MAT4056 - Estimation et analyse de données
TD – Introduction à la programmation sous R

ESIEA 4ème année – Novembre 2019

Résumé

Ce document présente rapidement le logiciel libre **R** en mettant l'accent sur les principales fonctionnalités du logiciel, la manipulation des données et des objets, l'accès aux fichiers et les procédures graphiques. **R** est un logiciel de statistique et de data mining, piloté en ligne de commande. Il est extensible (quasiment) à l'infini via le système des packages.

1 : Installation de RStudio

Commencez par installer RStudio sur votre machine : `https://www.rstudio.com/products/rstudio/`
RStudio est un IDE (environnement de développement intégré) pour R. Relativement léger et ergonomique il est à conseiller lorsqu'il faut passer à des développements plus conséquents. Une fois RStudio installé et lancé, créez un nouveau fichier (Ctrl+Shift+N)..

2 : Prise en main de R

Dans l'interface de console, tapez la commande suivante :

```
3 + 2
```

Appuyez sur *Enter* et observez le résultat suivant :

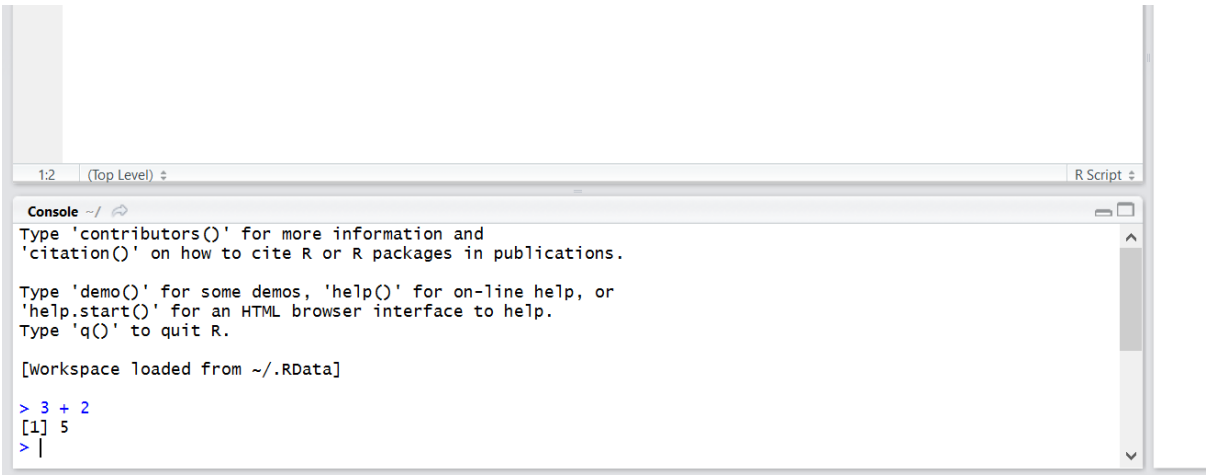

Figure 1 – Console R

Tapez ensuite les opérations suivantes :

```
> 3*5
[1] 15
> 50%%30
[1] 20
```

```
> 51%/%2
[1] 25
```

Le [1] qui apparaît devant chaque résultat indique que la réponse est un vecteur dont le premier élément est donné après le [1]. Il est possible de stocker un résultat dans les variables.

3 : Stockage d'une variable

Il est possible de déclarer et stocker une variable avec des valeurs quantitative ou qualitative. Pour que R «se souvienne» d'une variable, il suffit de l'initialiser. Cela passe par l'affectation d'une valeur à cette variable. L'opération d'affectation est réalisée par l'opérateur <-. La fonction *str()* retourne la description d'un objet. Cette fonction peut être utilisée sur n'importe quel objet R.

```
> maVariable <- 10
> maVariable
[1] 10
> maVariable * 2
[1] 20
> maVariable * maVariable
[1] 100
> maVariable <- maVariable + 1
> maVariable
[1] 11
> maVariable2 <- NULL
> maVariable2
NULL
> maVariable2 <- "Hello World"
> str(maVariable2)
chr "Hello World"
```

