Introduction à R

INESSS, BDCA

2025-02-10

Les bases

Arithmétique (1)

```
# Addition
5 + 5
```

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

```
# Soustraction
5 - 5
```

```
#> [1] 0
```

```
# Multiplication
3 * 5
```

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

```
# Division
(5 + 5) / 2
```

```
#> [1] 5
```

Arithmétique (2)

```
# Exposant
2^5
```

```
#> [1] 32
```

```
# Modulo
28 %% 6
```

```
#> [1] 4
```

Prendre note que le caractère # met la ligne de code en commentaire et n'est pas évalué par R.

Affectation de variables (1)

Une variable permet de stocker une valeur ou un objet. On peut ensuite l'utiliser ultérieurement pour accéder facilement à cette valeur.

```
# Assigner la valeur 42 à x
x <- 42

# Afficher/Imprimer la valeur de la variable x
x
```

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

```
print(x)
```

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

Affectation de variables (2)

Nous pouvons appliquer des opérations arithmétiques entre des variables numériques.

```
x <- 5
y <- 25
x + y
```

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

```
x - y
```

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

```
y / x
```

```
#> [1] 5
```

Type de données de base (1)

- Nombres entiers : `4` (`integer`/`int`)

```
integer <- 4
```

- Numérique : `4.5` (`numeric`/`num`)

```
numeric <- 4.5
```

- Logique ou booléen : `TRUE` ou `FALSE` (`logical`/`logi`)

```
logique1 <- TRUE
logique2 <- FALSE
```

- Texte : `Chaîne de caractères` (`character`/`chr`)

```
character1 <- "Chaine de caractères"
character2 <- 'Chaine de caractères'
```

Type de données de base (2)

R peut décider que c'est numérique même si on veut des nombres entiers. Ajouter `L` après un nombre force la classe d'un nombre à entier au lieu de numérique

```
class(5)
```

```
#> [1] "numeric"
```

```
class(5L)
```

```
#> [1] "integer"
```

```
class(1:5)
```

```
#> [1] "integer"
```

Vecteurs

Créer un vecteur

On crée un vecteur avec la fonction `c()`.

```
numeric_vec <- c(1, 10, 49)
character_vec <- c("a", "b", "c")
boolean_vec <- c(TRUE, FALSE, T, F)
```

Mise en situation

Après une semaine à Las Vegas, on décide de prendre en note nos gains et nos pertes :

```
# Poker
poker <- c(140, -50, 20, -120, 240)

# Roulette
roulette <- c(-24, 50, 100, -350, 10)
```

Nommer un vecteur (1)

Nous pouvons donner un nom aux éléments d'un vecteur avec la fonction `names()`.

```
# Poker - Gains/Pertes
poker <- c(140, -50, 20, -120, 240)

# Roulette - Gains/Pertes
roulette <- c(-24, 50, 100, -350, 10)

# Poker - Jour de la semaine
names(poker) <-
  c("Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi")

# Roulette - Jour de la semaine
names(roulette) <- c(
  "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi"
)

print(poker)
```

```
#>    Lundi    Mardi Mercredi    Jeudi Vendredi
#>      140      -50       20     -120      240
```

Nommer un vecteur (2)

Si une information est utilisé plus d'une fois, il est utile de créer une nouvelle variable.

```r
# Poker - Gains/Pertes
poker <- c(140, -50, 20, -120, 240)

# Roulette - Gains/Pertes
roulette <- c(-24, 50, 100, -350, 10)

# Jour de la semaine
jours <- c("Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi")

# Attribuer le jour de la semaine
names(poker) <- jours
names(roulette) <- jours

print(poker)
```

```
#>    Lundi    Mardi Mercredi    Jeudi Vendredi
#>      140      -50       20     -120      240
```

Calculs sur un vecteur (1)

