Commandes R

- l'utilisateur de R interagit avec l'interprète R en entrant des commandes à l'invite de commandes.
- Toute commande R est soit une expression, soit une affectation

Une expression est immédiatement évaluée et le résultat est affiché à l'écran :

```
> 25-12
[1] 13
> pi
[1] 3.141593
> sin(pi/2)
[1] 1
```

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:

```
> x<-10
> y<-log(100)
> x<-300
```

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

```
> x<-4+6
> x
[1] 10
> (x<-4+8)
[1] 12
```

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.

Le point-virgule peut être utile pour séparer deux courtes expressions ou

plus sur une même ligne :

```
> b<-12;b+4
[1] 16
```

On peut regrouper plusieurs commandes en une seule expression en les entourant d'accolades { }.

```
> {
+ x<-5+3
+ y<-x
+ x
+ }
[1] 8
```

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

```
> {
+ x<-5+3
+ y<-x
+ }
```

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.

Le R est sensible à la casse, ce qui signifie que foo, Foo et FOO sont trois objets distincts.
Un moyen simple d'éviter des erreurs liées à la casse consiste à n'employer que des lettres minuscules.

Sous R les objets sont 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.

```
> a=c(1,2,3,4)
> mode(a)
[1] "numeric"
> length(a)
[1] 4
```

Les objets de mode "numeric", "complex", "logical" et "character« sont des objets simples contenant le même type de données

Par opposition aux listes qui peuvent contenir des éléments hétérogènes,

```
> l=list(1,2,3,'f')
> l
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] "f"
```

La longueur d’un objet est égale au nombre d’éléments qu’il contient.

```
>
bi=c("mohamed","salma","rym","ala
","amine")
> length(bi)
[1] 5
> r="business intelligence"
> length(r)
[1] 1
```

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

```
> nchar(r)
[1] 21
```

L’objet spécial NULL représente « rien », ou le vide
Son mode est NULL
Sa longueur est 0
La fonction is.null teste si un objet est NULL ou non

```
> is.null(bi)
[1] FALSE
> is.null(r)
[1] FALSE
> n=NULL
> n
NULL
> is.null(n)
[1] TRUE
```

```
> v=complex(0)
> v
complex(0)
> mode(v)
[1] "complex"
> length(v)
[1] 0
> is.null(v)
[1] FALSE
```

les données manquantes sont remplacés par l'objet spécial NA
le mode de NA est logical, mais NA ne peut être considéré ni comme TRUE, ni comme FALSE.
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
> a=NA
> a
[1] NA
> is.na(a)
[1] TRUE
```

Inf représente +∞.
-Inf représente −∞.

NaN (Not a Number) représente une forme indéterminée.

Ces valeurs sont testées avec les fonctions is.infinite, is.finite et is.nan

```
> 1/0
[1] Inf
> -1/0
[1] -Inf
> 0/0
[1] NaN
> 0/0
[1] NaN
> i<-0/0
> i
[1] NaN
```

Les attributs d’un objet sont des éléments d’information additionnels liés à cet objet

```
> x<-sample(x = 100, size=20, replace = TRUE)
> x
 [1] 75 33 39 14 26 96 15 81 32  8 62 49 69 10 43 24 36 17 45 84
> attr(x,"tirage au hasard")<-"methode de tirage"
> attributes(x)
$`tirage au hasard`
[1] "methode de tirage"
```

Extraire un attribut qui n’existe pas retourne NULL

```
> nrow(x)
NULL
> #A l'inverse attribuer null à un attribut efface cet attribut
```

- attr(x,"tirage au hasard")<-NULL
- > attributes(x)
- NULL

Vecteurs

> #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).

```
> c(2,5,6)
[1] 2 5 6
> vecteur=c(2, 5, 6)
```

- vecteur<-c(2, 5, 6)

- > u=c(a=12 b=8 c=3)
- > u=c(a=12, b=8, c=3)
- > u
- a b c
- 12 8 3
- > names(u)
- [1] "a" "b" "c"

x L’indiçage dans un vecteur se fait avec les crochets [ ].

