Premiers pas avec R

Formation NGS CNRS

Philippe Veber

17 octobre 2023

Présentation

`R` est un système de calcul *open source* dédié aux statistiques. Il comprend :

- un langage
- des bibliothèques (graphiques, inférence statistique, manipulation de données, *etc*)
- un interpréteur (REPL)

L'interpréteur

Depuis le *shell*, on l'invoque avec la commande `R` :

```
$ R

R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
[...]
>
```

- on passe ainsi d'un interpréteur (le terminal) à un autre (notez le changement d'invite).
- pour quitter, `q()` ou `CTRL+D`

R est une calculatrice (en première approximation)

Au lieu d'entrer des commandes (comme dans le terminal), on rentre des **expressions**, c'est-à-dire des formules que l'on souhaite **évaluer**.

Par exemple :

```
42
```

```
## [1] 42
```

```
1 + 1
```

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

```
1 + 5 ** 2
```

```
## [1] 26
```

Notion de fonction

Une **fonction** est un traitement qui prend en entrée des **arguments** et renvoie un **résultat**.

Notation :

$$\underbrace{f}_{\text{nom de la fonction}}(\overbrace{x, y, \ldots}^{\text{arguments}})$$

Exemple d'évaluation :

$$\begin{aligned} 1 + cos(\frac{\pi}{3}) &\rightsquigarrow 1 + \frac{1}{2} \\ &\rightsquigarrow \frac{3}{2} \end{aligned}$$

Les fonctions mathématiques usuelles sont disponibles et nous allons en voir quelques autres.

Obtenir de l'aide

Pour accéder à une page d'aide sur une fonction donnée, on entre son nom précédé de ?

```
?cos
```

On arrive dans un environnement `less`, il faut donc taper `q` pour quitter.

Quand on ne connaît pas le nom de la fonction, il faut utiliser son moteur de recherche préféré sur le web !

Arguments d'une fonction

Tous les arguments d'une fonction ont un nom, que l'on peut utiliser au moment de l'appel. Ces noms permettent notamment d'entrer les arguments dans l'ordre que l'on souhaite

```
seq(from = 2, to = 4, length.out = 5)
seq(length.out = 5, from = 2, to = 4)
```

Les arguments d'une fonction peuvent avoir une valeur par défaut. On peut alors se passer d'en fournir une.

On trouve le nom des arguments et leur éventuelle valeur par défaut en consultant l'aide.

Exercice Déterminez le nom des arguments de la fonction `rnorm` et ceux qui ont une valeur par défaut.

Variables

On peut donner un nom à un résultat intermédiaire en créant une **variable**

```
x <- 41
x + 1
```

```
## [1] 42
```

Exercice Que se passe-t-il si on définit deux fois la même variable ?

Types scalaires

R connaît 4 types de base (dits atomiques)

- *numeric* (flottants en double précision)
- *integer*, notés avec le suffixe `L` pour les distinguer des flottants
- *character* (chaînes de caractères), notées avec des guillemets simples ou doubles
- *logical* (booléens), notés `TRUE` (ou `T`) et `FALSE` (ou `F`)

Exercice La fonction `str` affiche une description de la structure des valeurs. Essayez-la avec différentes expressions

```
str(2)
str(2L)
str("2")
str(TRUE)
```

Conversions automatiques

Si on combine des valeurs de types différents avec un opérateur binaire, R réalise certaines conversions automatiques. Essayez

```
2L + 2
2 + FALSE
2 + "2" # ERREUR !
```

ATTENTION les conversions effectuées ne sont pas forcément celles que vous pouvez souhaiter . . .

Vecteurs

R permet de manipuler des vecteurs de valeurs pour les 4 types de base, par exemple

```
v <- c(1, 2, 3)
str(v)
```

```
##  num [1:3] 1 2 3
```

```
2 * v + 1
```

```
## [1] 3 5 7
```

```
c("A","C","G","T")
```

```
## [1] "A" "C" "G" "T"
```

```
str(c(T, F))
```

```
##  logi [1:2] TRUE FALSE
```

Exercice Déterminer à quoi sert la fonction `length`

Créer des vecteurs

Par énumération/concaténation

```
c(1, 2, 3)
c(c(1, 2, 3), c(4, 5))
```

Par répétition

```
rep(c(1, 2, 3), 3)
```

Via des séquences

```
1:10
seq(from = 0, to = 1, by = 0.1)
seq(from = 0, to = 1, length.out = 10)
```

Opérations sur les vecteurs

Les fonctions sont très souvent **vectorisées** = elles s'appliquent à chaque élément du vecteur.