Pour chaque jour, calculons les gains et les pertes du poker et de la roulette.

```
poker
```

```
#>    Lundi    Mardi Mercredi    Jeudi Vendredi
#>      140      -50       20     -120      240
```

```
roulette
```

```
#>    Lundi    Mardi Mercredi    Jeudi Vendredi
#>      -24       50      100     -350       10
```

```
total_jour <- poker + roulette
total_jour
```

```
#>    Lundi    Mardi Mercredi    Jeudi Vendredi
#>      116        0      120     -470      250
```

Calculs sur un vecteur (2)

Calculons les totaux de la semaine

```
total_poker <- sum(poker)
total_poker
```

```
#> [1] 230
```

```
total_roulette <- sum(roulette)
total_roulette
```

```
#> [1] -214
```

```
total_semaine <- sum(total_jour)
total_semaine
```

```
#> [1] 16
```

Comparer les gains

Vérifions si nos gains sont meilleurs au poker qu'à la roulette

```
total_poker
```

```
#> [1] 230
```

```
total_roulette
```

```
#> [1] -214
```

```
total_poker > total_roulette
```

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

Sélection dans un vecteur (1)

Pour sélectionner une valeur dans un vecteur, nous utilisons les crochets `[]`.

```
poker
```

```
#>    Lundi    Mardi Mercredi    Jeudi Vendredi
#>      140      -50       20     -120      240
```

```
# Gains du mercredi
poker[3]
```

```
#> Mercredi
#>       20
```

```
poker[[3]]
```

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

Sélection dans un vecteur (2)

Il est possible de sélectionner plusieurs valeurs à l'aide d'un vecteur. Si nous voulons analyser les gains du lundi et du vendredi, nous pourrions utiliser le vecteur `c(1, 5)` à l'intérieur des crochets : `poker[c(1, 5)]`.
Analysons les gains du mardi au jeudi :

```
poker[c(2, 3, 4)]
```

```
#>    Mardi Mercredi    Jeudi
#>      -50        20     -120
```

```
poker[c(2:4)]
```

```
#>    Mardi Mercredi    Jeudi
#>      -50        20     -120
```

```
poker[2:4]
```

```
#>    Mardi Mercredi    Jeudi
#>      -50        20     -120
```

Sélection dans un vecteur (3)

Une autre manière de sélectionner des valeurs est d'utiliser les noms lorsqu'il y en a.

```
poker
```

```
#>    Lundi    Mardi Mercredi    Jeudi Vendredi
#>      140      -50       20     -120      240
```

```
poker[c("Mardi", "Jeudi")]
```

```
#> Mardi Jeudi
#>   -50  -120
```

```
poker["Mercredi"]
```

```
#> Mercredi
#>       20
```

Sélection par comparaison (1)

Il est aussi possible de sélectionner des valeurs par comparaison.

```
# Jours où les gains sont positifs
poker > 0
```

```
#>    Lundi    Mardi Mercredi    Jeudi Vendredi
#>     TRUE    FALSE     TRUE    FALSE     TRUE
```

```
poker[poker > 0]
```

```
#>    Lundi Mercredi Vendredi
#>      140       20      240
```

```
selection <- poker > 0
poker[selection]
```

```
#>    Lundi Mercredi Vendredi
#>      140       20      240
```

Sélection par comparaison (2)

Liste des opérateurs de comparaison :

- `<` : plus petit
- `>` : plus grand
- `<=` : plus petit ou égal
- `>=` : plus grand ou égal
- `==` : égal
- `!=` : pas égal, différent

Les *data frames*

Qu'est-ce qu'un *data frame* ?

Un *data frame* est tout simplement un tableau comme on peut en avoir avec Excel.
Chaque colonne est une variable d'un même type (numérique, caractère, logique...) et chaque ligne est une observation.

```
head(mtcars)
```

```
#>                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
#> Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
#> Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
#> Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
#> Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
#> Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
#> Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Les fonctions `head()` et `tail()` permettent de voir les premières ou les dernières observations d'un ensemble de données.

Structure des données

La fonction `str()` est semblable à la fonction `class()`, mais affiche plus d'informations.

