

# ***Travail Pratique 1***

## ***Découverte du logiciel R***

STT1039  
BIOLOGIE QUANTITATIVE

Réalisé par  
Marc Pépino



Département des sciences de l'environnement

## 1. Objectifs du TP1

Le travail pratique 1 (TP 1) a pour objectif général de familiariser les étudiants à l'utilisation du logiciel R. À la fin du TP 1, l'étudiant doit savoir comment installer le logiciel, comprendre l'organisation générale du logiciel (les fenêtres et la barre de menu), connaître la programmation de base (opérations mathématiques, manipulation des objets), être capable d'organiser et de sauvegarder son travail (importation/exportation des données, fichier texte de sauvegarde). L'accent est mis sur la documentation disponible en ligne en vue d'orienter de plus en plus les étudiants vers l'autonomie.

Le site de documentation officiel de R se trouve à l'adresse suivante :  
<http://www.r-project.org/>

## 2. Installation du logiciel R

Le logiciel R est un logiciel **gratuit** depuis 1995. Créé en 1990 par Ross Ihaka et Robert Gentleman à *University of Auckland*, son développement et sa distribution sont maintenant assurés par le *R Development Core Team*. Le logiciel R permet de faire des **analyses statistiques et graphiques**. Les versions du logiciel R sont régulièrement mises à jour et téléchargeables à l'adresse suivante :

Téléchargement du logiciel R à l'adresse suivante :  
<http://cran.r-project.org/bin/windows/base/>

Après installation et ouverture du logiciel, vous devriez voir apparaître une fenêtre. Il s'agit de la **console** dans laquelle doit être tapé le code de programmation. **La barre de menu** est au dessus de la console. L'emplacement des **fenêtres graphiques** est à droite de la console :

### Ouvrir une fenêtre graphique : x11()

Lors de l'installation, de nombreuses fonctionnalités de base sont automatiquement installées. Toutefois, certaines fonctionnalités plus avancées sont incluses dans des bibliothèques (« **packages** ») qu'il faut installer manuellement à partir de la barre de menu :

### Packages -> Installer le(s) package(s)...

Vous devez ensuite choisir l'emplacement le plus proche où vous vous trouvez, puis le nom de la bibliothèque que vous souhaitez installer.

### Utiliser un package : `library(nom_de_la_bibliotheque)`

Il est fortement recommandé aux étudiants d'utiliser un fichier texte pour écrire et sauvegarder le code de programmation (voir plus de détails à la section 5, Sauvegarde de son travail). Notepad++ est un éditeur de texte gratuit qui reconnaît le langage de programmation R :

**Téléchargement de l'éditeur de texte Notepad++ à l'adresse suivante :**  
<https://notepad-plus-plus.org/>

Une alternative à l'interface classique de R est l'interface de RStudio. L'interface de RStudio est très souvent appréciée des étudiants car elle permet aux étudiants d'avoir accès à tous les outils nécessaires dans un même environnement de travail (code, console, graphiques, bases de données, aide en ligne, etc.). Un bref survol de cette interface sera vu en classe mais l'interface classique de R sera privilégiée pour le reste du cours. Il existe de versions gratuites et payantes de RStudio qui peuvent être téléchargées à l'adresse suivante :

**Téléchargement de RStudio à l'adresse suivante :**  
<https://www.rstudio.com/products/rstudio/download/>

Notons enfin la bibliothèque *Rcmdr* qui offre un environnement de travail similaire à d'autres logiciels de statistiques payant tels que SAS ou Systat. Les codes utilisés sont souvent beaucoup plus complexes et la flexibilité de R est grandement réduite. Cette bibliothèque ne sera pas utilisée dans le cadre de ce cours.

