Concepts de base en R

Types de données, opérateurs et pré-requis pour la manipulation de données

Mouhamadou Hady Diallo

04 March 2025

Section 1

Types de données en R

Vecteurs atomiques

Les vecteurs sont les structures de données les plus fondamentales en R.

```r
# Vecteur numérique (numeric)
x_num <- c(1, 2, 3, 4, 5)
print(x_num)
```

```
#> [1] 1 2 3 4 5
```

```r
typeof(x_num)
```

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

```r
# Vecteur entier (integer)
x_int <- c(1L, 2L, 3L, 4L, 5L)  # Le L désigne un entier
print(x_int)
```

```
#> [1] 1 2 3 4 5
```

```r
typeof(x_int)
```

Vecteurs atomiques (suite)

```r
# Vecteur caractère (character)
x_char <- c("a", "b", "c", "d")
print(x_char)
```

```
#> [1] "a" "b" "c" "d"
```

```r
typeof(x_char)
```

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

```r
# Vecteur logique (logical)
x_log <- c(TRUE, FALSE, TRUE, FALSE)
print(x_log)
```

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

```r
typeof(x_log)
```

Vecteurs atomiques (suite)

```r
# Vecteur complexe (complex)
x_complex <- c(1+2i, 3+4i)
print(x_complex)
```

```
#> [1] 1+2i 3+4i
```

```r
typeof(x_complex)
```

```
#> [1] "complex"
```

```r
# Vecteur raw (binaire)
x_raw <- raw(4)
print(x_raw)
```

```
#> [1] 00 00 00 00
```

```r
typeof(x_raw)
```

Facteurs

Les facteurs sont utilisés pour représenter des données catégorielles.

```r
# Facteur (factor)
x_fact <- factor(c("homme", "femme", "homme", "femme", "homme"
print(x_fact)
```

```
#> [1] homme femme homme femme homme
#> Levels: femme homme
```

```r
levels(x_fact)  # Modalités du facteur
```

```
#> [1] "femme" "homme"
```

```r
# Facteur ordonné
x_ord <- factor(c("petit", "moyen", "grand", "petit"),
                levels = c("petit", "moyen", "grand"),
                ordered = TRUE)
print(x_ord)
```

Listes

Les listes peuvent contenir des éléments de types différents.

```
# Liste
ma_liste <- list(
  nombre = 42,
  texte = "Bonjour",
  logique = TRUE,
  vecteur = c(1, 2, 3)
)
print(ma_liste)
```

```
#> $nombre
#> [1] 42
#>
#> $texte
#> [1] "Bonjour"
#>
#> $logique
```

Data Frames

Les data frames sont les structures de données les plus utilisées pour l'analyse de données.

```r
# Data frame
mon_df <- data.frame(
  id = 1:3,
  nom = c("Alice", "Bob", "Charlie"),
  age = c(25, 30, 35),
  actif = c(TRUE, FALSE, TRUE)
)
print(mon_df)
```

```
#>   id      nom age actif
#> 1  1    Alice  25  TRUE
#> 2  2      Bob  30 FALSE
#> 3  3 Charlie  35  TRUE
```

Matrices

Les matrices sont des tableaux à deux dimensions avec des éléments de même type.

```r
# Matrice
ma_matrice <- matrix(1:9, nrow = 3, ncol = 3)
print(ma_matrice)
```

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

```r
# Dimensions
dim(ma_matrice)
```

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

Arrays

Les arrays sont des tableaux multidimensionnels.

```r
# Array (tableau à 3 dimensions)
mon_array <- array(1:24, dim = c(2, 3, 4))
print(mon_array)
```

```
#> , , 1
#>
#>      [,1] [,2] [,3]
#> [1,]    1    3    5
#> [2,]    2    4    6
#>
#> , , 2
#>
#>      [,1] [,2] [,3]
#> [1,]    7    9   11
#> [2,]    8   10   12
#>
```

Vérification du type de données

```r
# Vérifier le type d'objet
is.numeric(x_num)
```

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

```r
is.character(x_char)
```

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

```r
is.data.frame(mon_df)
```

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

```r
is.matrix(ma_matrice)
```

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

```r
# Vérifier la classe
class(x_num)
```

Section 2

Opérateurs en R

Opérateurs arithmétiques

```r
# Addition
5 + 3
```

```
#> [1] 8
```

```r
# Soustraction
5 - 3
```

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

```r
# Multiplication
5 * 3
```

```
#> [1] 15
```

```r
# Division
5 / 3
```

```
#> [1] 1.666667
```

Opérateurs arithmétiques (suite)

```
# Puissance
2^3
```

```
#> [1] 8
```

```
# Opérations vectorielles
c(1, 2, 3) + c(4, 5, 6)
```

```
#> [1] 5 7 9
```

```
c(1, 2, 3) * c(4, 5, 6)
```

```
#> [1]  4 10 18
```

Opérateurs de comparaison

```r
# Égalité
5 == 5
```

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

```r
# Différence
5 != 3
```

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

```r
# Supérieur et inférieur
5 > 3
```

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

```r
5 < 3
```

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

```r
# Supérieur ou égal, inférieur ou égal
```

Opérateurs logiques

```r
# ET logique
TRUE & FALSE
```

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

```r
# OU logique
TRUE | FALSE
```

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

```r
# NON logique
!TRUE
```

