

A photograph of a modern building with a glass facade and a courtyard. The building has a white brick wall on the left and a glass facade on the right. The courtyard is filled with green plants and a paved path. The sky is blue with a few clouds.

Commandes et objets de base

Vincent Guillemot
Amoury Vaysse
Lundi 15/11/2021

swiss pavilion

ONICS

Bien se préparer à coder

- Créer un projet dans ce dossier :
 - Bien réfléchir au nom de ce projet
 - Bien réfléchir à l'endroit où ce projet sera situé
- Créer un script
- Savoir où sont les données, et sous quel format
- Savoir ce que l'on veut faire !

C'est parti pour les commandes de base !

Le(s) prompt(s)

- > : R attend une commande à exécuter
- + : la commande qui a été entrée n'est pas complète car
 - il manque une parenthèse fermante
 - il manque un crochet fermant
 - il manque une accolade fermante
- : la commande est en cours de traitement. On peut l'arrêter en cliquant sur le bouton "Stop"

Une commande, c'est quoi ?

C'est une séquence d'opérations appliquées à des objets ou des valeurs qui sera donnée au compilateur de R pour interprétation.

- “opération” : permet de transformer des entrées en sorties, p.ex. : la fonction `log`;
- “objets” : structures de données, p.ex. : plus tard ;
- “valeurs” : des briques de base, p.ex. : la valeur 0.

Une fois la commande écrite dans la console, on l'exécute en appuyant sur “Entrée”.

Exercice

Exécutez les commandes suivantes :

1. `1 + 1`

2. `1+1`

3. `log(10)`

4. `log10(10)`

Remarquez l'utilisation des parenthèses pour appliquer une fonction !

Opérations de base

On peut effectuer toutes les opérations de base en R :

- addition (+), soustraction (-), multiplication (*), division (/), exponentiation (** ou ^)...

appliquer les fonctions mathématiques de base :

- logarithme (log, log2, log10), exponentielle (exp) sinus (sin), cosinus (cos), tangente (tan)

On peut combiner les opérations et les fonctions, et gérer les priorités avec des parenthèses !

Exercice

Imaginez une commande incluant le plus d'opérations et de fonctions de base et dont le résultat serait 2 !

Un opérateur bien pratique, le :

Comment créer une suite d'entier ?

- `c(1, 2, 3, 4)`
- `seq(1, 4, 1`
- `1:4`

L'opérateur `:` est très utilisé en R. Sa syntaxe est la suivante

- `i:j` va créer une suite d'entiers de `i` à `j`. Les entiers peuvent être négatifs ou positifs, et on peut avoir `i < j` ou `i > j`, ou même `i = j`.

Attention à bien mettre des **parenthèses** dans le cas d'entiers négatifs!

Exemples

- $3:7$: les entiers de 3 à 7
- $7:3$: les entiers de 7 à 3
- $-3:7$: les entiers de -3 à 7
- $-3:-7$: les entiers de -3 à -7
- $-(3:7)$: les entiers de -3 à -7

Exercice

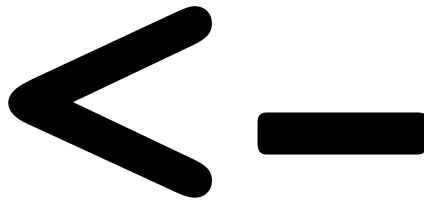
1. Créez un vecteur d'entiers de 0 à 3
2. Créez un autre vecteur d'entiers de 3 à 1
3. *Combinez* ces deux suites pour obtenir le vecteur suivant :

3	2	1	0	1	2	3
---	---	---	---	---	---	---

Assignation avec <-

Comment “sauvegarder” ces objets ?

En utilisant l’opérateur d’assignation



Utilisation de l'opération d'assignation

- A gauche : l'objet que l'on veut créer
- A droite : sa définition
- Lecture de l'opération : "assigner à cet objet (à gauche) le résultat de cette commande (à droite)"
- Assigner deux fois de suite écrasera la première valeur assignée
- Alternative : =
- Exemples : `a <- 1`, `b <- 1:10`, `a <- 2` etc.

Les noms d'objets

Règles absolues :

- Commence par une lettre ou un point, si le premier caractère est un point, le deuxième ne peut pas être un chiffre,
- Pas d'espace
- Pas de caractères correspondant à des opérations (+, -, *, /, ^, **, etc.)
- Les minuscules et les majuscules sont différentes !
- Certains "mots-clefs" sont strictement interdits (NA, TRUE, FALSE, for, if, else etc.)
- MAIS on peut utiliser un nom d'objet qui existe déjà !

