Les bases du Langage R

Ibrahima Sy

Institut Supérieure de Finance

10/8/2021

Intoducion

Les objets

Les Vecteurs

Matrices et tableaux

Les Listes

Data Frames

Indexation

Opérateurs arithmétiques

Intoducion

Introduction

- Pour utiliser un langage de programmation, il faut en connaître la syntaxe et la sémantique, du moins dans leurs grandes lignes.
- C'est dans cet esprit que ce chapitre introduit des notions de base du langage R telles que **l'expression, l'affectation, l'objet, les opérateurs, les fonctions**, etc.

Les commnades R

- l'utilisateur de R interagit avec l'interprète R en entrant des commandes à l'invite de commande.
- Toute commande R est soit une expression, soit une affectation.
- Normalement, une expression est immédiatement évaluée et le résultat est affiché à l'écran :

```
2+3
```

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

```
pi
```

```
## [1] 3.141593
```

```
cos(pi/4)
```

```
## [1] 0.7071068
```

Commandes R

Affectation

- Lors d'une affectation, une expression est évaluée, mais le résultat est stocké dans un objet (variable) et rien n'est affiché à l'écran.
- Le symbole d'affectation est <-

```
a <-5
a
```

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

```
b<-5
b
```

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

Commandes R

Affectation

- Pour affecter le résultat d'un calcul dans un objet et simultanément afficher ce résultat, il suffit de placer l'affectation entre parenthèses pour ainsi créer une nouvelle expression

```
(a <- 2 + 3)
```

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

- Le symbole d'affectation inversé -> existe aussi, mais il est rarement utilisé.

Remarque

- Éviter d'utiliser l'opérateur = pour affecter une valeur à une variable puisque cette pratique est susceptible d'engendrer de la confusion avec les constructions `nom = valeur` dans les appels de fonction.

Commande R

Séparation des commandes

- Que ce soit dans les fichiers de script ou à la ligne de commande, on sépare les commandes R les unes des autres par un point-virgule ou par un retour à la ligne.
- On considère généralement comme du mauvais style d'employer les deux, c'est- à-dire de placer des points-virgules à la fin de chaque ligne de code, surtout dans les fichiers de script.
- Le point-virgule peut être utile pour séparer deux courtes expressions ou plus sur une même ligne :

```
a <- 5; a+2
```

```
## [1] 7
```

Commandes R

- On peut regrouper plusieurs commandes en une seule expression en les entourant d'accolades {}
- Le résultat du regroupement est **la valeur de la dernière commande**

```
{
  a <- 2+3
  b <- a
  b
}
```

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

Commandes R

- Par conséquent, si le regroupement se termine par une assignation, aucune valeur n'est retournée ni affichée à l'écran

```
{
  a <- 2+3
  b <- a
}
```

- Comme on peut le voir ci-dessus, lorsqu'une commande n'est pas complète à la fin de la ligne, l'invite de commande de R change de > à + pour nous inciter à compléter notre commande.

Conventions pour les noms d'objets

- Les caractères permis pour les noms d'objets sont les lettres minuscules a-z et majuscules A-Z, les chiffres 0-9, le point « . » et le caractère de soulignement «_».
  - Selon l'environnement linguistique de l'ordinateur, il peut être permis d'utiliser des lettres accentuées, mais cette pratique est fortement découragée puisqu'elle risque de nuire à la portabilité du code.
- Les noms d'objets ne peuvent commencer par un chiffre. S'ils commencent par un point, le second caractère ne peut être un chiffre.
- R est sensible à la casse, ce qui signifie que **foo, Foo et FOO** sont trois objets distincts.

Coneventions pour les noms

Mots réservés

- Certains noms sont utilisés par le système R, aussi vaut-il mieux éviter de les utiliser. c, q, t, C, D, I, diff, length, mean, pi, range, var.

- Certains mots sont réservés et il est interdit de les utiliser comme nom d'objet. Les mots réservés pour le système sont : break, else, for, function, if, in, next, repeat, return, while, TRUE, FALSE, Inf, NA, NaN, NULL, NA_integer_, NA_real_, NA_complex_, NA_character_, 1, ..2, etc.