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

```r
# ET vectoriel
c(TRUE, FALSE, TRUE) & c(TRUE, TRUE, FALSE)
```

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

Opérateurs logiques (suite)

```
# ET avec évaluation court-circuit
TRUE && FALSE
```

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

```
# OU avec évaluation court-circuit
TRUE || FALSE
```

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

```
# Note: && et || n'opèrent que sur le premier élément des vec
```

Opérateurs d'assignation

```r
# Assignation classique
x <- 10
print(x)
```

```
#> [1] 10
```

```r
# Assignation alternative
y = 20
print(y)
```

```
#> [1] 20
```

```r
# Assignation de droite à gauche
30 -> z
print(z)
```

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

Opérateurs spéciaux

```r
# Opérateur pipe (%>%) du package magrittr
library(magrittr)
1:10 %>% mean() %>% round(2)
```

```
#> [1] 5.5
```

```r
# Opérateur natif (|>)
1:10 |> mean() |> round(2)
```

```
#> [1] 5.5
```

Section 3

Manipulation de base des données

Création de séquences

```r
# Séquence régulière
seq(1, 10, by = 2)
```

```
#> [1] 1 3 5 7 9
```

```r
# Séquence de longueur spécifique
seq(0, 1, length.out = 5)
```

```
#> [1] 0.00 0.25 0.50 0.75 1.00
```

```r
# Répétition
rep(1:3, times = 2)
```

```
#> [1] 1 2 3 1 2 3
```

```r
rep(1:3, each = 2)
```

```
#> [1] 1 1 2 2 3 3
```

Indexation des vecteurs

```r
# Vecteur exemple
x <- c(10, 20, 30, 40, 50)

# Indexation par position
x[1]          # Premier élément
```

```
#> [1] 10
```

```r
x[c(1, 3)]   # Premier et troisième éléments
```

```
#> [1] 10 30
```

```r
# Indexation négative (exclusion)
x[-2]         # Tous sauf le deuxième
```

```
#> [1] 10 30 40 50
```

```r
# Indexation logique
x[x > 30]     # Éléments supérieurs à 30
```

Indexation des data frames

```r
# Exemple de data frame
df <- data.frame(
  id = 1:5,
  valeur = c(10, 20, 30, 40, 50),
  groupe = c("A", "B", "A", "B", "A")
)

# Sélection de colonnes
df$valeur
```

```
#> [1] 10 20 30 40 50
```

```r
df[, "valeur"]
```

```
#> [1] 10 20 30 40 50
```

```r
df[, 2]
```

Indexation avancée

```r
# Avec dplyr
library(dplyr)

df %>%
  select(id, valeur) %>%
  filter(valeur > 20)
```

```
#>   id valeur
#> 1  3     30
#> 2  4     40
#> 3  5     50
```

```r
# Modification conditionnelle
df %>%
  mutate(categorie = if_else(valeur > 30, "Élevé", "Faible"))
```

```
#>   id valeur groupe categorie
#> 1  1     10      A    Faible
```

Section 4

Fonctions utiles

Fonctions statistiques de base

```r
# Vecteur exemple
donnees <- c(12, 15, 21, 18, 24, 19, 22)

# Statistiques descriptives
mean(donnees)    # Moyenne
```

```
#> [1] 18.71429
```

```r
median(donnees)  # Médiane
```

```
#> [1] 19
```

```r
sd(donnees)      # Écart-type
```

```
#> [1] 4.151879
```

```r
var(donnees)     # Variance
```

Fonctions d'importation de données

```r
# Importation CSV
data_csv <- read.csv("donnees.csv")

# Importation Excel (avec readxl)
library(readxl)
data_excel <- read_excel("donnees.xlsx")

# Importation SPSS, SAS, Stata (avec haven)
library(haven)
data_spss <- read_spss("donnees.sav")
data_sas <- read_sas("donnees.sas7bdat")
data_stata <- read_stata("donnees.dta")
```

Fonctions d'inspection des données

```r
# Aperçu des premières lignes
head(iris, 3)
```

```
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1          5.1         3.5          1.4         0.2  setosa
#> 2          4.9         3.0          1.4         0.2  setosa
#> 3          4.7         3.2          1.3         0.2  setosa
```

```r
# Aperçu des dernières lignes
tail(iris, 3)
```

```
#>     Sepal.Length Sepal.Width Petal.Length Petal.Width   Spe
#> 148          6.5         3.0          5.2         2.0 virgi
#> 149          6.2         3.4          5.4         2.3 virgi
#> 150          5.9         3.0          5.1         1.8 virgi
```

```r
# Structure des données
str(iris)
```

Gestion des valeurs manquantes

```r
# Créer des valeurs manquantes
x <- c(1, 2, NA, 4, 5, NA)

# Détecter les valeurs manquantes
is.na(x)
```

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

```r
# Compter les valeurs manquantes
sum(is.na(x))
```

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

```r
# Filtrer les valeurs manquantes
x[!is.na(x)]
```

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

```r
# Remplacer les valeurs manquantes
```

Section 5

Conclusion

Ressources supplémentaires

- Documentation officielle de R: www.r-project.org
- RStudio Cheatsheets: rstudio.com/resources/cheatsheets
- Livre R for Data Science: r4ds.had.co.nz
- Communauté Stack Overflow: stackoverflow.com/questions/tagged/r
- R-bloggers: r-bloggers.com

Merci!

Questions?