Bonnes pratiques :

- Utiliser un nom qui a du sens
- Ne pas utiliser des noms d'objets qui existent déjà et que l'on ne souhaite pas écraser !

Mini exercice

1. Créez un vecteur d'entiers de 0 à 3, appelez le `a`
2. Créez un autre vecteur d'entiers de 3 à 1, appelez le `b`
3. *Combinez* ces deux vecteurs pour obtenir un vecteur `ab` selon le modèle suivant :

3	2	1	0	1	2	3
---	---	---	---	---	---	---

Opérations

On peut appliquer des opérations à ces “vecteurs” !

```
a <- 1:5  
a + 1  
#> [1] 2 3 4 5 6  
a * 2  
#> [1] 2 4 6 8 10
```

Ces “vecteurs” sont des **objets**.

Les Objets

Classes d'objets

Nom	Appelation officielle	Exemple
Vecteur	???	<code>1:10</code>
Facteur	<code>factor</code>	<code>gl(2, 2)</code>
Matrice	<code>matrix</code>	<code>matrix(1:4, 2, 2)</code>
Tableau	<code>data.frame</code>	<code>mtcars</code>
Liste	<code>list</code>	<code>list(a = 1, b = 1:10, c = "Hello!")</code>
Fonction	<code>function</code>	<code>sin, exp, log</code>

Pour connaître la classe d'un objet : `class(objet)`.

Types de données

Nom	Appellation officielle	Exemple
Entier (\mathbb{Z})	<code>integer</code>	<code>1:10</code> , (ou <code>1L</code>)
Réel (\mathbb{R})	<code>double</code>	<code>2.3</code> , <code>1/3</code> , etc...
Caractères	<code>character</code>	<code>month.name</code> , <code>"Bonjour"</code>
Booléen	<code>logical</code>	<code>TRUE</code>

Bouh les quoi ?

MATH., néol. Qui est relatif aux théories du logicien et mathématicien anglais George Boole.

– *Trésor de la Langue Française informatisé*

- TRUE (ou bien T) et FALSE (ou bien F)
- Résultat d'une comparaison : ==, !=, <, >, <=, >=
- Opérations logiques : !, &, |, xor

Exercice

1. Effectuez les opérations suivantes :

- `1 == 2`
- `!(5 > -6)`
- `(1 <= 10) | (1 > 0)`

1. Prédisez le résultat de la commande suivante : `log(1) != 0`

“Classification” des objets

Les objets qui ne contiennent qu’un seul type de données : vecteurs et matrices.

Les objets pouvant contenir des données mixtes : tableaux et listes.

La flexibilité a un coût : on ne peut plus faire certaines opérations !

Les objets ayant des “dimensions” : vecteurs, tableaux et matrices

Les objets pour qui cela ne signifie rien ou presque : listes et fonctions

Petit détour par les tableaux

```
data("fruits", package = "debuter")
fruits
#> # A tibble: 51 × 18
#>   nom      groupe Energie   Eau Proteines Glucides Lipides Sucres
#>   <chr>    <chr>    <dbl> <dbl>    <dbl>    <dbl>    <dbl> <dbl>
#> 1 Abricot   crus      194  87.1     0.81     9.01     0.25   6.7
#> 2 Abricot   secs     1010  24.7     2.88    59.1     0.5    34.3
#> 3 Ananas    exoti...   304  81.3     0.94    15.1     0.25   14.9
#> 4 Banane    exoti...   383  75.8     1.06    19.7     0.25   15.6
#> 5 Canneberge secs     1410  14.6     0.25    76.4     1      72.8
#> 6 Cerise    crus      235  85.7     0.81    13      0.25   10
#> 7 Citron    crus      118  91.3     0.25    1.56    0.25   0.8
#> 8 Clementine crus      200  87      0.81    9.17    0.25   8.6
#> 9 CompoteMul... compo...   279  82.9     0.25    15.3    0.08   14.6
#> 10 CompotePom... compo...   432  72.9     0.23    24.4    0.21   20.7
#> # ... with 41 more rows, and 10 more variables: Fructose <dbl>,
#> #   Fibres <dbl>, Calcium <dbl>, Magnesium <dbl>, Phosphore <dbl>,
#> #   Potassium <dbl>, Zinc <dbl>, BetaCarotene <dbl>,
#> #   VitamineE <dbl>, VitamineC <dbl>
```

Importer des données en R

- Des données de *packages* : `data`
- Des données au format R (`RData`) : `load`
- Des données "tabulées" : `read.table`
- Des données Excel : `readxl::read_excel`
- Des données Stata, SPSS, images etc.