Coneventions pour les noms

Les booléens

- Les variables T et F prennent par défaut les valeurs TRUE et FALSE , respectivement, mais peuvent être réaffectées :

```
## [1] TRUE
## [1] FALSE
## [1] 3
```

- Il est recommandé de toujours écrire les valeurs booléennes TRUE et FALSE au long pour éviter des bogues difficiles à détecter.

Les objets

Les Objets R

- Tout dans le langage R est un objet : les variables contenant des données, les fonctions, les opérateurs, même le symbole représentant le nom d'un objet est lui-même un objet.
- Les objets possèdent au minimum un mode et une longueur et certains peuvent être dotés d'un ou plusieurs attributs
- Le mode d'un objet est obtenu avec la fonction mode

```
v <- c(1,2,3,4)
mode(v)
```

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

- La longueur d'un objet est obtenue avec la fonction length

```
length(v)
```

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

Modes et types de données

- Le mode d'un objet est la nature des éléments qui le composent, leur type. On y accède par la fonction mode() ou class() ou encore typeof().
- À chaque mode correspond une fonction du même nom servant à créer un objet de ce mode.
- Les objets de mode "numeric", "complex", "logical" et "character" sont des objets simples (atomic en anglais) qui ne peuvent contenir que des données d'un seul type.

Modes et types de données

- En revanche, les objets de mode "list" ou "expression" sont des objets récursifs qui peuvent contenir d'autres objets.
- Par exemple, une liste peut contenir une ou plusieurs autres listes ;
- La fonction typeof permet d'obtenir une description plus précise de la représentation interne d'un objet
- Le mode et le type d'un objet sont identiques.

Modes et types de données

numeric(numérique)

- On distingue deux types numériques, à savoir les integers (entiers) et les double ou real (réels)

```
a <- 2.0
typeof(a)
```

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

```
is.integer(a) # a est un réel, pas un entier
```

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

```
b <- 2
c <- as.integer(b)
typeof(c)
```

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

```
is.numeric(c)
```

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

Modes et types de données

numeric(numérique)

- la fonction is.integer() retourne TRUE lorsque l'objet qui est fourni en paramètre est un entier, FALSE sinon.
- De manière plus générale, les instructions commençant par is. et suivies du nom d'un mode permettent de tester si l'objet indiqué en paramètre est de ce mode

Modes et types de données

character (caractère).

- Les chaînes de caractères sont placées entre guillemets simples ’ ou doubles ".

logical (logique, booléen)

Les données de type logique peuvent prendre deux valeurs : TRUE ou FALSE. Elles répondent à une condition logique.

Remarque - Il peut parfois être pratique d’utiliser le fait que TRUE peut être automatiquement converti en 1 et FALSE en 0.

```
TRUE + TRUE + FALSE + TRUE*TRUE
```

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

Modes et types de données

```r
a <- "Hello world!"
a
```

```
## [1] "Hello world!"
```

```r
a <- 1 ; b <- 2
a < b
```

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

```r
a == 1 # Test d'égalité
```

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

```r
a != 1 # Test d'inégalité
```

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

```r
is.character(a) ## [1] FALSE (a <- TRUE)
```

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

Modes et types de données

- Les nombres complexes sont caractérisés par leur partie réelle, que l'on peut obtenir à l'aide de la fonction Re() ; et par leur partie imaginaire, que l'on obtient grâce à la fonction Im(). On créé un nombre complexe à l'aide de la lettre i.

```
1i
```

```
## [1] 0+1i
```

```
z <- 2+3i
Re(z) # Partie réelle de z
```

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

```
Im(z) # Partie imaginaire de z
```

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

```
Mod(z) # Module de z
```

```
## [1] 3.605551
```

Modes et types de données

Mode	Contenu de l'objet
`numeric`	nombres réels
`complex`	nombres complexes
`logical`	valeurs booléennes (vrai/faux)
`character`	chaînes de caractères
`function`	fonction
`list`	données quelconques
`expression`	expressions non évaluées

Longueur

- La longueur d'un objet est égale au nombre d'éléments qu'il contient.
- La longueur, au sens R du terme, d'une chaîne de caractères est toujours 1.
- Un objet de mode character doit contenir plusieurs chaînes de caractères pour que sa longueur soit supérieure à 1 :

```
v1 <- "actuariat"
length(v1)
```

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

```
v2 <- c("a", "c", "t", "u", "a", "r", "i", "a", "t")
length(v2)
```