4 : Manipuler des vecteurs

Le symbole *:* peut s'interpréter comme « jusqu'à ». Affichez les valeurs de 1 «jusqu'à» 10 dans votre console. Affectez à la variable *a* les valeurs de 1 « jusqu'à » 10. Quand vous tapez la valeur de *a* vous devriez obtenir les valeurs suivantes. Vous pouvez additionner des vecteurs ensembles ou encore les multiplier par une valeur (entre autre).

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> a <- 1:10
> a
 [1]  1  2  3  4  5  6  7  8  9 10
> a + a
 [1]  2  4  6  8 10 12 14 16 18 20
> a * 10
 [1]  10  20  30  40  50  60  70  80  90 100
```

a est un vecteur d'entiers, possédant les valeurs de 1 à 10. Vous pouvez le vérifier en haut à droite dans Rstudio.

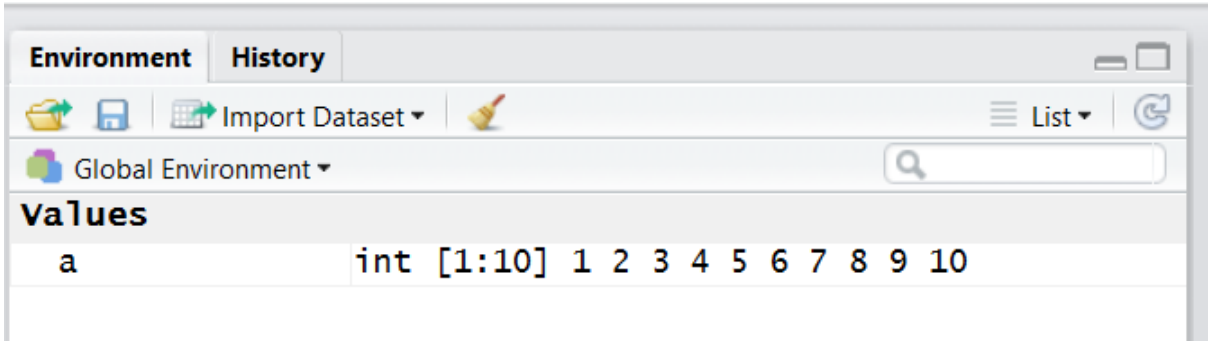

FIGURE 2 – Valeurs stockées

Cherchez maintenant à répondre aux fonctions suivantes. Vous pouvez vous aider de QuickR qui fournit un tour d'horizon rapide et complet des principales fonctions de R. `http://www.statmethods.net/`

À vous !

1. Associez les nombres de 5 à 8 dans un vecteur *b*.
2. Affichez le premier élément de *b*.
3. Affichez le 4ème élément de *b*.
4. Affichez le 5ème élément de *b*.
5. Affectez 9 au 7ème élément de *b*.
6. Affichez les nombres de 5 à 30 de 5 en 5, grâce à la fonction *seq()*.
7. Vérifiez, pour chaque élément de *a*, s'il est supérieur à 5.
8. Sélectionnez les éléments de *a* supérieurs à 5

Solution

```
> b <- 5:8
> b
[1] 5 6 7 8
> b[1]
[1] 5
> b[4]
[1] 8
> b[5]
[1] NA
> b[7]<-9
> b
[1]  5  6  7  8 NA NA  9
> seq(5,30,by=5)
[1]  5 10 15 20 25 30
> a>5
[1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
> a[a>5]
[1]  6  7  8  9 10
```

5 : Définir une fonction

Avec R, vous pouvez créer vos propres fonctions, par exemple pour la fonction :

$$f(x) = 3 * x^2 \tag{1}$$

Tapez les commandes suivantes :