Les données “de R”

- Utiliser la commande `data()` pour avoir une liste (presque ?) exhaustive.
- Bonne pratique : pour charger un jeu de données, utiliser la commande complète `data("nom_des_data", package = "nom_du_package")`
- Mais ces alternatives fonctionnent également :
 - `data(mtcars)`
 - `DNase`
 - `library(ggplot2) ; data(diamonds)`

Utilisation des guillemets

- Obligation : quand l'argument doit être une chaîne de caractères
- Oubli : `library, require, data`
- Guillemets simples : fonctionnent comme les guillemets doubles. Ex.:
"bonjour" est équivalent à 'bonjour'.
- Le "backtick" ou "backquote" : "`"

Bref, c'est quoi un **tibble** ?

Mise à jour de la table des classes d'objets !

Nom	Appelation officielle	Exemple
Vecteur	???	<code>1:10</code>
Facteur	<code>factor</code>	<code>gl(2, 2)</code>
Matrice	<code>matrix</code>	<code>matrix(1:4, 2, 2)</code>
Tableau	<code>data.frame</code>	<code>mtcars</code>
Tableau	<code>tibble</code>	<code>fruits</code>
Liste	<code>list</code>	<code>list(a = 1, b = 1:10, c = "Hello!")</code>
Fonction	<code>function</code>	<code>sin, exp, log</code>

Explorer les données `fruits`

[]

[,]

[[]]

\$

Sur quels objets les utiliser ?

Opérateur	Vecteurs	Matrices	Tableaux	Listes
[]	X		X	X
[,]		X	X	
[[]]			X	X
\$			X	X

L'opérateur de sélection classique : [,]

- Pour sélectionner la première ligne : `fruits[1,]`
- Pour sélectionner la deuxième colonne : `fruits[, 2]`
- Pour enlever la troisième ligne : `fruits[-3,]`
- Pour enlever la quatrième colonne : `fruits[, -4]`

Exercice

Comment faire pour sélectionner les fruits numéro 1, 3 et 5 ?

Je veux deux solutions : une “normale” et une “créative” !

Sélectionner plusieurs lignes / colonnes

- Pour sélectionner les lignes 1 et 3 : `fruits[c(1, 3),]`
- Pour sélectionner les colonnes 2 et 4 : `fruits[, c(2, 4)]`
- Pour enlever les lignes 5 et 7 : `fruits[-c(5, 7),]`
- Pour enlever les colonnes 6 et 8 : `fruits[, -c(6, 8)]`

De l'utilité des deux points

Pour sélectionner une plage entière de lignes ou de colonnes adjacentes :

- Pour sélectionner les lignes 11 à 17 : `fruits[11:17,]`
- Pour sélectionner les colonnes 3 à 5 : `fruits[, 3:5]`
- Pour enlever les trois premiers fruits : `fruits[-(1:3),]`
- Pour enlever les cinq premières variables : `fruits[, -(1:5)]`

Exercice

Que se passe-t-il quand on oublie les parenthèses dans la commande `fruits[-(1:3),]` ? Commentez !

Faites de même avec le jeu de données `mtcars`.

Pour extraire une seule colonne : le \$

La syntaxe `donnees$cible` permet de sélectionner la colonne `cible` du tableau `donnees`.

- Par exemple : `fruits$Eau`
- Autre exemple : `fruits$groupe`

Exercice

Extrayez la colonne de la teneur en sucres de la table des fruits... de deux façons différentes !

Créez un objets contenant la teneur en sucres : quelle est la classe de cet objet ?

Les vecteurs...

- ... sont “unidimensionnels”
- ... ont une classe qui est égale au type de données qu’ils contiennent (R !!!)
- ... sont indexés avec des crochets simples

Exemples :

- `i <- 1:10`
- `eau <- fruits$Eau`
- `eau[i]`

Exercice

Créez un vecteur `groupe` contenant les groupes de fruits. Donnez deux façons différentes d'extraire les dix premières valeurs de ce vecteur.

