cours R 01: Types et structures des données sur R

Une brève présentation de R

R est un langage de programmation et un logiciel libre destiné aux statistiques et à la science des données soutenu par la R Foundation for Statistical Computing

- **Date de première version** : Août 1993

- **Dernière version** : 4.2.2 (2022-10-31)
- **Auteur** : Ross Ihaka et Robert Gentleman
- **Écrit en** : C, Fortran et R
- **Extensions de fichiers** : r,.R,.RDATA,.rds,.rda,.Rpres,.Rmd,.fst
- **Typage** : dynamique
- **Développeurs** : R Core Team

Le nom du logiciel est inspiré des initiales des noms des auteurs

Interfaces

- **RStudio**: environnement de développement intégré qui permet de travailler en R
- **Jupyter**: une application web permettant de développer des notebooks en Python, Julia ou en **R**. lien pour voir comment installer R sur Jupyter.
- Le langage R est intégré à certains Système de Gestion de Bases de Données Relationelles (SGBDR) comme **SQL Server** depuis la version

communauté de contributeurs

- Comprehensive R Archive Network (CRAN)
- GitHub
- Bioconductor

La mission du projet Bioconductor est de développer, soutenir et diffuser des logiciels libres et gratuits qui facilitent l'analyse rigoureuse et reproductible des données issues d'essais biologiques actuels et émergents.

Communautés d'utilisateurs

Stack Overflow site de questions & réponses autour de R de RStudio

Prise en main

création d'un objet R

```
mon_objet <- "mon objet"
mon_objet
```

```
## [1] "mon objet"
```

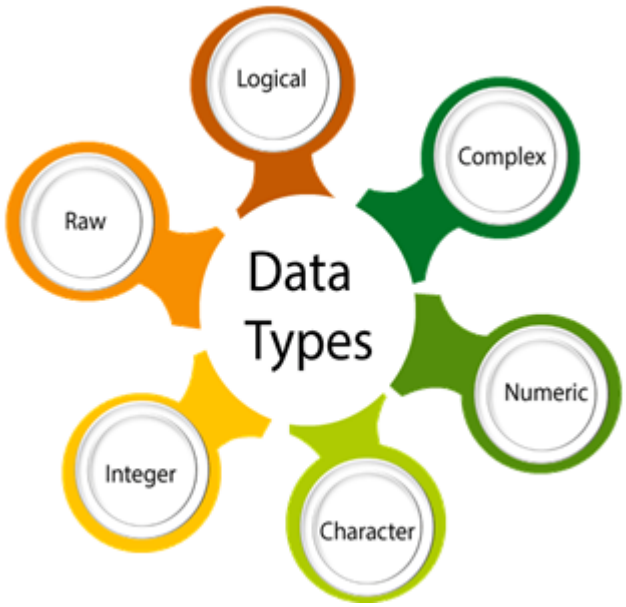
```r
object=1

print(object)
```

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

types et structures des données

- quels sont les differents types de données (usuels) sur R?
- quelles sont les différentes structures de données (usuelles) sur R?
- comment accéder à une information stockée sous une structure donnée ?