```
class(mtcars)
```

```
#> [1] "data.frame"
```

```
str(mtcars)
```

```
#> 'data.frame':    32 obs. of  11 variables:
#>  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
#>  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
#>  $ disp: num  160 160 108 258 360 ...
#>  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
#>  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
#>  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
#>  $ qsec: num  16.5 17 18.6 19.4 17 ...
#>  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
#>  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
#>  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
#>  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

`mtcars` est un `data.frame` de 32 observations et 11 colonnes, toutes les variables sont de type numériques (`num`) et les dix premières observations de chaque colonne sont affichées.

Créer un data.frame (1)

La fonction `data.frame()` permet de créer un `data.frame`. Chaque colonne est un vecteur et tous les vecteurs doivent contenir le même nombre de valeurs.

```
personne <- c("Jean Airoldi", "Ricardo", "George Lucas",
              "FredK Le Roi", "Dre Samson")
main <- c("droitier", "gaucher", "ambidextre", "gaucher",
          "droitier")
gentillesse_sur_10 <- c(9, 2, 0, 10, 20)
id <- 1:5
on_laime <- c(FALSE, TRUE, FALSE, TRUE, TRUE)
```

Créer un data.frame (2)

Chaque vecteur doivent contenir le même nombre de valeurs.
Les colonnes sont dans le même ordre que les vecteurs inscrits dans la fonction `data.frame()`.

```
mon_data <- data.frame(id, personne, main, gentillesse_sur_10, on_laime)
mon_data
```

```
#>   id     personne       main gentillesse_sur_10 on_laime
#> 1  1 Jean Airoldi   droitier                  9    FALSE
#> 2  2      Ricardo    gaucher                  2     TRUE
#> 3  3 George Lucas ambidextre                  0    FALSE
#> 4  4 FredK Le Roi    gaucher                 10     TRUE
#> 5  5   Dre Samson   droitier                 20     TRUE
```

```
str(mon_data)
```

```
#> 'data.frame':    5 obs. of  5 variables:
#>  $ id                : int  1 2 3 4 5
#>  $ personne          : chr  "Jean Airoldi" "Ricardo" "George Lucas" "FredK Le
#>  $ main              : chr  "droitier" "gaucher" "ambidextre" "gaucher" ...
#>  $ gentillesse_sur_10: num  9 2 0 10 20
#>  $ on_laime          : logi  FALSE TRUE FALSE TRUE TRUE
```

Sélection des éléments d'un *data.frame* (1)

Comme pour les vecteurs, on sélectionne les éléments d'un tableau avec les crochets `[]`. À l'aide d'une virgule, on peut décider entre les lignes et les colonnes à afficher :

- `mon_data[1, 2]` sélectionne la première ligne et la deuxième colonne
- `mon_data[1:3, 2:4]` sélectionne les trois premières lignes et les colonnes 2, 3 et 4.

Sélection des éléments d'un *data.frame* (2)

```
mon_data
```

```
#>   id     personne       main gentillesse_sur_10 on_laime
#> 1  1 Jean Airoldi   droitier                  9    FALSE
#> 2  2      Ricardo    gaucher                  2     TRUE
#> 3  3 George Lucas ambidextre                  0    FALSE
#> 4  4 FredK Le Roi    gaucher                 10     TRUE
#> 5  5   Dre Samson   droitier                 20     TRUE
```

```
mon_data[1, 2]
```

```
#> [1] "Jean Airoldi"
```

```
mon_data[1:3, 2:4]
```

```
#>       personne       main gentillesse_sur_10
#> 1 Jean Airoldi   droitier                  9
#> 2      Ricardo    gaucher                  2
#> 3 George Lucas ambidextre                  0
```

Sélection des éléments d'un *data.frame* (3)