Extraction avec des Booléens

Comment extraire les fruits ... * dont la teneur en eau est supérieure à 60 ? * exotiques ? * secs contenant moins de 40g/100g de sucres ?

Réponse : en utilisant des vecteurs booléens

1. Créer le vecteur de booléens `fruits$Eau >= 60`
2. Utiliser le résultat dans les crochets carrés `fruits[fruits$Eau >= 60,`
]

Ne pas oublier la virgule !

Le principe

Pour un vecteur `v` :

- `v[bool]` extrait les valeurs de `v` pour lesquelles `bool` est vrai (TRUE).
Contrainte : `v` et `bool` doivent contenir **le même nombre d'éléments**.

Pour un tableau `tab` :

- `tab[brow,]` pour extraire les lignes
- `tab[, bcol]`
- Contrainte 1 : `brow` doit avoir autant d'éléments que `tab` de lignes
- Contrainte 2 : `bcol` doit avoir autant d'éléments que `tab` de colonnes

Attention

Vous verrez souvent des opérations logiques à l'intérieur des crochets carrés : cela permet d'aller plus vite !

Par exemple, en deux étapes :

```
1. bool <- fruits$groupe == "secs" & fruits$Sucres < 40  
2. fruits[bool, ]
```

Devient, en une étape :

```
• fruits[fruits$groupe == "secs" & fruits$Sucres < 40, ]
```

Attention bis

On peut combiner deux méthodes d'extraction de données pour un tableau : une sur les lignes et une sur les colonnes !

Par exemple : `tab[brow, icol]`, où `brow` est un vecteur de booléens et `icol` un vecteur d'indices.

Exercice

Construisez la sous-table contenant la teneur en protéines, en glucides et en lipides des fruits secs.

Les objets nommés

En R, on peut donner des “noms”...

- aux éléments d'un vecteur,
- aux lignes d'un tableau ou d'une matrice,
- aux colonnes d'un tableau ou d'une matrice,
- aux éléments d'une liste

Pourquoi ? Pour pouvoir disposer d'une nouvelle méthode d'extraction de données !

Pour un tableau

On utilise :

- `rownames(tab)` pour connaître le nom des lignes
- `colnames(tab)` pour connaître le nom des colonnes

Et, en bonus, on peut :

- changer les noms des lignes `rownames(tab) <- new1`
- changer les noms des colonnes `colnames(tab) <- new2`

Et, en super bonus, on peut :

- modifier quelques noms de lignes `rownames(tab)[sel1] <- new1`
- modifier quelques noms de colonnes `colnames(tab)[sel1] <- new2`

Modifier un objet ou son contenu

La syntaxe `obj[i] <- newvalue` (et ses variations) peut être utilisée pour tous les types d'objets indexables. Mais il faut l'utiliser avec prudence !

Exemple : `fruits$Energie[1:10] <- 0`

Que s'est-il passé ? **Au secours !!!!**

Pour revenir en arrière : `data("fruits", package = "debuter")`

Extraction avec des noms

Exemple :

- Pour extraire l'énergie : `fruits[, "Energie"]`,
- Pour extraire le groupe : `fruits[, "groupe"]`,
- Pour extraire l'énergie et le groupe : `fruits[, c("Energie", "groupe")]`,
- Pour enlever le groupe : `fruits[, -"groupe"]` ?

Bilan

Mode d'extraction	Exemples
Indices	<code>fruits[, 2]</code>
Booléens	<code>fruits[fruits\$nom == "Abricot",]</code>
Noms	<code>fruits\$nom</code> OU <code>fruits[, "nom"]</code>

Exercice

Lister le maximum de façons possibles d'extraire du tableau `fruits` les fruits crus sucrés riches en Vitamine C !

Construire ses propres objets

Vecteurs et facteurs

- La fonction `c()` permet de combiner des valeurs dans un vecteur. Attention, tout doit être du même "type" !
- La fonction `seq` permet de créer des suites.
- La fonction `rep` permet de créer des vecteurs en répétant des valeurs. Ex:
`rep(c("a", "b"), c(3, 4))`

Les facteurs sont une particularité de R !