les type de données

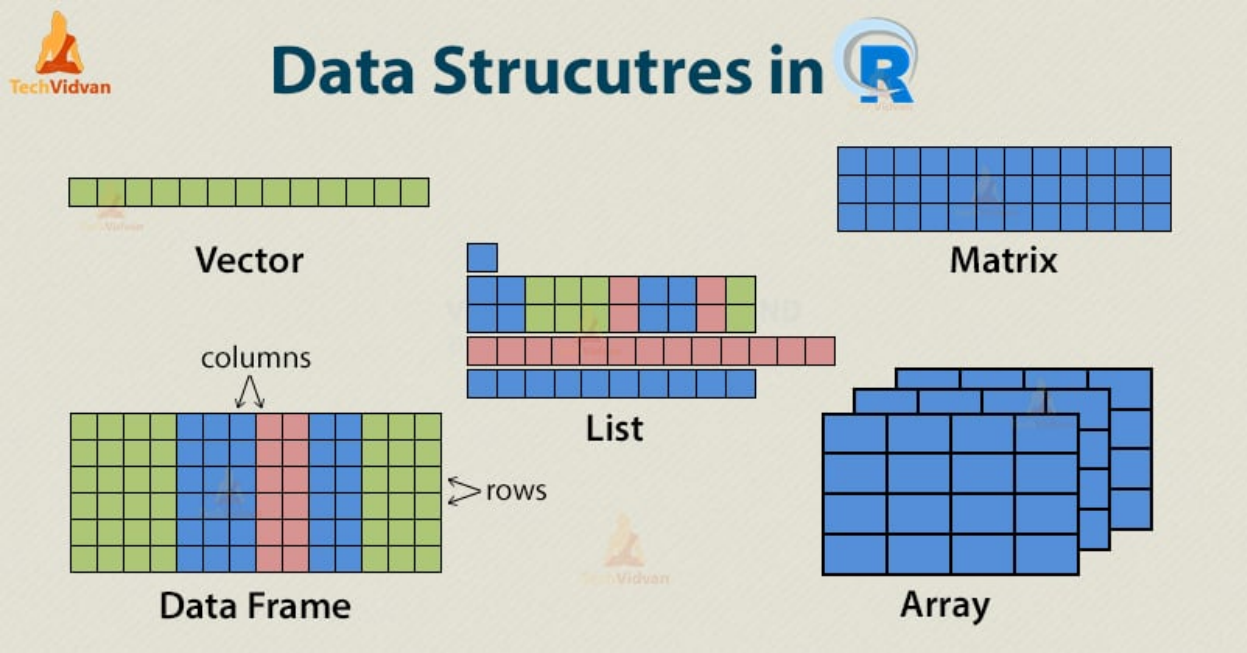


- les entiers (`integer`)
- Les réels (`double`)
- Les chaine de caractères (`charecter`)
- Les bouléens (`TRUE` ou `FALSE`)
- Les complexes (`complex`)

Dans la mesure du possible (règle de coercion), on peut convertir un objet d'un type donné à un autre type.

Les structures des données

nous allons voir les 5 structures des données les plus usuels (`vector`, `matrix`,`List`,`dataframe` et `array`) en data science.



Les vecteurs les vecteurs stockent des données de même types

- raccourci <- : **Alt + - (Windows) ou Option + - (Mac)**

```
#créer un vecteur d'entier
entier=c(2L,3L,10L,20L)
entier
```

```
## [1]  2  3 10 20
```

```
typeof(entier)
```

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

```
is.integer(entier)
```

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

```
#créer un vecteur de réél
reel=c(2,3,10,20)
reel
```

```
## [1]  2  3 10 20
```

```
typeof(reel)
```

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

```
is.integer(reel)
```

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

```
is.double(reel)
```

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

```
#créer un vecteur de chaine de caractère
texte=c("Amsa","Amadou","Najib","Alassane")
texte
```

```
## [1] "Amsa"     "Amadou"   "Najib"    "Alassane"
```

```
bool=c(TRUE,FALSE,FALSE)
print(bool)
```

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

```
length(bool)
```

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

On peut utiliser les fonctions `length()` pour afficher la longueur du vecteur(le nombres d'éléments) et la fonction `typeof()` pour voir le type de données stockées dans le vecteur.

Selon le types de données, il existe des fonctions intégrés dans R qu'on peut appliquer à un vecteur.

```
# afficher le type de données de "entier"
typeof(entier)
```

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

```
# afficher le type de données de "entier"
typeof(reel)
```

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

```
# afficher le type de données de "entier"
typeof(text)
```

```
## [1] "closure"
```

```
# afficher le type de données de "entier"
typeof(bool)
```

```
## [1] "logical"
```

```
# afficher le type de données de "entier"
mean(reel)
```

```
## [1] 8.75
```

```
# afficher le type de données de "entier"

# repliquer le vecteur "entier" 2 fois
rep(entier,times=2)
```

```
## [1]  2  3 10 20  2  3 10 20
```

```
#verifier si les éléments du vecteur "booleen" sont des booléens
is.logical(bool)
```

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

```
# verifier si les élements du vecteur "reel" sont des entiers
is.integer(reel)
```

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

```
# convert les éléments du vecteurs "texte" en majiscule
toupper(texte)
```

```
## [1] "AMSA"     "AMADOU"   "NAJIB"    "ALASSANE"
```

```r
# convertir les les éléments du vecteur "reel" en textes
as.character(reel)
```

```
## [1] "2"  "3"  "10" "20"
```

Exercice 1

Créer le vecteur nommé `mon_vecteur` qui contient les éléménes 1,2,0.5 et "mon nom".
1. Quel est le type de `mon_vecteur`. Pourquoi il est de ce type?