```
f <- function(x) {
return (3*x^2)
}
```

À vous !

1. Affectez à la fonction *f* la valeur suivante : $f(x) = \frac{5*x^5}{4}$
2. Calculez avec R : $f(3)$
3. Créez la fonction g : $g(x, a) = a * x^2$

Solution

```
> f <- function(x) { return ((5*x^5)/4) }
> f(3)
[1] 303.75
> g <- function(x,a) { return (a*x^2) }
```

6 : Structures de contrôle

Affectez la valeur trois à la variable *a*. Comme pour les langages informatiques usuels, R dispose de structures de contrôle sur des valeurs :

```
> a <- 3
> if (a > 5) {
        print('a est plus grand que 5')
} else {
print ('a est plus petit que 5')
}
[1] "a est plus petit que 5"
```

À vous !

1. Affichez 5 fois, à l'aide d'une boucle for, le message «Hello world».

Solution

```
>for (i in 1:3) {
 print('Hello world')
 }
 [1] "Hello world"
 [1] "Hello world"
 [1] "Hello world"
 [1] "Hello world"
 [1] "Hello world"
```

7 : Les matrices et tableaux

Cette partie est la plus importante car nous manipulons beaucoup des tableaux à plus d'une dimension pour de l'analyse de données. En R, les tableaux à deux dimensions s'utilisent très simplement par l'intermédiaire de la classe matrix. Pour déclarer une matrice nous utiliserons la fonction *matrix()*. Il sera ensuite possible de réaliser vos opérations classiques sur vos matrices.

```
> m <- matrix(1,3,5) # matrice de 3 lignes 5 colonnes remplie de 1
> m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    1    1    1    1
[2,]    1    1    1    1    1
[3,]    1    1    1    1    1

> m + m
     [,1] [,2] [,3] [,4] [,5]
[1,]    2    2    2    2    2
[2,]    2    2    2    2    2
[3,]    2    2    2    2    2

> m[3,4] <- 8
> m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    1    1    1    1
[2,]    1    1    1    1    1
[3,]    1    1    1    8    1
```

Pour des tableaux multidimensionnels, il est possible de définir des tableaux à plus de 2 dimensions grâce à la fonction *array()*.

```
 p <- array(1:8,c(2,2,2)) # tableau avec 3 dimensions
 #(longueur de 2 pour chaque dimension)
> p
, , 1

     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2

     [,1] [,2]
[1,]    5    7
[2,]    6    8
```

À vous !

1. Multipliez la matrice *m* par sa transposée.
2. Utilisez la fonction retournant la valeur maximale de la matrice *m*.
3. Utilisez la fonction retournant la somme de toutes les valeurs de la matrice *m*.
4. Remplacez la 2ème colonne par une série de 4 (toute la 2ème colonne contient des 4) sur la matrice *m*.
5. Divisez par 2 chaque élément du tableau *p* grâce à la fonction *apply()*.

6. Appliquez la fonction *str()* au tableau *p*.

Solution

```
> m %*% t(m)
 [,1] [,2] [,3]
 [1,]    5    5    5
 [2,]    5    5    5
 [3,]    5    5    5
> sum(m)
 [1] 22
> max(m)
 [1] 8
> m[,2] <- rep(4,3)
> m
 [,1] [,2] [,3] [,4] [,5]
 [1,]    1    4    1    1    1
 [2,]    1    4    1    1    1
 [3,]    1    4    1    8    1

> apply(p, 1:3, function(x) x/2)
, , 1

[,1] [,2]
[1,]  0.5  1.5
[2,]  1.0  2.0

, , 2

[,1] [,2]
[1,]  2.5  3.5
[2,]  3.0  4.0

> str(p)
int [1:2, 1:2, 1:2] 1 2 3 4 5 6 7 8
```

Si vous souhaitez aller plus loin :
Des problèmes de programmation de plus en plus difficiles : `http://projecteuler.net/`.