- On les crée avec la fonction `factor` ou `as.factor`
- Par exemple : `factor(ruits$groupe)`

Matrices et tableaux

- Les fonctions `matrix`, `rbind` et `cbind` pour créer des matrices. Attention, tout doit être du même “type” !
- Les fonctions `data.frame` ou `as.data.frame` pour créer des

Ajouter des noms

Directement à la création de l'objet. Ex: `x <- c(a = 1, b = 2)`, `d <- data.frame(a = 1:26, b = letters)`

Ou bien après la création de l'objet :

- `names(obj) <- lesNoms` pour un vecteur
- `rownames(obj) <- lesLignes` pour les lignes d'un tableau ou d'une matrice,
- `colnames(obj) <- lesColonnes` pour les colonnes d'un tableau ou d'une matrice.

Exercice

Créez un facteur à partir des groupes de fruits, puis testez la commande suivante :

```
factor(fruits$groupe, levels = c("secs", "compote", "crus", "exotique"))
```

Que se passe-t-il ? Sauvez le résultat dans un objet et faites un diagramme en bâton avec ! Commentez !

Estimation ponctuelle

Définition

Il s'agit d'estimer une caractéristique statistique d'un ensemble de données avec une seule valeur.

Paramètre	Grandeur statistique	Commande
Position	Moyenne	mean
Position	Médiane	median
Position	Minimum	min
Position	Maximum	max
Dispersion	Variance	var
Dispersion	Ecart-type	sd
Dispersion	Intervalle inter-quartiles	IQR
Lien	Covariance	cov
Lien	Corrélation	cor

Rappel : la covariance

Permet de mesurer le degré de co-variation de deux variables :

$$\operatorname{cov}(x, y) = \frac{1}{n-1} \sum_{i=1}^n (x_i - m_x)(y_i - m_y)$$

Rappel : corrélation de Pearson

C'est une covariance normalisée entre -1 et 1 !

$$\rho(x, y) = \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x)} \sqrt{\text{var}(y)}}$$

Rappel : corrélation de Spearman

C'est la corrélation (de Pearson) calculée sur les rangs !

$\rho = \text{cor}(r_x, r_y)$ avec (r_x) le vecteur des rangs de (x) ($\text{rank}(x)$), et (r_y) le vecteur des rangs de (y) ($\text{rank}(y)$).

Rappel (?) : corrélation de Kendall

- Paire concordante : $(x_i < x_j \text{ et } y_i < y_j)$ OU $(x_i > x_j \text{ et } y_i > y_j)$
- Paire discordante : $(x_i < x_j \text{ et } y_i > y_j)$ OU $(x_i > x_j \text{ et } y_i < y_j)$

$\tau = \frac{n_C - n_D}{n_0}$ avec (n_C) le nombre de paires concordantes, (n_D) le nombre de paires discordantes et (n_0) le nombre total de paires de points.

Exercice

Calculez

- la médiane de la teneur en sucres
- la moyenne de la teneur en eau
- l'écart-type de la teneur en eau

Appliquez la fonction `summary` aux données `fruits`.

- Calculez la corrélation de Pearson entre la teneur en eau et la teneur en sucres,
- Calculez la corrélation de Spearman

Les fonctions astucieuses

- `summary` pour obtenir des statistiques
- `str` pour la structure des données
- `table` pour faire des tables de comptage
- `seq_along` pour créer un vecteur d'indices de même longueur qu'un vecteur donné

D'autres fonctions très utiles

- `sum` pour calculer la somme de nombres
- `sort`, `order` et `rank` pour ordonner, et calculer les rangs
- `rowSums` et `colSums` pour calculer les sommes des lignes et colonnes d'une table,
- `rowMeans` et `colMeans` pour calculer les moyennes des lignes et colonnes d'une table,