Créer un vecteur nommé `logique` qui contient les éléménts suivant TRUE, FALSE,TRUE.
2. exécuter la commande `sum(logique)`. Y'a t-il une explication à ce résultat?

3. qu'apelle t-on règle de coersion en R?

Accès au élémént d' un vecteur

pour acceder aux éléments d'un vecteur on utilise les signes `[` et `]` `ou $ (si le vecteur est nommé)`

*`vecteur[liste_des_indices]`

```r
#extraire le premier élément du vecteur "entier"
entier[1]
```

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

```r
#extraire les éléments du vecteur "entier" sauf le premier élément
entier[-1]
```

```
## [1]  3 10 20
```

```r
#extraire les 3 premiers éléments du vecteur "texte"
texte[1:3]
```

```
## [1] "Amsa"   "Amadou" "Najib"
```

```r
#extraire le premier et le troisième élément du vecteur "reel"
reel[c(1,3)]
```

```
## [1]  2 10
```

```r
indices=c(1,3)
#extraire le premier et le troisième élément du vecteur "reel"
reel[indices]
```

```
## [1]  2 10
```

```r
#extraire tous les éléments du vecteur "reel" sauf le premier et le troisième élément.
reel[-indices]
```

```
## [1]  3 20
```

Les matrices

stockent des informations tabulaires ayant des éléments de même type
Un attribut important de la matrice et la dimension (nombre de ligne et nombre de colonnes): `dim(matrice)`

```r
# créer une matrice de réels de dimension 3x3
mat1=matrix(1:9, nrow = 3)
mat1
```

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

```
## [3,]    3    6    9
```

```
#créer une matrice 2x13 contenant les 26 lettres de alphabet
mat=matrix(LETTERS,nrow = 2)
mat
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,] "A"  "C"  "E"  "G"  "I"  "K"  "M"  "O"  "Q"  "S"   "U"   "W"   "Y"
## [2,] "B"  "D"  "F"  "H"  "J"  "L"  "N"  "P"  "R"  "T"   "V"   "X"   "Z"
```

```
#céer une matrice de dimension 3x3 contenant des nombres aléatoires
m=matrix(round(runif(16)*100),nrow = 4)
m
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   13   86   87   35
## [2,]   61    3   97   45
## [3,]   86   24   87   76
## [4,]   73   11   97    1
```

comment acceder aux éléments d'une matrice

- `m[i,j]`=éléments se situant à l'intersection de la i ieme ligne et de la jieme colonne
- `m[i,]`=extraire la ieme ligne
  *`m[,j]`=extraire la jieme colonne

Au cas où la matrice a des noms de lignes et de colonnes, `i` et `j` peuvent être ,respectivement,remplacés par "nom ligne", "nom colonne"

comment acceder aux éléments d'une matrice

```
#créer une matrice 2x2
m=matrix(round(rnorm(4)*100),ncol = 2)
m
```

```
##      [,1] [,2]
## [1,]  -23  -77
## [2,]   -5  -86
```

```
#extraire l'élément se trouvant à l'intersection de la première ligne et de la deuxième colonne
m[1,2]
```

```
## [1] -77
```

```
#extraire la deuxième ligne
m[2,]
```

```
## [1]  -5 -86
```

```
#extraire la première colonne
m[,1]
```

```
## [1] -23  -5
```

```
colnames(m)=c("col1","col2")
rownames(m)=c("row1","row2")
m
```

```
##      col1 col2
## row1  -23  -77
## row2   -5  -86
```

```r
#un autre manière de faire
m["row1","col2"]
```

```
## [1] -77
```

```r
m["row2",]
```

```
## col1 col2
##   -5  -86
```

```r
m[,"col1"]
```

```
## row1 row2
##  -23   -5
```

exercice 2

Créer une matrice quelconque de rééls de dimension 4*4.