```
> u[1]
 a
12
> u[3]
c
3
```

```
> u[b]
<NA>
  NA

> u["b"]
b
8
```

Opérations sur les vecteurs

```
> v=sample(c(1:12), 15, replace = T)

> v
 [1]  4  1  5  2 11  9  9 11  4  2  1  3  3 12  4
```

```
> ww=sample(c(1:12), 15, replace = F)
Error in sample.int(length(x), size, replace, prob) :
  cannot take a sample larger than the population when 'replace = FALSE'
> ww=sample(c(1:12), 15, replace = F)
Error in sample.int(length(x), size, replace, prob) :
  cannot take a sample larger than the population when 'replace = FALSE'
> ww=sample(c(1:12), 10, replace = F)
> w
Error: object 'w' not found
> ww
 [1] 11  9  1  8  6 12  2  7 10  4
```

Matrices

```
> m=matrix(1:12, nrow=3, ncol=4)
> m
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> m=matrix(c(4, 5, 9, -1, -5, 6, 5, 2, 1,0,0, 2), nrow=3, ncol=4)
> m
     [,1] [,2] [,3] [,4]
[1,]    4   -1    5    0
[2,]    5   -5    2    0
[3,]    9    6    1    2
```

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

Comme pour les vecteurs, l'indiçage des matrices et tableaux se fait avec les crochets [ ]

```
> array(c(2, -1, 3,4, 1, 0),dim=c(2, 3, 2))
, , 1

     [,1] [,2] [,3]
[1,]    2    3    1
[2,]   -1    4    0

, , 2

     [,1] [,2] [,3]
[1,]    2    3    1
[2,]   -1    4    0
```

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

```
> m=matrix(c(0, 5, 2,3, -2, pi, 3, 5, 3),nrow=3,
ncol=3)
> m
     [,1]     [,2] [,3]
[1,]    0 3.000000    3
[2,]    5 -2.000000    5
[3,]    2 3.141593    3
> m[2,4]
Error in m[2, 4] : subscript out of bounds
> m[2,3]
[1] 5
> m[1,]
[1] 0 3 3
> m[,2]
[1]  3.000000 -2.000000  3.141593
```

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

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.

```
> p=matrix(1:9,nrow=3, ncol=3)
> p
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> rbind(m,p)
     [,1]     [,2] [,3]
[1,]    0 3.000000    3
[2,]    5 -2.000000    5
[3,]    2 3.141593    3
[4,]    1 4.000000    7
[5,]    2 5.000000    8
[6,]    3 6.000000    9
```

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

```
> cbind(m,p)
     [,1]      [,2] [,3] [,4] [,5] [,6]
[1,]    0  3.000000    3    1    4    7
[2,]    5 -2.000000    5    2    5    8
[3,]    2  3.141593    3    3    6    9
```

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,

```
> l=list(size=c(2,1,4),user="CA", new=TRUE)
> l
$size
[1] 2 1 4
$user
[1] "CA"
$new
[1] TRUE
```

```
> l=list(salah=c(15, 12, 8, 6,
14), moez=c(6, 12, 4, 3),
meriem=c(12, 5,10, 6))
> l
$salah
[1] 15 12  8  6 14

$moez
[1]  6 12  4  3

$meriem
[1] 12  5 10  6
```

```
> names(l)
[1] "salah"  "moez"   "meriem«
> length(l)
[1] 3
```

Indexation

Le principe d'indexation d'une liste reste similaire à celui rencontré précédemment avec les vecteurs. Cependant, afin notamment de distinguer ces deux structures, leur syntaxe diffère légèrement. En effet, là où les vecteurs demandent une paire de crochets pour accéder à un élément précis, les listes utilisent des doubles crochets :

liste[[element]]

Les index spécifiés peuvent alors être les noms associés aux éléments de la liste, ou alors leur index numérique

```
> l[[1]]
[1] 15 12  8  6 14
> l[[salah]]
Error: object 'salah' not found
> l[["salah"]]
[1] 15 12  8  6 14
> l[[2]]
[1]  6 12  4  3
> l[[moez]]
Error: object 'moez' not found
> l[["moez"]]
[1]  6 12  4  3
```

Il est possible de sélectionner plusieurs éléments d'une liste en même temps mais il faut pour cela utiliser une syntaxe avec des crochets simples et non des crochets doubles.

Pour sélectionner les deux premiers éléments de notre liste il faudra donc employer la commande,

```
> l[1:2]
$salah
[1] 15 12  8  6 14

$moez
[1]  6 12  4  3
```

les éléments d'une liste sont affichés précédés d'un symbole $

```
> l$meriem
[1] 12  5 10  6
```

Traitement des éléments d'une liste

La fonction lapply()

Appliquer une même fonction aux différents éléments d'une liste.

lapply(x, FUN, ...)

Exemple

```
> candidats<-list(salah=c(25, 12, 45, 98, 47), bachir=c(12, 56, 24, 47, 36), mariem=c(25, 3, 18, 16, 23), imen=c(23, 14, 85, 36, 25))
> #nous avons créé une liste de candidats
> #Maintenant on va l'afficher
> candidats
$salah
[1] 25 12 45 98 47
$bachir
[1] 12 56 24 47 36
$mariem
[1] 25  3 18 16 23
$imen
[1] 23 14 85 36 25
> names(candidats)
[1] "salah"  "bachir" "mariem" "imen"
> length(candidats)
[1] 4
```