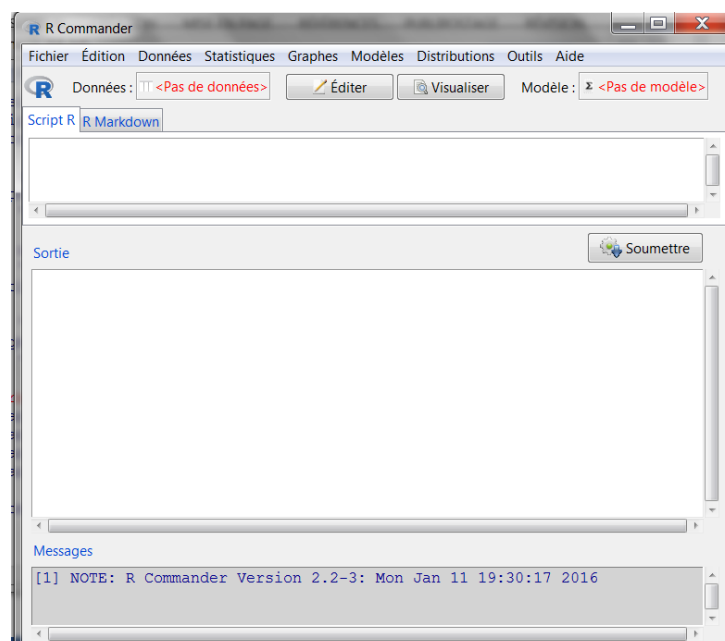


Figure 1. Interface de la bibliothèque *Rcmdr*

## 3. Programmation de base

### 3.1. Calculatrice et opérateurs

Le logiciel R fonctionne à partir d'un langage de programmation. On peut utiliser R comme **calculatrice** et taper les opérations à effectuer à la suite du curseur :

```
> #Exemple 1
> # Calculatrice: additionner 15 + 2
> 15 + 2
[1] 17
```

**Tableau 1. Opérateurs du langage R**

| Opérateur                        | Fonction                                                                                                              |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| <u>Opérateurs arithmétiques</u>  |                                                                                                                       |
| +                                | Addition                                                                                                              |
| -                                | Soustraction                                                                                                          |
| *                                | Multiplication (pour les vecteurs et les matrices, ceci est le produit Hadamard, soit le produit élément par élément) |
| /                                | Division (pour les matrices, il s'agit de la division élément par élément)                                            |
| % / %                            | Division entière de deux nombres                                                                                      |
| % %                              | Modulo (reste de la division entière)                                                                                 |
| % * %                            | Produit scalaire de deux matrices                                                                                     |
| ^                                | Exposant                                                                                                              |
| <u>Opérateurs de comparaison</u> |                                                                                                                       |
| >                                | Supérieur à                                                                                                           |
| <                                | Inférieur à                                                                                                           |
| >=                               | Supérieur ou égal à                                                                                                   |
| <=                               | Inférieur ou égal à                                                                                                   |
| = =                              | Égal à                                                                                                                |
| ! =                              | Différent de                                                                                                          |
| <u>Opérateurs logiques</u>       |                                                                                                                       |
| & ou &&                          | « et »                                                                                                                |
| ou                               | « ou »                                                                                                                |
| !                                | « non »                                                                                                               |
| <u>Opérateurs d'assignation</u>  |                                                                                                                       |
| < - ou =                         | Assigner une valeur à un objet                                                                                        |

### 3.2. Les objets

Les opérateurs d'assignation sont très utiles car ils permettent de stocker des valeurs dans une variable (objet). Par exemple, on peut assigner la valeur 17 à x et 2 à y et additionner ensuite x et y :

```
> #Exemple 2
> #Créer les variables x et y
> x <- 15
> y <- 2
> x + y
[1] 17
```

Dans cet exemple, l'objet x est un vecteur dont le mode est numérique et la longueur est 1 (x contient un élément). Les deux modes que nous utiliserons sont le **mode numérique** et le **mode catégorique**. Pour indiquer que le mode est catégorique, il faut utiliser les **guillemets**. Par exemple, si on veut assigner à la variable *periode* la valeur *jour* (catégorique), il faut taper :

```
> #Exemple 3
> #Créer une variable catégorique
> periode <- "jour"
> periode
[1] "jour"
```

Dans cet exemple, *periode* est toujours un vecteur. Les trois objets les plus courants sont les **vecteurs**, les **matrices** et les **tableaux de données** (*data frame*). Les tableaux de données sont très largement utilisés car c'est l'objet par défaut lorsqu'on importe une base de données à partir d'Excel. Nous verrons la structure de cet objet plus en détails dans les sections 3.3 et 4.

**Note importante :** le nom de l'objet est laissé à la discrétion de l'utilisateur. Dans l'exemple précédent, on pourrait remplacer *periode* par tout autre nom défini par l'utilisateur (ex : *quand*). Il faut toutefois éviter l'utilisation des accents ou des espaces dans le nom des objets.