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

Longeur

- Il faut utiliser la fonction nchar pour obtenir le nombre de caractères dans une chaîne :

```
nchar(v1)
```

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

```
nchar(v2)
```

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

```
v <- numeric(0)
length(v)
```

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

- Un objet peut être de longueur 0 et doit alors être interprété comme un contenant qui existe, mais qui est vide

Objet spécial NULL

- L'objet spécial NULL représente « rien », ou le vide.
- Son mode est NULL .
- Sa longueur est 0.
- Toutefois différent d'un objet vide :
  - un objet de longueur 0 est un contenant vide ;
  - NULL est « pas de contenant ».
- La fonction is.null() teste si un objet est NULL ou non.

Valeurs manquantes, indéterminées et infinies

- Dans les applications statistiques, il est souvent utile de pouvoir représenter des données manquantes.
- Dans R, l'objet spécial NA remplit ce rôle.
- Par défaut, le mode de NA est logical, mais NA ne peut être considéré ni comme TRUE, ni comme FALSE.
- Toute opération impliquant une donnée NA a comme résultat NA .
- Certaines fonctions (sum, mean, par exemple) ont par conséquent un argument na.rm qui, lorsque TRUE, élimine les données manquantes avant de faire un calcul.

Valeurs manquantes, indéterminées et infinies

- La valeur NA n'est égale à aucune autre, pas même elle-même (selon la règle ci- dessus, le résultat de la comparaison est NA) :

```
## [1] NA
```

- Par conséquent, pour tester si les éléments d'un objet sont NA ou non il faut utiliser la fonction is.na:

```
is.na(NA)
```

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

Valeurs manquantes, indéterminées et infinies

- La norme IEEE 754 régissant la représentation interne des nombres dans un ordinateur (IEEE, 2003) prévoit les valeurs mathématiques spéciales $+\infty$ et $-\infty$ ainsi que les formes indéterminées du type type $0/0$ ou "$\infty - infty$".
- R dispose d'objets spéciaux pour représenter ces valeurs.
- Inf représente $+\infty$.
- -Inf représente $-\infty$.
- NaN (Not a Number) représente une forme indéterminée.
- Ces valeurs sont testées avec les fonctions is.infinite,is.finite et is.nan

Attributs

- Les attributs d'un objet sont des éléments d'information additionnels liés à cet objet.
- La liste des attributs les plus fréquemment rencontrés :

Attribut	Utilisation
`class`	affecte le comportement d'un objet
`dim`	dimensions des matrices et tableaux
`dimnames`	étiquettes des dimensions des matrices et tableaux
`names`	étiquettes des éléments d'un objet

- Pour chaque attribut, il existe une fonction du même nom servant à extraire l'attribut correspondant d'un objet.
- Plus généralement, la fonction attributes permet d'extraire ou de modifier la liste des attributs d'un objet.
- On peut aussi travailler sur un seul attribut à la fois avec la fonction attr.

Attributes

- On peut ajouter à peu près ce que l'on veut à la liste des attributs d'un objet.
- Par exemple, on pourrait vouloir attacher au résultat d'un calcul la méthode de calcul utilisée :

```
x <- 3
attr(x, "methode") <- "au pif"
attributes(x)
```

```
## $methode
## [1] "au pif"
```

- Extraire un attribut qui n'existe pas retourne NULL :

```
dim(x)
```

```
## NULL
```

Attributs

-À l'inverse, donner à un attribut la valeur NULL efface cet attribut :

```
attr(x, "methode") <- NULL
attributes(x)
```

```
## NULL
```

Les Vecteurs

Les vecteurs

- En R, à toutes fins pratiques, tout est un **vecteur**.
- Contrairement à certains autres langages de programmation, il n'y a pas de notion de scalaire en R ; un scalaire est simplement un vecteur de longueur 1.
- Le vecteur est l'unité de base dans les calculs.
- Dans un vecteur simple, tous les éléments doivent être du même mode.
- Les fonctions de base pour créer des vecteurs sont :
- c : concaténation;
- numeric : vecteur de mode numeric ;
- logical : vecteur de mode logical;
- character : vecteur de mode character.