```
> lapply(candidats, min)
$salah
[1] 12

$bachir
[1] 12

$mariem
[1] 3

$imen
[1] 14
```

- lapply(candidats, min, na.rm=T)
- #na.rm: argument (données manquantes)

```
$salah
[1] 12

$bachir
[1] 12

$mariem
[1] 3

$imen
[1] 14
```

```
> lapply(candidats, max, na.rm=T)
$salah
[1] 98

$bachir
[1] 56

$mariem
[1] 25

$imen
[1] 85
```

```
> lapply(candidats, mean, na.rm=T)
$salah
[1] 45.4

$bachir
[1] 35

$mariem
[1] 17

$imen
[1] 36.6
```

objets renvoyés par lapply sont des listes,

certaines fonctions n'acceptent pas les listes comme argument d'entrée.
Exemple de travail souhaité:

- classer automatiquement les athlètes selon leur meilleure performance moyenne

- extraire les informations sur la distribution des valeurs enregistrées pour chaque athlètes grâce à la fonction summary()

La fonction **sort()** ne prend pas de liste en entrée
On applique alors la fonction **sapply**

```
> sort(sapply(candidats, mean))
mariem bachir   imen  salah
  17.0   35.0   36.6   45.4
> sort(sapply(candidats, mean, decreasing=T))
mariem bachir   imen  salah
  17.0   35.0   36.6   45.4
> sort(sapply(candidats, mean, decreasing=F))
mariem bachir   imen  salah
  17.0   35.0   36.6   45.4
> sort(sapply(candidats, mean), decreasing=F)
mariem bachir   imen  salah
  17.0   35.0   36.6   45.4
> sort(sapply(candidats, mean), decreasing=T)
 salah   imen bachir mariem
  45.4   36.6   35.0   17.0
```

Le résultat, retourné sous la forme d'un vecteur, sera alors plus simple à lire mais aussi à réutiliser. Par exemple, cet objet sera facilement traité par la fonctionsort(), ce qui n'aurait pas été possible avec une liste.

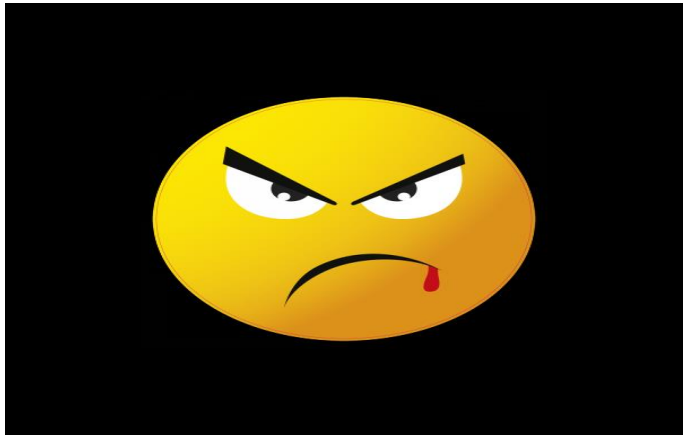
```
> sapply(candidats, summary)
        salah bachir mariem imen
Min.     12.0     12      3 14.0
1st Qu.  25.0     24     16 23.0
Median   45.0     36     18 25.0
Mean     45.4     35     17 36.6
3rd Qu.  47.0     47     23 36.0
Max.     98.0     56     25 85.0
```

Dans ce cas, R nous aura retourné une jolie matrice qui sera alors bien plus facile à lire et manipuler que la liste renvoyée par la fonctionlapply()

```
> cor.test(candidats$bachir, cnadidats$imen)
Error in cor.test.default(candidats$bachir, cnadidats$imen) :
  object 'cnadidats' not found
> cor.test(candidats$bachir, candidats$imen)

        Pearson's product-moment correlation

data:  candidats$bachir and candidats$imen
t = -0.6684, df = 3, p-value = 0.5517
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.9428168  0.7653440
sample estimates:
       cor
-0.3600241
```



```
> test<-
cor.test(candidats$bach
ir, candidats$imen)
> is.list(test)
[1] TRUE
```

Ce qui veut dire que R la renvoie comme une liste,

mais lorsqu'on veut l'afficher:

```
 > test

     Pearson's product-moment correlation

data:  candidats$bachir and candidats$imen
t = -0.6684, df = 3, p-value = 0.5517
alternative hypothesis: true correlation is not
equal to 0
95 percent confidence interval:
 -0.9428168  0.7653440
sample estimates:
       cor
-0.3600241
```

Elle n'a guère la forme d'une liste

Pour pallier ce problème nous pouvons utiliser deux fonctions. Une que vous connaissez déjà (names()) et une nouvelle :str()(structure en abrégé). La fonctionnames()vous retournera donc les noms des différents éléments de la liste. La fonctionstr()aussi, mais elle donne aussi un aperçu rapide de ce qui y est contenu.