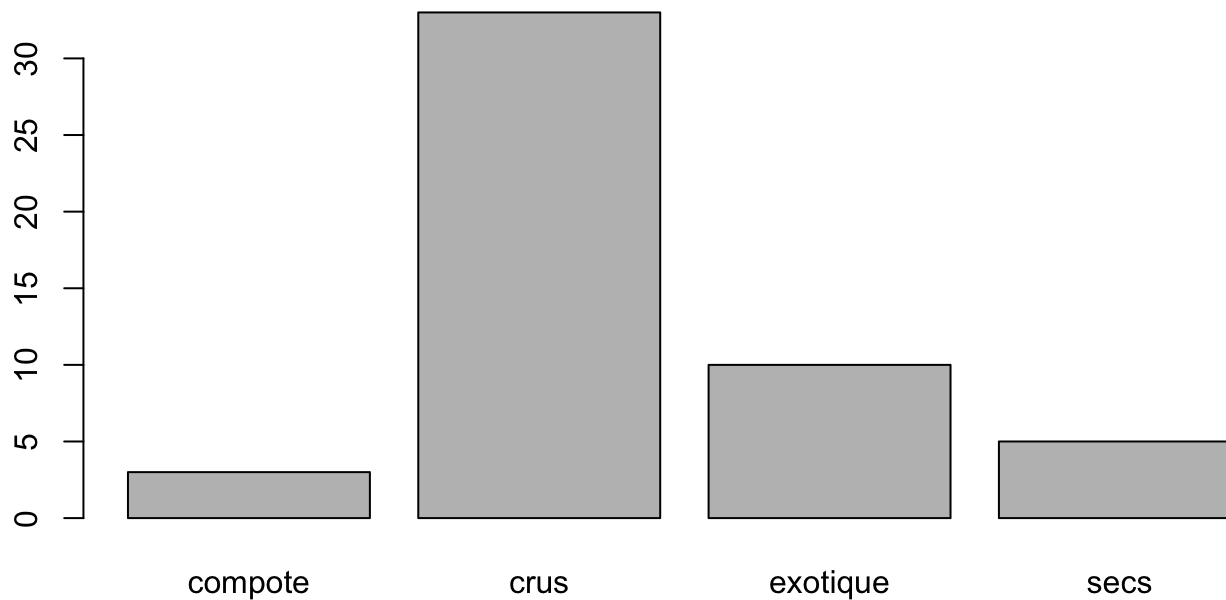
Les graphes de base

- `barplot` : diagrammes en bâtons
- `hist` : histogrammes
- `plot` : nuages de points

La fonction `barplot`

Permet de réaliser des diagrammes en bâtons :