### 3.3. Fonctions simples et manipulations d'objets

Toute fonction se compose d'un nom suivi de **parenthèses** à l'intérieur desquelles figurent un ou plusieurs arguments. Les arguments d'une fonction sont séparés par des virgules. Les trois fonctions courantes pour créer des vecteurs avec plus qu'un **élément** sont **c()**, **rep()** et **seq()**. Vous en découvrirez beaucoup d'autres en consultant la bibliographie. Pour obtenir de l'**aide en ligne** sur les fonctions, il suffit de taper « ? » suivi du nom de la fonction (voir section 6. Conseils pratiques à retenir).

## Création d'un vecteur

Un vecteur se compose d'une suite d'éléments :

```
> #Exemple 4
> #Créer un vecteur
> c(3,1,7,8)
[1] 3 1 7 8
> rep(1:3,2)
[1] 1 2 3 1 2 3
> seq(from=1,to=2,by=0.5)
[1] 1.0 1.5 2.0
```

Remarquez l'utilisation des « : » dans la fonction `rep()` et l'utilisation des arguments dans la fonction `seq()`. Pour connaître la valeur du  $i^{\text{ème}}$  élément d'un vecteur, utilisez les **crochets** :

```
> #Exemple 5
> #Récupérer la troisième valeur d'un vecteur
> x <- c(3,1,7,8)
> x[3]
[1] 7
```

## Création d'une matrice

Une matrice se compose de lignes et de colonnes. Utilisez la fonction **`matrix()`** pour spécifier le nombre de lignes et de colonnes :

```
> #Exemple 6
> #Créer une matrice
> M <- matrix(data=NA,nrow=2,ncol=3)
> M
      [,1] [,2] [,3]
[1,] NA  NA  NA
[2,] NA  NA  NA
> M <- matrix(data=1:6,nrow=2,ncol=3)
> M
      [,1] [,2] [,3]
[1,]  1   3   5
[2,]  2   4   6
> M[1,2]
[1] 3
> M[,1]
      [,1] [,2]
[1,]  1   3
[2,]  2   4
```

Remarquez l'utilisation des crochets pour récupérer une valeur particulière de la matrice, l'absence d'indice de ligne pour récupérer toutes les lignes et l'opérateur arithmétique de soustraction pour enlever la première colonne de la matrice. Dans le logiciel R, « NA » est utilisé dans le langage R pour indiquer qu'il n'y a pas de données (données manquantes).

### Création d'un tableau de données

Un tableau de données a une structure similaire à une matrice. Les colonnes correspondent aux variables (exemples : longueur, poids, âge, etc.) et les lignes aux individus. Il s'agit du format standard des bases de données utilisées en biologie. La **conversion** d'une matrice en tableau de données se fait à l'aide de la fonction ***as.data.frame()***. Lors de l'importation d'une base de données dans le logiciel R, un objet de type « *data frame* » (tableau de données) est créé (voir section 4. Importer et exporter des données).

```
> #Exemple 7
> #Conversion d'une matrice en tableau de données
> M <- matrix(data=1:6,nrow=2,ncol=3)
> D <- as.data.frame(M)
> M
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> D
      V1 V2 V3
1 1 1 3 5
2 2 2 4 6
```

Lors de la conversion de la matrice en tableau de données, les colonnes sont renommées par défaut par les variables V1, V2 et V3. Pour renommer les colonnes, utilisez la fonction ***colnames()***.

```
> #Exemple 8
> #Renommer des colonnes
> colnames(D) <- c("x","y","z")
> D
      x y z
1 1 1 3 5
2 2 2 4 6
```

**IMPORTANT** : la **structure** d'un objet peut être analysé à l'aide des fonctions ***length()*** pour la longueur d'un objet, ***dim()*** pour la dimension d'un objet à au moins deux dimensions (ex : matrice et tableau de données) et ***str()*** pour la structure d'un objet.

```

> #Exemple 9
> #Dimension de la matrice M
> dim(M)
[1] 2 3
> #Structure d'un tableau de données
> str(D)
'data.frame': 2 obs. of 3 variables:
 $ x: int 1 2
 $ y: int 3 4
 $ z: int 5 6

```

Remarquez le signe « \$ » qui précède chacune des variables. Pour afficher les valeurs de la variable x du tableau de données D, il suffit de taper « D\$x ».