Les vecteurs

- Il est possible et souvent souhaitable de donner une étiquette à chacun des éléments d'un vecteur.

```
(v <- c(a = 1, b = 2, c = 5))
```

```
## a b c
## 1 2 5
```

```
 v <- c(1, 2, 5)
names(v) <- c("a", "b", "c")
v
```

```
## a b c
## 1 2 5
```

- Ces étiquettes font alors partie des attributs du vecteur.

Les Vecteurs

- L'indiçage dans un vecteur se fait avec les crochets `[]`.
- On peut extraire un élément d'un vecteur par sa position ou par son étiquette, si elle existe.

```
v[3]
```

```
## c
## 5
```

```
v["c"]
```

```
## c
## 5
```

Quelques vecteurs remarquables

R fournit quelques vecteurs particuliers qui sont directement accessibles :

- **LETTERS**: les 26 lettres de l'alphabet en majuscules
- **letters** : les 26 lettres de l'alphabet en minuscules
- **month.name** : les noms des 12 mois de l'année en anglais
- **month.abb**: la version abrégée des 12 mois en anglais
- **pi** : la constante mathématique *pi*

Exercices 1

- Affecter à l'objet A17 la valeur 2017
  - Quel est le type de A17 avec la fonction mode, avec la fonction typeof
  - Convertir A17 en entier,
  - Afficher son type en utilisant les deux fonctions précédente
  - Vérifier que A17 est un entier
- Créer le complexe Z=9+5i puis extraire les parties réelle et imaginaire
- Calculer le module de Z
- Créer le vecteur v des entiers compris entre 5 et 17
  - Quelle est la longueur de v

Exercices 1

- Créer le vecteur noms contenant au moins 3 noms d'étudiants
  - Quelle la longueur de noms (nombre d'éléments)
  - Donner la longueur (nombre de caractères) de chaque nom du vecteur noms
  - Ajouter au vecteur v deux éléments sans valeurs (NA)
  - Calculer la moyenne des éléments de v (fonction mean)
  - Soit une variable unEtudiant ayant comme valeur votre numéro d'étudiant. Ajouter les attributs nom, prenom et age. Afficher touts le attributs de unEtudiant.

Exercices 2

- Créer un vecteur de 5 entiers, puis un autre de 5 booléens et un dernier de 5 chaines de caractères
- Nommer les éléments de chacun des vecteurs ci-dessus.
- Créer en nommant les éléments un vecteur contenant des âges de 5 étudiants
- Afficher les deux derniers éléments du vecteur en utilisant les noms
- Vecteurs prédéfinies

Matrices et tableaux

Matrices et tableaux

- Le R étant un langage spécialisé pour les calculs mathématiques, il supporte la manipulation des matrices et, plus généralement, les tableaux à plusieurs dimensions.
- Les matrices et tableaux ne sont rien d'autre que des vecteurs dotés d'un attribut dim.
- Ces objets sont donc stockés, et peuvent être manipulés, exactement comme des vecteurs simples.

Matrices et tableaux

- Une matrice est un vecteur avec un attribut dim de longueur 2.
- Cela change implicitement la classe de l'objet pour **matrix** et, de ce fait,
- le mode d'affichage de l'objet ainsi que son interaction avec plusieurs opérateurs et fonctions.
- La fonction de base pour créer des matrices est matrix:

```
matrix(1:6, nrow = 3, ncol = 2)
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

Matrices et tableaux

- La fonction matrix a un argument byrow qui permet d'inverser l'ordre de remplissage.

```
matrix(1:6, nrow=2, ncol=3, byrow=TRUE)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

Matrices et tableaux

- La généralisation d'une matrice à plus de deux dimensions est un tableau (array).
- Le nombre de dimensions du tableau est toujours égal à la longueur de l'attribut dim. La classe implicite d'un tableau est "array".

Matrices et tableaux

- La fonction de base pour créer des tableaux est array :

```
array(1:24, dim = c(3, 4, 2))
```

```
## , , 1
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
##
## , , 2
##
##      [,1] [,2] [,3] [,4]
## [1,]   13   16   19   22
## [2,]   14   17   20   23
## [3,]   15   18   21   24
```

Matrices et tableaux

- On extrait un élément d'une matrice en précisant sa position dans chaque dimension de celle-ci, séparées par des virgules :

```
##      [,1] [,2] [,3]
## [1,]   40   45   55
## [2,]   80   21   32
```

```
## [1] 45
```

Matrics et tableaux

- On peut aussi ne donner que la position de l'élément dans le vecteur sousjacent :

```
m[3]
```

```
## [1] 45
```

- Lorsqu'une dimension est omise dans les crochets, tous les éléments de cette dimension sont extraits :

```
m[2, ]
```

```
## [1] 80 21 32
```

- Les idées sont les mêmes pour les tableaux.