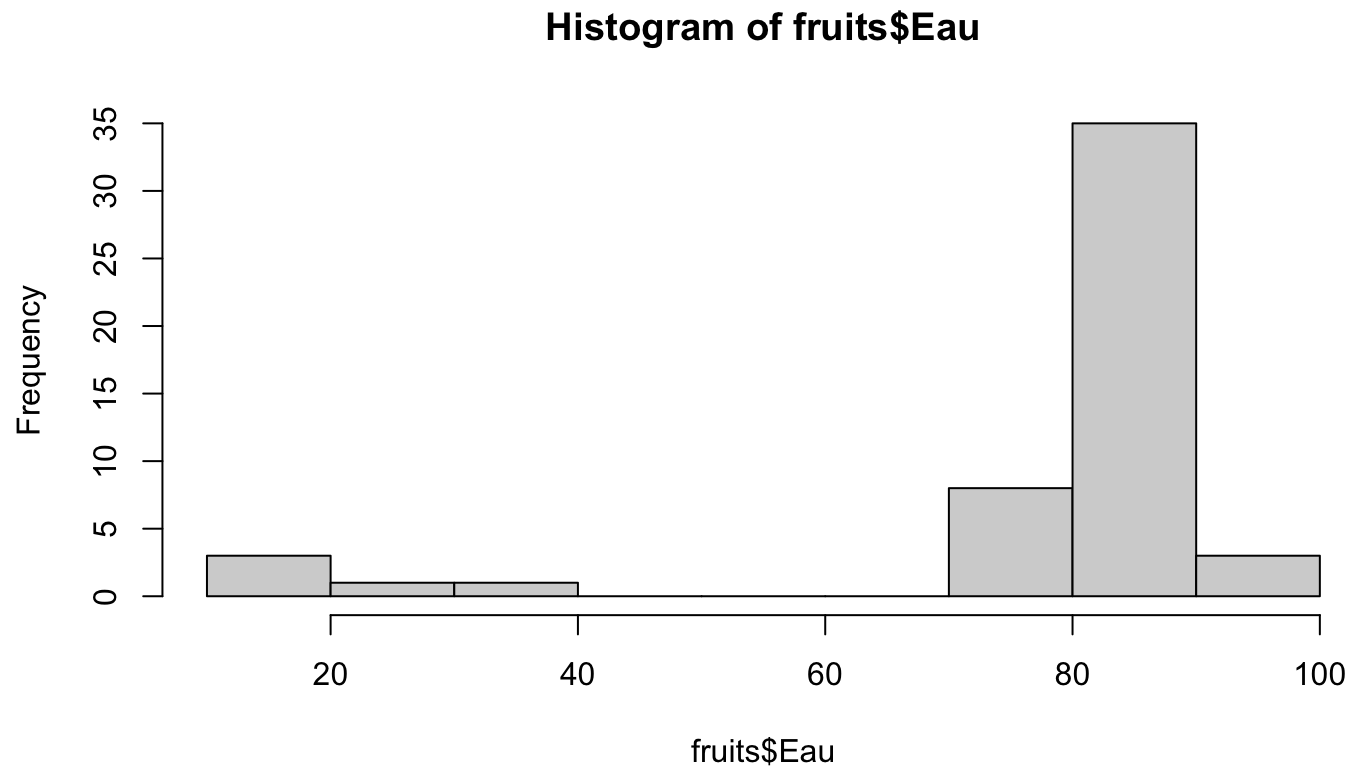
```
barplot(table(fruits$groupe))
```



La fonction `hist`

Permet de réaliser des histogrammes :

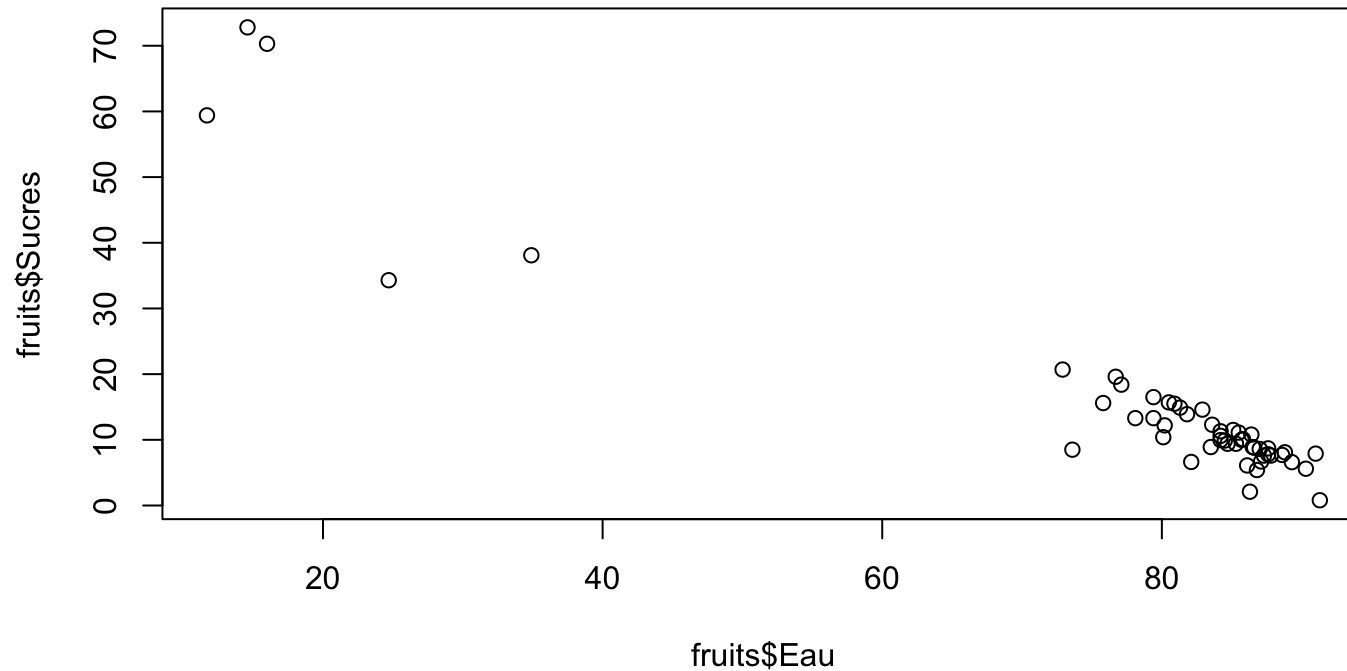
```
hist(fruits$Eau)
```



La fonction `plot`

Permet de tracer des nuages de points :

```
plot(fruits$Eau, fruits$Sucres)
```



Exercice

Faire un histogramme de la teneur en Vitamine C des fruits crus.