### Manipulation d'objet

La manipulation des objets permet de modifier une base de données, de sélectionner une partie de la base de données ou de naviguer à l'intérieur d'une base de données. Pour les bases de données volumineuses, il sera important de maîtriser la manipulation des objets à l'aide des fonctions appropriées.

Les fonctions **rbind()** et **cbind()** permettent respectivement d'ajouter des lignes ou des colonnes à un objet. Attention, il faut respecter les dimensions entre les objets. Par exemple, pour une matrice ayant trois colonnes et deux lignes, on peut ajouter un vecteur à deux éléments comme nouvelle colonne de la matrice mais pas comme ligne.

```

> #Exemple 10
> #Ajouter des lignes ou des colonnes à une matrice
> M <- matrix(1:6,2,3)
> y <- c(7,8)
> cbind(M,y)
      y
[1,] 1 3 5 7
[2,] 2 4 6 8
> rbind(M,y)
[,1] [,2] [,3]
  1   3   5
  2   4   6
y  7   8   7
Message d'avis :
In rbind(M, y) :
number of columns of result is not a multiple of vector length (arg 2)

```

Remarquez le message d'avertissement. L'ordinateur nous donne un résultat même si la commande n'est pas correcte.

**Les opérateurs de comparaison** peuvent être utilisés pour récupérer une partie d'un tableau de données. Dans l'exemple suivant, on crée un tableau de données *D* contenant la variable *x*, puis on ajoute la variable *periode* (variable catégorique prenant les valeurs *jour* ou *nuit*) et on récupère dans un tableau de données *J* les données dont la variable *periode* prend les valeurs *jour* :

```
> #Exemple 11
> #Créer un tableau de données
> x      <- c(2,3,1,9)
> D      <- as.data.frame(x)
> #Ajouter une variable à un tableau de données
> D$periode <- c("jour","nuit","nuit","jour")
> D
  x periode
1 2  jour
2 3  nuit
3 1  nuit
4 9  jour
> #Récupérer un sous-échantillon
> J      <- D[ D$periode == "jour", ]
> J
  x periode
1 2  jour
4 9  jour
```

### La fonction *subset()*

La fonction *subset()* est une fonction permettant de récupérer une partie d'un tableau de données dans un nouvel objet. Dans l'exemple suivant, nous verrons deux façons d'effectuer cette tâche, l'une avec la fonction *subset()*, l'autre avec les opérateurs de comparaison comme dans l'exemple précédent :

```
> #Exemple 12
> #Récupérer un sous-échantillon : option 1
> J1 <- D[ D$periode == "jour", ]
> J1
  x periode
1 2  jour
4 9  jour
> #Récupérer un sous-échantillon : option 2
> J2 <- subset(D, periode == "jour")
> J2
  x periode
1 2  jour
4 9  jour
```

### La fonction *merge()*

Lorsque les données récoltées sur le terrain sont inscrites dans deux bases de données distinctes, il est possible de fusionner ces deux bases de données sous R à condition qu'il existe un identifiant unique qui lie ces deux bases de données. Cet identifiant fait souvent référence à l'unité d'échantillonnage. Par exemple, lors de la détermination de la diversité végétale plusieurs parcelles sont échantillonnées et deux bases de données sont construites : l'une contenant les variables environnementales (ex : pH du sol, ensoleillement, etc.), l'autre avec les variables sur les espèces végétales. Le nom de la parcelle permet de faire le pont entre ces deux bases de données, à condition que chaque parcelle ait un identifiant unique.

**IMPORTANT :** Un exemple d'utilisation de la fonction *merge()* sera vu dans les exercices.

### 3.4. Fonctions de programmation avancées

Pour de la programmation plus avancée, mentionnons l'utilisation des fonctions *if()* pour donner des instructions de condition et *for()* pour effectuer des instructions en boucle. L'aide sur ces fonctions s'obtient en tapant « ? Control » ou « **help("if")** ». La fonction *ifelse()* est dans certains cas une alternative efficace à l'utilisation combiner des fonctions *if()* et *for()*. À partir du tableau de données D de l'exemple 11, on utilise les deux fonctions *if()* et *for()* pour ajouter une variable *periode2* qui prend respectivement les valeurs 0 et 1 lorsque la variable *periode* prend les valeurs *jour* et *nuit* :

```
> #Exemple 13
> #Créer la variable periode2
> #Boucle avec l'utilisation de if() et for()
> D$periode2=NA
> for(i in 1:length(D$periode)){
+ if(D$periode[i] == "jour") D$periode2[i] <- 0 else D$periode2[i] <- 1
+ }
> #Une alternative avec ifelse()
> D$periode3 = ifelse(D$periode == "jour", 0, 1)
> #Afficher le résultat
> D
  x periode periode2 Periode3
1 2  jour         0         0
2 3  nuit         1         1
3 1  nuit         1         1
4 9  jour         0         0
```