On peut utiliser le nom des colonnes au lieu les nombres

```
mon_data
```

```
#>   id     personne       main gentillesse_sur_10 on_laime
#> 1  1 Jean Airoldi   droitier                 9    FALSE
#> 2  2      Ricardo    gaucher                 2     TRUE
#> 3  3 George Lucas ambidextre                 0    FALSE
#> 4  4 FredK Le Roi    gaucher                10     TRUE
#> 5  5   Dre Samson   droitier                20     TRUE
```

```
mon_data[2, "gentillesse_sur_10"]
```

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

```
mon_data[c(1, 4, 5), c("personne", "main")]
```

```
#>       personne     main
#> 1 Jean Airoldi droitier
#> 4 FredK Le Roi  gaucher
#> 5   Dre Samson droitier
```

Sélection d'une colonne uniquement

Indiquer uniquement la ou les colonnes après la virgule retournera toutes les lignes.
En plus des crochets, l'utilisation du symbole `$` est aussi possible.

```
mon_data[, 2]
```

```
#> [1] "Jean Airoldi" "Ricardo"      "George Lucas" "FredK Le Roi" "Dre Samson"
```

```
mon_data[, "personne"]
```

```
#> [1] "Jean Airoldi" "Ricardo"      "George Lucas" "FredK Le Roi" "Dre Samson"
```

```
mon_data$personne
```

```
#> [1] "Jean Airoldi" "Ricardo"      "George Lucas" "FredK Le Roi" "Dre Samson"
```

Sélection d'observation selon condition (1)

À l'aide d'un vecteur contenant des `TRUE` ou des `FALSE`, on peut sélectionner les données des gens qu'on aime.

```
mon_data[, c("personne", "on_laime")]
```

```
#>       personne on_laime
#> 1 Jean Airoldi    FALSE
#> 2      Ricardo     TRUE
#> 3 George Lucas    FALSE
#> 4 FredK Le Roi     TRUE
#> 5   Dre Samson     TRUE
```

```
mon_data$on_laime
```

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

```
mon_data[c(FALSE, TRUE, FALSE, TRUE, TRUE),]  # la virgule est importante
```

```
#>   id     personne     main gentillesse_sur_10 on_laime
#> 2  2      Ricardo  gaucher                 2     TRUE
#> 4  4 FredK Le Roi  gaucher                10     TRUE
#> 5  5   Dre Samson droitier                20     TRUE
```

Sélection d'observation selon condition (2)

Même chose que précédemment, mais en utilisant le vecteur `on_laime`

```
on_laime <- mon_data$on_laime
mon_data[on_laime,]  # avec vecteur créé
```

```
#>   id     personne     main gentillesse_sur_10 on_laime
#> 2  2      Ricardo  gaucher                 2     TRUE
#> 4  4 FredK Le Roi  gaucher                10     TRUE
#> 5  5   Dre Samson droitier                20     TRUE
```

```
mon_data[mon_data$on_laime,]  # avec colonne du data
```

```
#>   id     personne     main gentillesse_sur_10 on_laime
#> 2  2      Ricardo  gaucher                 2     TRUE
#> 4  4 FredK Le Roi  gaucher                10     TRUE
#> 5  5   Dre Samson droitier                20     TRUE
```

Sélection d'observation selon condition (3)

Cherchons les gens qui ont une gentillesse sur 10 d'au moins 5.

```
mon_data$gentillesse_sur_10 > 5
```

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

```
mon_data[mon_data$gentillesse_sur_10 > 5,]
```

```
#>   id     personne     main gentillesse_sur_10 on_laime
#> 1  1 Jean Airoldi droitier                 9    FALSE
#> 4  4 FredK Le Roi  gaucher                10     TRUE
#> 5  5   Dre Samson droitier                20     TRUE
```

```
gentil5 <- mon_data$gentillesse_sur_10 > 5
mon_data[gentil5,]
```

```
#>   id     personne     main gentillesse_sur_10 on_laime
#> 1  1 Jean Airoldi droitier                 9    FALSE
#> 4  4 FredK Le Roi  gaucher                10     TRUE
#> 5  5   Dre Samson droitier                20     TRUE
```