Il existe aussi des fonctions de **réduction** (`sum`, `mean`, `sd`, *etc*)

Exercice Vérifiez sur quelques exemples la formule

$$\sum_{i=0}^{n} i = 0 + 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

Exercice Calculez une approximation de π en utilisant l'identité

$$\sum_{i=1}^{+\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$$

Vecteurs booléens

On a très souvent recours à des opérateurs de comparaison pour filtrer ses données ou réaliser des comptages. Essayez

```
v <- c(1, 2, 3)
v == 2
v >= 2
v > 2
```

Exercice Calculez le nombre d'entiers entre 1 et 100 dont le logarithme est supérieur à 3.

Opérateurs booléens

On peut combiner les vecteurs booléens à l'aide d'opérateurs logiques pour exprimer des propriétés plus complexes

- conjonction ("et" logique) avec `&`
- disjonction ("ou" logique) avec `|`
- négation avec `!`

Exercice Calculez le nombre d'entiers entre 1 et 100 tels que leur carré est compris entre 100 et 1000

Indexation des vecteurs

On peut accéder à un élément d'un vecteur avec l'opérateur []

```
v <- c(10, 20, 30)
v[2]
```

ATTENTION La numérotation démarre à 1

On peut aussi indexer avec un vecteur

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

ou même filtrer un vecteur

```
v[c(T, F, T)]
```

Indexation des vecteurs

Exercice Soit un vecteur `v` de taille 100. Écrire l'expression calculant

- le vecteur des 10 premiers éléments
- le vecteur des éléments de 10 à 42
- le vecteur des éléments dont l'indice est un multiple de 2 (utiliser l'opérateur modulo %%)
- le vecteur renversé

Listes

On peut construire des collections hétérogènes de valeurs, avec un nom optionnel pour chaque élément

```
l1 <- list(1, 2, "3")
l2 <- list(alphabet = c("A", "C", "G", "T"),
           seq = "AGGTCA",
           GC = 0.5)
```

On peut accéder aux éléments avec `[[]]` ou `$`

```
l1[[2]]
l2$seq
```

Data frame

Lorsqu'une liste ne contient que des vecteurs de même longueur (mais pas forcément de même type), elle constitue un `data.frame`, i.e. une représentation de données tabulées.

On peut en créer avec la fonction `data.frame`

```
df <- data.frame(size = c(24, 21, 27),
                 weight = c(1.2, 1.1, 1.5),
                 genotype = c("WT", "M", "M"))
```

On peut extraire une colonne du tableau avec l'opérateur $, ou extraire un sous-tableau avec l'opérateur `[]`

- `df[3,4]` est le 3e élément de la 4e colonne
- `df[3,]` est la 3e ligne (sous forme de vecteur)
- `df[,4]` est la 4e colonne (sous forme de vecteur)

Chargement d'un tableau

On peut charger des données tabulées depuis un fichier avec la fonction `read.table`.

Les options utiles :

- `sep` pour préciser le séparateur des colonnes (tabulation : `"\t"`)
- `header`, un booléen précisant si la première ligne du fichier donne le nom des colonnes

Exercice Télécharger le fichier de données à `iris` [1], chargez-le dans R et inspectez-le avec `str`. Avec la fonction `summary`, déterminez les statistiques résumées des largeurs de pétales.

[1] https://gist.githubusercontent.com/netj/8836201/raw/6f9306ad21398ea43cba4f7d537619d0e07d5ae3/iris.csv

Génération de nombres aléatoires

Il existe des fonctions pour différentes distributions de probabilité, nommées `r` + nom de la distribution

- `rnorm` pour la gaussienne
- `runif` pour la loi uniforme
- `rbinom` pour la loi binomiale, *etc*

Exercice Générez un échantillon gaussien de taille 30 et calculez sa moyenne. Recommencez et observez les fluctuations.

Nuage de points

On utilise la fonction `plot`

```
N <- 100
x <- seq(0, 1, length=N)
y <- 3 * x + 1 + rnorm(100)
plot(x, y)
plot(x, y,
     pch = 19, col = "blue", main = "Regression",
     xlab = "Length", ylab = "Weight")
```

Exercice Déterminer l'effet des différents arguments de la fonction `plot`

Graphes de fonction

On utilise encore la fonction `plot` en spécifiant l'argument `type` :

```
N <- 100
x <- seq(- 2 * pi, 2 * pi, length=N)
y <- cos(x)
plot(x, y,
     type = "l", col = "red", main = "Cosinus",
     xlab = "x", ylab = "cos(x)")
```

Histogramme

On utilise la fonction `hist`

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

Exercice A quoi sert l'argument `breaks` dans `hist` ?

Exercice Représentez la distribution des tailles de pétales dans le jeu de données `iris`. Recommencez en utilisant uniquement les données pour la variété "Setosa".