Remarquez l'utilisation des  **accolades**  dans la boucle *for()* pour indiquer toutes les instructions à effectuer en séquences à l'intérieur de la boucle (ici, on ne mentionne qu'une seule instruction). Le signe « + » apparaît à la place du curseur pour indiquer que les instructions ne sont pas terminées.

## 4. Importer et exporter des données

Il est important d'exporter les bases de données à l'extérieur du logiciel R et d'importer par la suite les bases de données sur R pour continuer à travailler lors d'une prochaine session (voir aussi *Sauvegarde de son travail* à la section 5). Je vous recommande pour débiter d'exporter et d'importer toutes vos bases de données sous format « .csv ». Ce format permet notamment de lire le fichier avec Excel.

### 4.1 Exportation

Avant d'exporter votre base de données, je vous conseille de convertir votre objet en tableau de données. La fonction **`write.table()`** permet l'exportation :

```
> #Exemple 14
> #Créer un tableau de données
> x <- seq(from=100, to=1, by=-1)
> d <- as.data.frame(x)
> #Exporter un tableau de données en un fichier csv
> write.table(d,"D:/bureau/TP1.csv",sep=";",dec=".") #Adresse sur l'ordinateur des salles UQTR
```

La fonction **`write.table`** contient quatre arguments: le nom de l'objet à exporter (*d*), le chemin d'accès entre guillemet avec le nom du fichier et son extension (ici .csv), le séparateur de données entre guillemet (";") ainsi que le séparateur de décimale entre guillemet (".").

### 4.2 Importation

Si votre base de données est sur Excell, je vous conseille de l'enregistrer sous format « .csv » avant l'importation :

**Fichier -> Enregistrer sous -> Type de fichier = CSV(DOS) -> Enregistrer.**

Lors de l'importation, l'objet se présente sous la forme d'un tableau de données. La fonction **`read.csv()`** permet l'importation.

```
> #Exemple 15
> #Importer un fichier csv
> toto <- read.csv("D:/bureau/TP1.csv",sep=";",dec=".", header=TRUE)
> tata <- read.csv(file.choose(),sep=";",dec=".", header=TRUE)
> #Structure du tableau de données
> str(tata)
'data.frame': 100 obs. of 1 variable:
 $ x: num 100 99 98 97 96 95 94 93 92 91 ...
```

Les trois premiers arguments de la fonction **`read.csv`** sont le chemin d'accès au fichier de données, le séparateur de données et le séparateur de décimale. Le chemin d'accès d'un fichier s'obtient avec un clic droit sur le fichier puis sous l'onglet *Propriétés*. L'argument **`header=TRUE`** indique que la première ligne du fichier correspond au nom des colonnes (les variables de la base de données).

**Rappel** : *toto* et *tata* sont des noms choisis par l'utilisateur. Ce ne sont pas des noms spécifiques au langage de programmation R. Vous pouvez décider de nommer ces objets différemment!

**Note importante** : Assurez-vous que l'importation s'est bien faite et que votre base de données est bien formatée sous un tableau de données. Vérifiez la dimension de votre base de données avec `dim()`, sa structure et le nom des variables avec `str()` et visualiser les premières lignes de la base de données avec `head()`. Une fois l'importation de la base de données, tout se passe à l'intérieur du logiciel R. La base de données originale n'est en aucun cas modifiée. Les données originales restent intactes et sont ainsi préservées de toute mauvaise manipulation.

**Conseil** : si l'exportation ou l'importation ne fonctionnent pas correctement, assurez-vous que le chemin d'accès ne comporte pas d'erreur et que le symbole « / » est utilisé pour séparer les répertoires. Une source d'erreur fréquente est le séparateur de données, essayer « `sep = ","` », ou le séparateur de décimale, essayer « `dec = "."` ». L'option `file.choose()` permet de parcourir son répertoire pour sélectionner son fichier en cliquant dessus.