Trier son *data frame*

La fonction `order()` indique le rang, la position de la donnée. On peut ensuite l'utiliser pour afficher les lignes dans l'ordre voulu.

```
mon_data
```

```
#>   id     personne       main gentillesse_sur_10 on_laime
#> 1  1 Jean Airoldi   droitier                 9    FALSE
#> 2  2      Ricardo    gaucher                 2     TRUE
#> 3  3 George Lucas ambidextre                 0    FALSE
#> 4  4 FredK Le Roi    gaucher                10     TRUE
#> 5  5   Dre Samson   droitier                20     TRUE
```

```
tri_gentil <- order(mon_data$gentillesse_sur_10)
tri_gentil
```

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

```
mon_data[tri_gentil,]
```

```
#>   id     personne       main gentillesse_sur_10 on_laime
#> 3  3 George Lucas ambidextre                 0    FALSE
#> 2  2      Ricardo    gaucher                 2     TRUE
#> 1  1 Jean Airoldi   droitier                 9    FALSE
#> 4  4 FredK Le Roi    gaucher                10     TRUE
#> 5  5   Dre Samson   droitier                20     TRUE
```

Matrices

Qu'est-ce qu'une matrice

Une matrice est semblable à un `data.frame`, mais toutes les colonnes sont du même type (numérique, caractère ou logique).
Une matrice est construite à partir de la fonction `matrix()`.

```
matrix(1:9, nrow = 3, byrow = TRUE)
```

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

```
matrix(101:110, ncol = 5, byrow = FALSE)
```

```
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]  101  103  105  107  109
#> [2,]  102  104  106  108  110
```

Nommer une matrice (ou un *data frame*)

Les fonctions `rownames()` et `colnames()` permettent de nommer les lignes et les colonnes d'une matrice ou d'un tableau.

```
ma_matrice <- matrix(1:9, nrow = 3, byrow = TRUE)
colnames(ma_matrice) <- c(LETTERS[1:3])
rownames(ma_matrice) <- c(letters[24:26])
ma_matrice
```

```
#>   A B C
#> x 1 2 3
#> y 4 5 6
#> z 7 8 9
```

Calcul sur une matrice

Les fonctions `rowSums` et `colSums` permettent de calculer la somme d'une ligne et d'une colonne.

```
ma_matrice
```

```
#>   A B C
#> x 1 2 3
#> y 4 5 6
#> z 7 8 9
```

```
rowSums(ma_matrice)
```

```
#>  x  y  z
#>  6 15 24
```

```
colSums(ma_matrice)
```

```
#>  A  B  C
#> 12 15 18
```

Ajouter une ligne (ou colonne)

La fonction `rbind()` permet d'ajouter des lignes et `cbind()` des colonnes.

```
ma_matrice
```

```
#>   A B C
#> x 1 2 3
#> y 4 5 6
#> z 7 8 9
```

```
rowSums(ma_matrice)
```

```
#>  x  y  z
#>  6 15 24
```

```
rbind(ma_matrice, rowSums(ma_matrice))
```

```
#>   A  B  C
#> x 1  2  3
#> y 4  5  6
#> z 7  8  9
#>   6 15 24
```

Opérations sur une matrice (1)

```
ma_matrice
```

```
#>   A B C
#> x 1 2 3
#> y 4 5 6
#> z 7 8 9
```

```
mean(ma_matrice[1,])  # de la première ligne
```

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

```
mean(ma_matrice[, 1])  # de la première colonne
```

```
#> [1] 4
```

```
mean(ma_matrice)  # de toute la matrice
```

```
#> [1] 5
```

Opération sur une matrice (2)

```
ma_matrice
```

```
#>   A B C
#> x 1 2 3
#> y 4 5 6
#> z 7 8 9
```

```
ma_matrice * 3
```

```
#>    A  B  C
#> x  3  6  9
#> y 12 15 18
#> z 21 24 27
```

```
ma_matrice[, 1] * 3
```

```
#>  x  y  z
#>  3 12 21
```