```
> str(test)
List of 9
 $ statistic  : Named num -0.668
  ..- attr(*, "names")= chr "t"
 $ parameter  : Named int 3
  ..- attr(*, "names")= chr "df"
 $ p.value    : num 0.552
 $ estimate   : Named num -0.36
  ..- attr(*, "names")= chr "cor"
 $ null.value : Named num 0
  ..- attr(*, "names")= chr "correlation"
 $ alternative: chr "two.sided"
 $ method     : chr "Pearson's product-moment correlation"
 $ data.name  : chr "candidats$bachir and candidats$imen"
 $ conf.int   : atomic [1:2] -0.943 0.765
  ..- attr(*, "conf.level")= num 0.95
 - attr(*, "class")= chr "htest"
```

```
> names(test)
[1] "statistic"
"parameter"   "p.value"
"estimate"    "null.value"
"alternative"
[7] "method"
"data.name"   "conf.int"
```

```
> test$p.value
[1] 0.5517078
```

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

Data frame

Un data frame est une liste de classe "data.frame" dont tous les éléments sont de la même longueur

Bien que visuellement similaire à une matrice un data frame est plus général puisque les colonnes peuvent être de modes différents ; (mode character) dans une colonne et des notes (mode numeric) dans une autre.

On crée un data frame avec la fonction **data.frame**

Les fonctions rbind et cbind peuvent être utilisées pour ajouter des lignes ou des colonnes à un data frame

```
> groupe<-c("sfax","gabes", "ariane", "bizerte","béja")
> salah<-c(23, 14,10, 86, 22)
> chaima<-c(10, 15, 36, 85, 45)
> abeur<-c(3, 12, 18, 65, 14)
> meriem<-c(23, 14, 15, 85, 69)
> #on combine ensuite alors sous la forme d'une liste.
> liste<-list(groupe=groupe, salah=salah, chaime=chaima, abeur=abeur, meriem=meriem)
> liste
$groupe
[1] "sfax"    "gabes"   "ariane"  "bizerte" "béja"
$salah
[1] 23 14 10 86 22
$chaime
[1] 10 15 36 85 45 22
$abeur
[1]  3 12 18 65 14
$meriem
[1] 23 14 15 85 69
```

On constate que R ne présente pas le contenu de la liste sous forme d'un tableau.
Ce n'est donc pas le type d'objet le mieux approprié pour stocker un classement. En fait, la liste est un mode de stockage trop général pour le type de données

```
> tf<-data.frame(groupe, salah, chaima,abeur, meriem)
> tf
   groupe salah chaima abeur meriem
1    sfax    23     22     3     23
2   gabes    14     13    12     14
3  ariane    10     15    18     15
4 bizerte    86     89    65     85
5    béja    22     20    14     69
```

Exercice

Créer un data frame contenant

```
> # Création du tableau de données
>


>
> # Affichage du tableau de données
> resultats
         taille poids  QI sexe
Paul        185    82 110    M
Matthieu    178    81 108    M
Camille     165    55 125    F
Mireille    171    65  99    F
Capucine    172    68 124    F
```

Donner:

- ❑ 1) la taille de Camille.
- ❑ 2) le QI et le sexe des trois premiers individus.
- ❑ 3) toutes les données relatives à Paul et Capucine.

```
> resultats["Camille","taille"]
[1] 165

> resultats[1:3,c("QI","sexe")]
          QI sexe
Paul     110    M
Matthieu 108    M
Camille  125    F

 resultats[c("Paul", "Capucine"),]
         taille poids  QI sexe
Paul        185    82 110    M
Capucine    172    68 124    F
```

```
> resultats$taille
[1] 185 178 165 171 172
> resultats$sexe
[1] M M F F F
Levels: F M
```

créer des variables correspondant à chacune des colonnes du tableau de données. En effet, la fonction attach(x)créera pour chaque colonne du tableau de données x une variable donc le nom sera le nom de la dite colonne et le contenu les éléments de la colonne.

- attach(resultats)
- > taille
- [1] 185 178 165 171 172
- > poids
- [1] 82 81 55 65 68
- > sexe
- [1] M M F F F
- Levels: F M