1.calculer la moyenne de la premiere ligne 2. calculer la moyenne de la deuxieme colonne 3. calculer la somme dela diagonal 4. calci=uler la transposé de la matrice 5. extraire le diagonal de la matrice 6. calculer la trace de la matrice

Listes

Les listes stockent des données de même type ou de types différents. Un attribut important d'une list et sa longueur (le nombre d'éléments qu'elle contient)

On peut donner un nom à chaque element d'une liste à travers la fonction `names()`

NB: les éléments d'une liste peuvent avoir différentes longueurs

```r
#exemple de création d'une liste
l=list(c(1),"IAS",TRUE,entier,mat,reel, texte)
#donner un nom au différent éléments
names(l)=LETTERS[1:length(l)]
l
```

```
## $A
## [1] 1
##
## $B
## [1] "IAS"
##
## $C
## [1] TRUE
##
## $D
## [1]  2  3 10 20
##
## $E
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,] "A"  "C"  "E"  "G"  "I"  "K"  "M"  "O"  "Q"  "S"   "U"   "W"   "Y"
## [2,] "B"  "D"  "F"  "H"  "J"  "L"  "N"  "P"  "R"  "T"   "V"   "X"   "Z"
##
## $F
## [1]  2  3 10 20
##
## $G
```

```
## [1] "Amsa"     "Amadou"   "Najib"    "Alassane"
```

**Comment acceder aux éléments d'une liste

pour extraire les éléments d"une liste on utilise `[`, ou `[[`.

- `[`: extrait les éléments en tant que liste. Peut prendre plusieurs indices
- `[[`: extrait un élément d'une liste en conservant le type de l'élément.

Au cas où les éléments ont des noms, on peut utiliser `$` pour acceder aux éléments

```
#créer une nouvelle liste contituer des éléments 1 et 3

l[c(1,3)]
```

```
## $A
## [1] 1
##
## $C
## [1] TRUE
```

```
l[[4]]
```

```
## [1]  2  3 10 20
```

```
l$D
```

```
## [1]  2  3 10 20
```

```
#extrait l'élément 3 en tant que liste
class(l[[3]])
```

```
## [1] "logical"
```

```
#extrait l'élémént 3 en conservant son type
l[[3]]
```

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

```
l[c("A","C")]
```

```
## $A
## [1] 1
##
## $C
## [1] TRUE
```

```
l$C
```

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

Data Frames

un dataframe peut être vu comme une liste dont les éléments sont des vecteurs ayant la même longueur.

Les jeux de données tabulaires utilsé en statistique ou en science des données sont souvent stockés sous forme de `dataframe`. Il existe beaucoup de jeux de données disponibles sur `R` comme `iris`, `mtcars`, `USArrests`, `PlantGrowth` and `ToothGrowth`.

```
#créer des vecteurs de même longueur
nom=c("Niang","BA","Seck","Ndiaye","Correa")
prenom=c("Amsata","Amadou","Najib","Alasane","Pascal")
region=c("Louga","Tamba","Dakar","Matam","Dakar")
```

```r
num=c(7L,3L,7L,8L,7L)
foot=c(FALSE,TRUE,FALSE,TRUE,TRUE)

#créer un dataframe contenant les 3 vecteurs
df=data.frame(Nom=nom,Prenom=prenom, Region=region,Number=num,Foot=foot)

df
```

```
##      Nom  Prenom Region Number  Foot
## 1  Niang  Amsata  Louga      7 FALSE
## 2     BA  Amadou  Tamba      3  TRUE
## 3   Seck   Najib  Dakar      7 FALSE
## 4 Ndiaye Alasane  Matam      8  TRUE
## 5 Correa  Pascal  Dakar      7  TRUE
```

```r
df[1,]
```

```
##     Nom Prenom Region Number  Foot
## 1 Niang Amsata  Louga      7 FALSE
```

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

```
## [1] "Amsata"  "Amadou"  "Najib"   "Alasane" "Pascal"
```

```r
df[,"Region"]
```

```
## [1] "Louga" "Tamba" "Dakar" "Matam" "Dakar"
```

```r
df$Foot
```

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

```r
df[df$Region=="Dakar" ,]
```

```
##      Nom Prenom Region Number  Foot
## 3   Seck  Najib  Dakar      7 FALSE
## 5 Correa Pascal  Dakar      7  TRUE
```

comment acceder aux éléments d'un `dataframe`?