Matrices et tableaux

- Des fonctions permettent de fusionner des matrices et des tableaux ayant au moins une dimension identique.
- La fonction rbind permet de fusionner verticalement deux matrices (ou plus) ayant le même nombre de colonnes.

```
n <- matrix(1:9, nrow = 3)
rbind(m, n)
```

```
##      [,1] [,2] [,3]
## [1,]   40   45   55
## [2,]   80   21   32
## [3,]    1    4    7
## [4,]    2    5    8
## [5,]    3    6    9
```

Matrices et tableaux

- La fonction cbind permet de fusionner horizontalement deux matrices (ou plus) ayant le même nombre de lignes.

```
n <- matrix(1:4, nrow = 2)
cbind(m, n)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   40   45   55    1    3
## [2,]   80   21   32    2    4
```

Exercices

- Créer une matrice $M1$ (4x5) d'entiers
- Créer une matrice $M2$ (4x3) contenant les entiers de 10 à 21 et ranger les éléments par ligne
- Créer une matrice $M3$ (2x5) contenant les nombres entiers compris entre 60 et 69
- Créer la matrice MC est le résultat de la fusion de $M1$ et$M2$
- Créer la matrice MR est le résultat de la fusion de $M1$ et $M3$
- Extraction d'éléments d'une matrice

Les Listes

Listes

- La liste est le mode de stockage le plus général et polyvalent du langage R.
- Il s'agit d'un type de vecteur spécial dont les éléments peuvent être de n'importe quel mode, y compris le mode list.
- Cela permet donc d'emboîter des listes, d'où le qualificatif de **récursif** pour ce type d'objet.

Listes

- La fonction de base pour créer des listes est list :

```
(x <- list(size = c(1, 5, 2), user = "Joe", new = TRUE))
```

```
## $size
## [1] 1 5 2
##
## $user
## [1] "Joe"
##
## $new
## [1] TRUE
```

Listes

- Ci-dessus, le premier élément de la liste est de mode "numeric", le second de mode "character" et le troisième de mode "logical".
- Il est recommandé de nommer les éléments d'une liste.
  - En effet, les listes contiennent souvent des données de types différents et il peut s'avérer difficile d'identifier les éléments s'ils ne sont pas nommés.
  - De plus, comme nous le verrons ci-dessous, il est très simple d'extraire les éléments d'une liste par leur étiquette.
- La liste demeure un vecteur. On peut donc l'indicer avec l'opérateur [ ].
  - Cependant, cela retourne une liste contenant le ou les éléments indicés. C'est rarement ce que l'on souhaite.

Listes

- Pour indicer un élément d'une liste et n'obtenir que cet élément, et non une liste contenant l'élément, il faut utiliser l'opérateur d'indiçage [[ ]]
- Comparer

```
x[1]
```

```
## $size
## [1] 1 5 2
```

```
x[[1]]
```

```
## [1] 1 5 2
```

listes

- Évidemment, on ne peut extraire qu'un seul élément à la fois avec les crochets doubles [[ ]].
- Si l'indice utilisé dans [[ ]] est un vecteur, il est utilisé récursivement pour indicer la liste :
  - cela sélectionnera la composante de la liste correspondant au premier élément du vecteur, puis l'élément de la composante correspondant au second élément du vecteur, et ainsi de suite.

listes

- La meilleure façon d'indicer un seul élément d'une liste est par son étiquette avec l'opérateur $

```
x$size
```

```
## [1] 1 5 2
```

- La fonction unlist convertit une liste en un vecteur simple.
- Elle est surtout utile pour concaténer les éléments d'une liste lorsque ceux-ci sont des scalaires.
- Attention, cette fonction peut être destructrice si la structure interne de la liste est importante.

