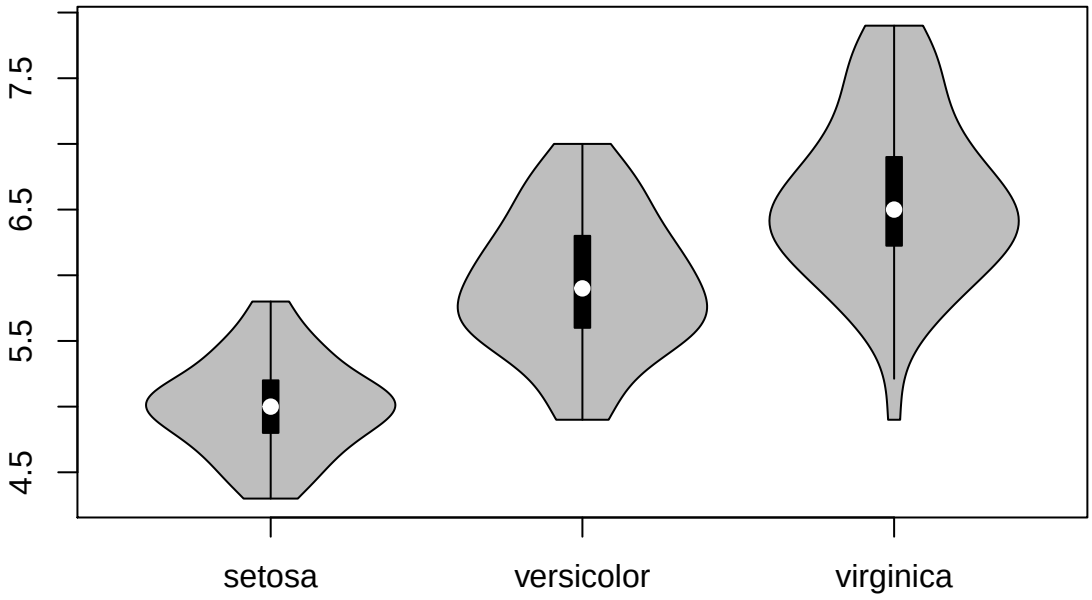


Si vous avez la possibilité d'installer un nouveau package (avec la fonction `install.packages("vioplot")`), installez le package `vioplot`, et executez les lignes suivantes pour obtenir un violin plot des mêmes données

```
library(vioplot)
```

```
## Loading required package: sm
## Package 'sm', version 2.2-5.4: type help(sm) for summary information
```

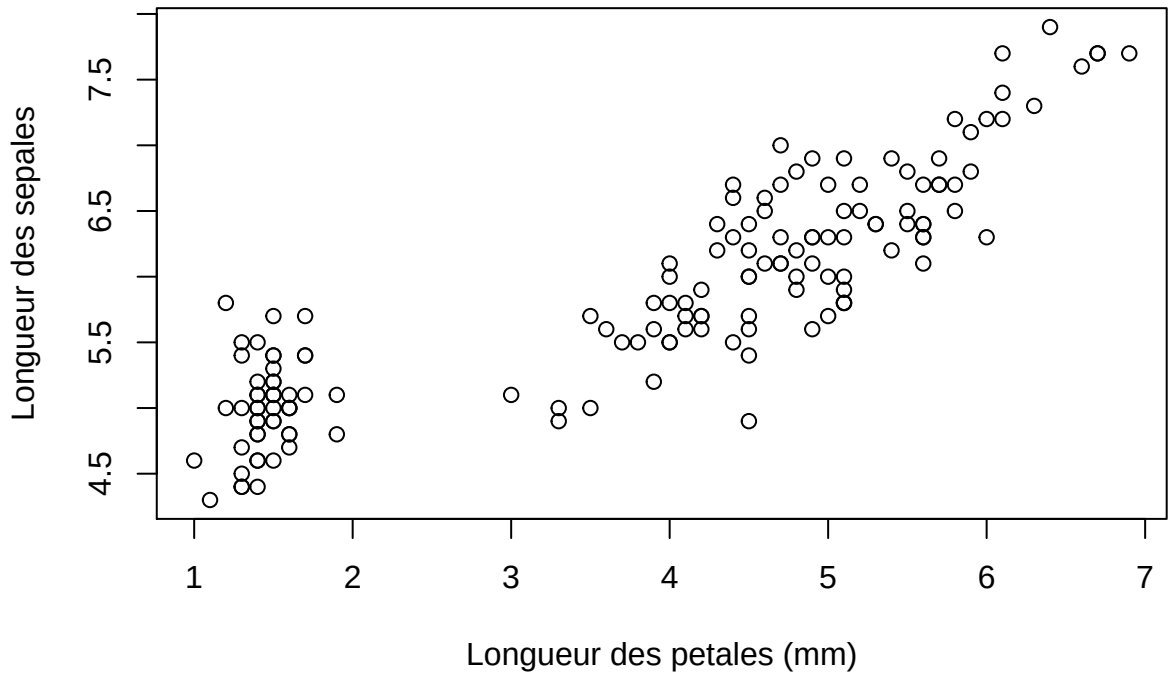
```
vioplot(iris[iris$Species == "setosa","Sepal.Length"],
        iris[iris$Species == "versicolor","Sepal.Length"],
        iris[iris$Species == "virginica","Sepal.Length"],
        col = "grey",
        names = c("setosa","versicolor","virginica"))
```