## 5. Sauvegarde de son travail

Le code de programmation doit être écrit et sauvegardé dans un **fichier texte (extension .txt)** :

**Démarrer -> Tous les programmes -> Accessoires -> Bloc note (ou WordPad)**  
**Ou**  
**Console -> Fichier -> Nouveau script -> Fichier -> Sauver (extension .txt)**

Lors de l'ouverture du logiciel, un simple « copier-coller » (« Ctrl + C »- (« Ctrl + V ») du code dans la console permettra de récupérer tout le travail. Il est aussi possible d'ouvrir le script dans le logiciel R et de lancer les lignes de code dans la console (« Ctrl + R »).

Notepad++ est un éditeur de texte gratuit (<http://notepad-plus-plus.org/>) et reconnaît plusieurs langages de programmation dont celui utilisé par le logiciel R :

**Langage -> R -> R**

Le texte apparaît alors en couleur pour distinguer les fonctions (violet), les commentaires (vert), les nombres (orange) et les autres codes définis par l'utilisateur (noir). Enfin, il est fortement conseillé aux étudiants de structurer leur document texte en y ajoutant des commentaires et de sauvegarder leur document à la fin de chaque session afin de poursuivre le travail à une session ultérieure :

**Commentaires : Taper « # » suivi de ses propres commentaires**

```
> #Exemple 16
> #Je veux créer une variable x qui contient une série aléatoire de nombre allant de 1 à 10
> x <- sample(1:10)
> x
[1] 6 10 7 1 8 3 5 4 9 2
```

## 6. Conseils pratiques à retenir

**Sauvegarder son travail.** Se créer un document texte (extension .txt) dans lequel sera enregistré tout le code écrit durant le TP. Ouvrir le fichier texte lors d'une prochaine séance pour continuer son travail.

**Ajouter des commentaires.** Taper « # » suivi de votre commentaire.

**Importer des données en csv.** Utiliser la fonction *read.csv()*.

**Structure de la base de données.** Utilisez la fonction *str()*.

**Utiliser la fonction *subset()* et les opérateurs de comparaison** pour extraire un sous-ensemble d'un tableau de données.

**Utiliser la fonction *merge()*** pour fusionner deux tableaux de données ayant un identifiant unique (voir Exercice 5).

**Le signe « \$ »** fait le pont entre un tableau de données et les variables contenues dans ce même tableau de données.

**La virgule** peut séparer (1) les arguments d'une fonction ou (2) les lignes (premier argument) et les colonnes (deuxième argument) d'une matrice ou d'un tableau de données.

**Aide en ligne.** Tapez « ? » suivi de la fonction dont on souhaite avoir des informations

```
> #Exemple 17  
> #Obtenir de l'information sur une fonction  
> ?matrix  
> ?read.table
```

**Effacer la console.** Utilisez « Ctrl + L » ou « Edition-> Effacer la console » pour avoir une page blanche dans la console (tous les objets restent enregistrés dans la console). Dans la barre de menu, cliquez « Misc -> Effacer tous les objets » pour vider la mémoire de la console (tous les objets sont effacés... et donc perdus).

**Accès aux lignes de code antérieures.** Lors d'une même session, vous avez accès aux lignes de code antérieures en cliquant sur la flèche du haut sur votre clavier.

**Quitter le logiciel.** Cliquez sur la croix rouge en haut à droite. À la question « Sauver une image de la session », répondez NON.

## 7. Exercices

Répondez aux exercices dans un fichier texte « Nom\_Prenom\_TP1.txt ». Au début du fichier, identifiez-vous par votre nom et prénom, ajouter la date et le numéro du TP, le tout en utilisant des commentaires. Utilisez des commentaires pour structurer vos réponses. Les exercices 1 à 3 permettent de vérifier l'intégration de la programmation de base. Les exercices 4 à 6 permettent de voir comment importer et manipuler des jeux de données.

**Exercice 1.** Créer un vecteur  $x$  contenant les valeurs 2, 6 et 9. Créer un vecteur  $y$  contenant les valeurs 3, 2 et 4. Additionner les vecteurs  $x$  et  $y$ . Multiplier les vecteurs  $x$  et  $y$ .