Exercices

1. Créer une liste contenant trois éléments dont :
   - Le nom de votre classe
   - Les noms de 4 étudiants de votre classe
   - Et l'année encours
2. Afficher tous les éléments de la liste
3. Afficher tous les noms des étudiants
4. Afficher le nom de chaque étudiants un à un
5. Afficher les noms des premier et dernier étudiants
6. Nommer les éléments du vecteur correspondant au nom des étudiants par les prénoms des étudiants concernés
7. Afficher à nouveau les étudiants

Data Frames

Data Frames

- Les **vecteurs, les matrices, les tableaux** et les **listes** sont les types d'objets les plus fréquemment utilisés en programmation en R.
- Toutefois, un grand nombre de procédures statistiques (pensons à la régression linéaire, par exemple) repose davantage sur les data frames pour le stockage des données.
- Un data frame est une liste de classe "data.frame" dont tous les éléments sont de la même longueur (ou comptent le même nombre de lignes si les éléments sont des matrices).

Data Frames

- Il est généralement représenté sous la forme d'un tableau à deux dimensions.
- Chaque élément de la liste sous-jacente correspond à une colonne.
- Bien que visuellement similaire à une matrice un data frame est plus général puisque les colonnes peuvent être de **modes** différents ;
  - pensons à un tableau avec des noms (mode character) dans une colonne et des notes (mode numeric) dans une autre.

Data Frames

- On crée un data frame avec la fonction `data.frame()` ou, pour convertir un autre type d'objet en data frame, avec `as.data.frame()`.
- Le data frame peut être indicé à la fois comme une liste et comme une matrice.
- Les fonctions `rbind` et `cbind` peuvent être utilisées pour ajouter des lignes ou des colonnes à un data frame.

Remarque

- L'élément distinctif entre un data frame et une liste générale, c'est que tous les éléments du premier doivent être de la même longueur et que, par conséquent, R les dispose en colonnes.
- Nous avons donc ici le type d'objet tout désigné pour stocker des données de modes différents, mais qui se présentent sous forme de tableau à deux dimensions.

Exercices

- Créer 3 vecteurs nom, prenom et age contenant respectivement les noms, prénoms et ages de 5 étudiants
- Créer un dataframe nommé etudiants à partir des 3 vecteurs précédents
- Afficher le dataframe etudiants
- Afficher les deux premières colonnes du dataframe
- Afficher le prénom du troisième étudiant
- Modifier l'age du 5ième étudiant et afficher le ligne de cet étudiant
- Ajouter un sixième étudiant au dataframe
- Ajouter une colonne adresse avec des valeurs au dataframe

Indexation

Indexation

- L'indexation sert principalement à deux choses : soit extraire des éléments d'un objet avec la construction `x[i]`, ou les remplacer avec la construction `x[i] <- y`.
- De même, les opérations d'extraction et de remplacement d'un élément d'une liste sont de la forme `x$etiquette` et `x$etiquette <- y`
- Il existe plusieurs façons d'indicer un vecteur dans le langage R.
- Dans tous les cas, l'indiçage se fait à l'intérieur de crochets [ ].

Indexation

- Avec un vecteur d'entiers positifs. Les éléments se trouvant aux positions correspondant aux entiers sont extraits du vecteur, dans l'ordre. C'est la technique la plus courante :

```
x <-  c(A=1 , B=2, C=3, D=20, F=-10)
x[c(1,2)]
```

```
## A B
## 1 2
```

- Avec un vecteur d'entiers négatifs. Les éléments se trouvant aux positions correspondant aux entiers négatifs sont alors éliminés du vecteur :

```
x <-  c(A=1 , B=2, C=3, D=20, F=-10)
x[c(-1,-5)]
```

```
##  B  C  D
##  2  3 20
```

Indexation

Avec un vecteur booléen. Le vecteur d'indiçage doit alors être de la **même longueur** que le vecteur indicé. Les éléments correspondant à une valeur `TRUE` sont extraits du vecteur, alors que ceux correspondant à `FALSE` sont éliminés :

```
x <-  c(A=1 , B=2, C=3, D=20, F=-10)

x[c(TRUE, TRUE , FALSE , FALSE, FALSE)]
```

```
## A B
## 1 2
```

Indexation

- Avec un vecteur de chaînes de caractères. Utile pour extraire les éléments d'un vecteur à condition que ceux-ci soient nommés :

```
v <-c(A=1, B=2, C=3)
v[c("A","B")]
```

```
## A B
## 1 2
```

Indexation

- L'indice est laissé vide. Tous les éléments du vecteur sont alors sélectionnés :

```
x[]
```

```
##   A   B   C   D   F
##   1   2   3  20 -10
```

- Cette méthode est essentiellement utilisée avec les matrices et tableaux pour sélectionner tous les éléments d'une dimension.
  - Laisser l'indice vide est différent d'indicer avec un vecteur vide ; cette dernière opération retourne un vecteur vide.