III.4.2. Nuage de points

Le nuage de point permet de représenter graphiquement la relation entre deux variables quantitatives et s'obtient à l'aide de la fonction `plot()`. Cette fonction a beaucoup d'autres usages, comme nous le verrons plus loin, mais nous nous limitons pour le moment à illustrer son utilisation pour construire un graphique en nuages de points.

```
plot(iris$Petal.Length, iris$Sepal.Length, xlab = "Longueur des petales (mm)"
     , ylab = "Longueur des sepales")
```



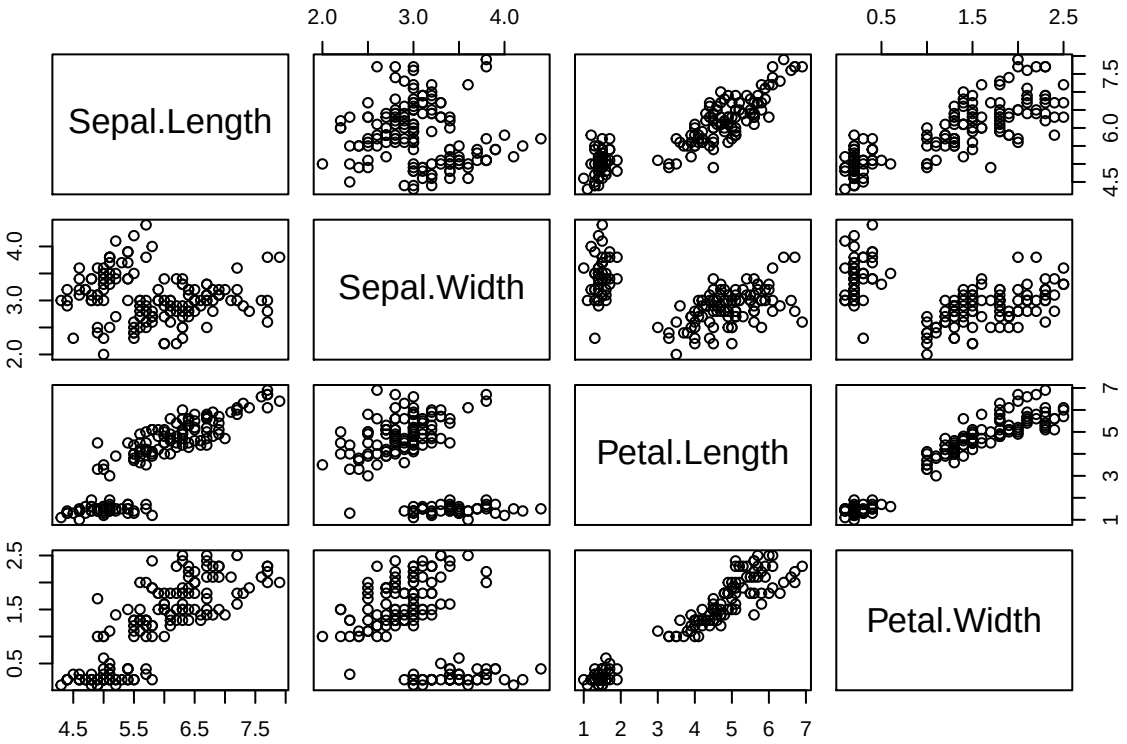
Ce graphique peut également être obtenu de la manière suivante :

```
plot(Sepal.Length ~ Petal.Length, data = iris, xlab = "Longueur des petales (mm)"
     , ylab = "Longueur des sepales")
```

III.4.3. La matrice de nuages de points

La matrice de nuages de points est une manière de représenter les relations bi-variées entre plusieurs variables quantitatives. Les nuages de points de chaque paire de variables sont représentés aux différentes cellules de la matrice. Dans la commande illustrée ci-dessous, on tire encore profit de l'indexation pour représenter la matrice de corrélation entre les 4 premières variables du jeu de données iris.

```
pairs(iris[,1:4])
```



Exercice : chargez le jeux de données « Orange ». Calculez la moyenne et l'écart type des variables age et circumference. Affichez l'histogramme de ces deux variables. Calculez la corrélation entre ces deux variables. Affichez les boîtes de dispersion de la variable circumference selon la variable Tree. Affichez le nuage de point de la variable circumference en fonction de la variable age.

III.5 Exercices supplémentaire de statistiques descriptives

- Au choix, chargez l'un de ces jeux de données : *swiss*, *longley*, *OrchardSprays* et *PlantGrowth* (utilisez la fonction data(swiss), par exemple)
- Décrivez le jeu de données :
  - De quoi s'agit-il ? De quelle expérience ? Qu'est-ce qui est mesuré ?
  - Combien il y a-t-il d'observations ? de variables ? Quels types de variables ?
  - Présentez vos données sous forme numérique (moyennes, variance, médianes, quartiles)
  - S'il existe une variable de groupement (variable qualitative à plusieurs niveaux), quelles sont les moyennes et écart-types par groupe ?
- Présentez vos données sous forme de graphiques :
  - Univariés
  - Bivariés
- Si vous avez terminé avec un jeu de données, procédez de la même manière avec un second jeu de données.