Facteurs

Qu'est-ce qu'un facteur? (1)

Le terme facteur est un type de données pour stocker des variables catégoriques et potentiellement les trier.

```
sex <- c("Femme", "Homme", "Homme", "Femme", "Femme")
factor_sex <- factor(sex)
factor_sex
```

```
#> [1] Femme Homme Homme Femme Femme
#> Levels: Femme Homme
```

Qu'est-ce qu'un facteur (2)

Comment trier une variables catégorique :

```
temperature <- c("Élevée", "Faible", "Élevée", "Faible", "Moyenne")
factor_temperature <- factor(
  temperature,
  order = TRUE,
  levels = c("Faible", "Moyenne", "Élevée")
)
factor_temperature
```

```
#> [1] Élevée  Faible  Élevée  Faible  Moyenne
#> Levels: Faible < Moyenne < Élevée
```

Très utile avec les légendes de graphiques.

Liste

Qu'est-ce qu'une liste?

Une liste dans R permet de réunir une variété d'objets sous une même variable. Peut contenir des matrices, des vecteurs, des *data.frame* et même d'autres listes.

Comment créer une liste (1)

```
vecteur <- 1:10
matrice <- matrix(1:9, ncol = 3)
datafr <- mtcars[1:3, 1:3]
ma_liste <- list(vecteur, matrice, datafr)
ma_liste
```

```
#> [[1]]
#>  [1]  1  2  3  4  5  6  7  8  9 10
#>
#> [[2]]
#>      [,1] [,2] [,3]
#> [1,]    1    4    7
#> [2,]    2    5    8
#> [3,]    3    6    9
#>
#> [[3]]
#>                mpg cyl disp
#> Mazda RX4     21.0   6  160
#> Mazda RX4 Wag 21.0   6  160
#> Datsun 710    22.8   4  108
```

Comment créer une liste (2)

Nommer directement les éléments d'une liste

```
ma_liste <- list(vec = vecteur, mat = matrice, df = datafr)
ma_liste
```

```
#> $vec
#>  [1]  1  2  3  4  5  6  7  8  9 10
#>
#> $mat
#>      [,1] [,2] [,3]
#> [1,]    1    4    7
#> [2,]    2    5    8
#> [3,]    3    6    9
#>
#> $df
#>                mpg cyl disp
#> Mazda RX4     21.0   6  160
#> Mazda RX4 Wag 21.0   6  160
#> Datsun 710    22.8   4  108
```

Comment créer une liste (3)

Nommer les éléments d'une liste après l'avoir créée

```
names(ma_liste) <- c("MonVecteur", "MaMatrice", "MonDf")
ma_liste
```

```
#> $MonVecteur
#>  [1]  1  2  3  4  5  6  7  8  9 10
#>
#> $MaMatrice
#>      [,1] [,2] [,3]
#> [1,]    1    4    7
#> [2,]    2    5    8
#> [3,]    3    6    9
#>
#> $MonDf
#>                mpg cyl disp
#> Mazda RX4     21.0   6  160
#> Mazda RX4 Wag 21.0   6  160
#> Datsun 710    22.8   4  108
```

Sélectionner les éléments d'une liste (1)

Semblable à la sélection d'un vecteur ou d'un *data frame* avec une particularité au niveau des crochets `[]`

```
ma_liste[1]
```

```
#> $MonVecteur
#>  [1]  1  2  3  4  5  6  7  8  9 10
```

```
ma_liste[[1]]
```

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

```
ma_liste[["MonVecteur"]]
```

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

```
ma_liste$MonVecteur
```

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

Sélectionner les éléments d'une liste (2)

Si ce n'est pas un vecteur, l'utilisation du double crochet n'a pas d'impact

```
ma_liste[2]
```

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

```
ma_liste[[2]]
```

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

BRAVO!

Vous êtes maintenant de jeunes padawans Rinessiens