Exercice

- Créer 2 vecteurs nom et age contenant respectivement les noms et âges de 5 étudiants
- Afficher le nom du troisième étudiant
- Afficher les âges des 2ième ,3ième et 4ième étudiants
- Afficher les noms des 1er et 4ième étudiants
- Afficher les noms de tous les étudiants sauf le 3ième

Exercice(suite)

- Créer un vecteur prenom contenant les prénoms des 5 étudiants
- Nommer les éléments des deux vecteurs par les prénoms des étudiants
- Afficher le nom du troisième étudiant en indiçant par son prénom
- Afficher les ages des 2ième ,3ième et 4ième étudiants en indiçant par leurs prénoms
- Afficher les noms des 1er et 4ième étudiants en indiçant par leurs prénoms

Exercice(suite)

- Créer un vecteur booléen avec la condition age>23
- Afficher les noms des étudiants âgés de plus de 23 ans en utilisant le vecteur booléen précédent

Opérateurs arithmétiques

Opérateurs arithmétiques

- L'unité de base en R est le vecteur.
- Les opérations sur les vecteurs sont effectuées **élément par élément** :

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

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

```
1:3 * 4:6
```

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

Opérateurs arithmétiques

- Si les vecteurs impliqués dans une expression arithmétique ne sont pas de la même longueur, les plus courts sont recyclés de façon à correspondre au plus long vecteur.
  - Cette règle est particulièrement apparente avec les vecteurs de longueur 1 :

```
1:10 + 2
```

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

```
1:10 + rep(2, 10)
```

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

Opérateurs arithmétiques

- Si la longueur du plus long vecteur est un multiple de celle du ou des autres vecteurs, ces derniers sont recyclés un nombre entier de fois :

```
1:10 + 1:5 + c(2, 4) # vecteurs recyclés 2 et 5 fois
```

```
##  [1]  4  8  8 12 12 11 11 15 15 19
```

```
1:10 + rep(1:5, 2) + rep(c(2, 4), 5) # équivalent
```

```
##  [1]  4  8  8 12 12 11 11 15 15 19
```

Opérateurs arithmétiques

- Sinon, le plus court vecteur est recyclé un nombre fractionnaire de fois, mais comme ce résultat est rarement souhaité et provient généralement d'une erreur de programmation, un avertissement est affiché :

```
1:10 + c(2, 4, 6)
```

```
## Warning in 1:10 + c(2, 4, 6): longer object length is no
## object length
```

```
##  [1]  3  6  9  6  9 12  9 12 15 12
```

Opérateurs arithmétiques

Opérateur	Fonction
$	extraction d'une liste
^	puissance
-	changement de signe
:	génération de suites
%*% %% %/%	produit matriciel, modulo, division entière
* /	multiplication, division
+ -	addition, soustraction
< <= == >= > !=	plus petit, plus petit ou égal, égal, plus grand ou égal, plus grand, différent de
!	négation logique
& &&	« et » logique
|, ||	« ou » logique
-> ->>	assignation
<- <<-	assignation

Opérateurs

- Les opérateurs de puissance (`^`) et d'assignation à gauche (`<-`, `<<-`) sont évalués de droite à gauche
- Tous les autres de gauche à droite.
  - Ainsi, `2^2^3` est `2^8`, et non `4^3`, alors que `1 - 1 - 1` vaut `-1`, et non `1`.

Exercices

- Dans la suite de l'exercice précédent afficher les nom des étudiants agés au moins de 23 ans.
- Calculer le produit de deux matrices M1 et M2
- Calculer le produit matriciel de deux matrices M3 et M4
- Calculer la somme des matrices M5 et M6 de mêmes formats
- Créer deux vecteurs booléens et afficher le résultat de la conjonction puis de la disjonction de ces vecteurs