**Exercice 2.** Créer une matrice  $M$  de deux lignes et deux colonnes. La première ligne se compose des éléments 4 et 3. La deuxième ligne se compose des éléments 2 et 1. Convertir cette matrice en tableau de données  $D$ . Afficher la matrice  $M$  et le tableau de données  $D$ .

**Exercice 3.** Créer la matrice  $M$  en utilisant le code suivant :

```
M <- matrix(data=seq(from=199.99,0,by=-0.01),nrow=1000,ncol=200)
```

Afficher la matrice  $M$  puis effacer la console en utilisant « Ctrl + L ». Quelle est la valeur de la 425<sup>ième</sup> ligne et 75<sup>ième</sup> colonne. Combien d'éléments à la matrice  $M$ . Quelle est la dimension de la matrice  $M$ . Récupérer la deuxième ligne de la matrice  $M$  pour créer un vecteur  $x$ . Enlever les 8<sup>ième</sup> et 99<sup>ième</sup> éléments de  $x$ . **Optionnel** : calculer alors la somme des éléments de  $x$  (utiliser la fonction `sum()`).

**Exercice 4.** Importer la base de données « TP1\_Exercices.xlsx » se trouvant sur votre portail étudiant (convertir d'abord le fichier en format .csv). Ces données sont aussi disponibles à l'adresse suivante : <http://datadryad.org/resource/doi:10.5061/dryad.p2k02>. Appeler ce tableau de données  $D$ . A partir du tableau de données  $D$ , récupérer dans un tableau de données  $D2$  le sous-ensemble de données répondant aux critères suivants : les données correspondant à la catégorie "12P" de la variable *Family*, les données supérieures ou égales à la valeur 0.2 de la variable *SGRlength*, récupérer seulement les colonnes 1 à 11. Afficher  $D2$ .

**Exercice 5.** A partir du tableau de données  $D$  de l'exercice 4, récupérer dans un tableau de données  $D3$  le sous-ensemble de données répondant aux critères suivants : les données correspondant à la catégorie "12P" de la variable *Family*, les données supérieures ou égales à la valeur 0.2 de la variable *SGRlength*, récupérer seulement les colonnes 1 et 12 à 23. Afficher  $D3$ . Fusionner les tableaux de données  $D2$  (de l'exercice 4) et  $D3$  dans un nouveau tableau de données  $D4$  à l'aide de la variable *ID* qui constitue l'identifiant unique (numéro du poisson). Afficher  $D4$ . Cet exercice permet de voir comment la base de données portant sur la croissance a été fusionnée avec la base de données portant sur les caractères morphologiques avant d'être déposée sur le site Dryad Digital Repository.

**Exercice 6.** A partir du tableau de données  $D$  de l'exercice 4, ajouter un variable  $Z2$  qui prends les valeurs -1 et 1 lorsque la variable *Zone* prend les valeurs *L* et *P*, respectivement. Cet exercice permet de voir comment recoder facilement une variable.

## Bibliographie

- Crawley, M. J. 2013. The R book (2<sup>ème</sup> éd.). John Wiley and Sons, Chichester, England.  
[Disponible en ligne à la bibliothèque de l'UQTR \(voir la Webthèque sur le portail de cours\)](#)
- Legendre, P. 2009. Introduction au langage statistique R.  
[http://biol09.biol.umontreal.ca/BIO6077/Introduction\\_R.pdf](http://biol09.biol.umontreal.ca/BIO6077/Introduction_R.pdf).
- Paradis, E. 2005. R pour les débutants.  
[http://cran.r-project.org/doc/contrib/Paradis-rdebuts\\_fr.pdf](http://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf).
- Short, T. 2004. R Reference Card.  
<http://cran.r-project.org/doc/contrib/Short-refcard.pdf>.

### Liens utiles :

- Site officiel de R.  
<http://www.r-project.org>
- Ateliers R du CSBQ.  
<https://qcbs.ca/wiki/r>
- Quick R.  
<http://www.statmethods.net/>
- Apprentissage en ligne du logiciel R.  
<https://www.datacamp.com/>
- Beginners statistics introduction.  
[http://influentialpoints.com/Training/basic\\_statistics\\_introduction.htm](http://influentialpoints.com/Training/basic_statistics_introduction.htm)
- Liste des bases de données disponibles sous R.  
<https://vincentarelbundock.github.io/Rdatasets/datasets.html>