- similaires au matrices avec l'opérateurs `[`
- similaire aux listes avec l'opérateurs `$`

```r
#créer un dataframe avec le jeux de donnée iris
df=iris

#voir les premieres lignes du dataframe
head(df)
```

```
##   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
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
#premiere éléments de la deuxième colonne
df[1,2]
```

```
## [1] 3.5
```

```r
#extraire la colonne "SepalWidth"
df$Sepal.Width
```

```
##   [1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 3.4 3.0 3.0 4.0 4.4 3.9 3.5
##  [19] 3.8 3.8 3.4 3.7 3.6 3.3 3.4 3.0 3.4 3.5 3.4 3.2 3.1 3.4 4.1 4.2 3.1 3.2
##  [37] 3.5 3.6 3.0 3.4 3.5 2.3 3.2 3.5 3.8 3.0 3.8 3.2 3.7 3.3 3.2 3.2 3.1 2.3
##  [55] 2.8 2.8 3.3 2.4 2.9 2.7 2.0 3.0 2.2 2.9 2.9 3.1 3.0 2.7 2.2 2.5 3.2 2.8
##  [73] 2.5 2.8 2.9 3.0 2.8 3.0 2.9 2.6 2.4 2.4 2.7 2.7 3.0 3.4 3.1 2.3 3.0 2.5
##  [91] 2.6 3.0 2.6 2.3 2.7 3.0 2.9 2.9 2.5 2.8 3.3 2.7 3.0 2.9 3.0 3.0 2.5 2.9
## [109] 2.5 3.6 3.2 2.7 3.0 2.5 2.8 3.2 3.0 3.8 2.6 2.2 3.2 2.8 2.8 2.7 3.3 3.2
## [127] 2.8 3.0 2.8 3.0 2.8 3.8 2.8 2.8 2.6 3.0 3.4 3.1 3.0 3.1 3.1 3.1 2.7 3.2
## [145] 3.3 3.0 2.5 3.0 3.4 3.0
```

```r
df[df$Sepal.Length<2,"Sepal.Length"]
```

```
## numeric(0)
```

Il existe plusieurs `package` permettant de manipuler avec beaucoup de facilité les dataframes comme ceux de `tidyverse`, `datatable`, `sparklyr` etc.

les operations entre vecteur se fait par element (element wise)

```r
a=c(2,3.4,2.5)
b=c(-5,2.5,10)
#addion par element
a+b
```

```
## [1] -3.0  5.9 12.5
```

```r
#sosutraction par éléments
a-b
```

```
## [1]  7.0  0.9 -7.5
```

```r
#division par éléments
a/b
```

```
## [1] -0.40  1.36  0.25
```

```r
#division modulo par éléments

#eleve chaque éléments de a à la puissance 3
a^3
```

```
## [1]  8.000 39.304 15.625
```

Exercice 3 Que se passe t-il si a et b sont des vecteurs de longueur différentes (tester!) ### Les opérateurs relationnels

- `==`: égal à
- `!=`: pas égale à
- `>` : supérieur
- `<` : inférieur
- `<=`: inferieur ou egal
- `>=`: supérieur ou egale

Les opérateurs Logiques

- `!`: NON logique
- `&`: ET logique (element wise)
- `&&`: ET logique (just the first element)

- `|`: OU logique (element wise)
- `||`: OU logique (juste the first element)

Opérateurs d'affectation

- `<-, <<-, =`: Affectation à gauche
- `->, ->>`: Affectation à droite

TP à faire avant mercredi:

1. Créér un compte [Github] (https://github.com/)
2. Télécharger et installer Git bash
3. Créer un ripository github avec comme nom `R_Nom_Prenom`
4. Ajouter moi comme collaborateur pour ce ripo avec moi à l'avec l'adresse amsata_niang@yahoo.fr
5. Créer un projet R sur le ripo et script R contenant les exercices à faire dans le ripo à partir de R Studio
6. Soumettre le script à votre ripo github à partir de R Studio

liens utiles https://happygitwithr.com/rstudio-git-github.html

https://www.youtube.com/watch?v=E2d91v1Twcc&t=422s&ab_channel=JamesDayhuff