
TP1 : Prise en main du logiciel RStudio

---

1 Introduction à R

Le logiciel **R** est un logiciel libre, gratuit et multiplateformes très répandu, tant dans les milieux académiques qu'industriels. Il est orienté vers l'analyse statistique et le traitement des données. De nombreuses méthodes modernes de traitement des données sont implémentées dans **R** et leur nombre est sans cesse augmenté par la communauté scientifique qui développe de multiples *packages*.

Nous travaillerons avec **RStudio** qui comporte une interface graphique et un jeu de fenêtres et menus qui rendent l'utilisation de **R** plus conviviale.

Si vous le souhaitez, vous pouvez installer **R** et **RStudio** sur votre ordinateur personnel. Pour télécharger **R** et **RStudio**, vous pouvez vous rendre sur le site `https://posit.co/download/rstudio-desktop/`.

Le but de ce premier TP est d'apprendre les bases de la manipulation de **R** (opérations élémentaires, sauvegarde de l'espace de travail et de données, importation d'un jeu de données). Au fur et à mesure des prochains TP, nous apprendrons la syntaxe et les principales commandes de **R** pour différents types d'analyses statistiques.

1.1 Environnement de travail

A) Organisation de l'espace disque

Il est important d'organiser son espace disque afin de pouvoir retrouver facilement ses fichiers.

↪ Pour les TP de *Statistique*, créer un répertoire `TpStat` dans le dossier `K:/Documents`. Vous y sauvegarderez tous les codes exécutés pendant les séances de TP, ainsi que les jeux de données sur lesquels nous travaillerons.

Dans **RStudio**, au début de chaque session, il faudra définir le répertoire par défaut dans lequel les fichiers seront enregistrés (le dossier que vous venez de créer, `K:/Documents/TpStat`). Pour cela, on utilise la commande : `setwd("K:/Documents/TpStat")` ou bien vous pouvez également passer par les menus : `Session` puis `Set Working Directory` puis `Choose Directory` et choisir le dossier `TpStat`.

B) Lancer RStudio

En salle de TP, aller sur `Démarrer/R` et lancer `RStudio`.

C) Ligne de commande

Après le lancement de **RStudio**, dans la fenêtre `Console`, l'utilisateur se retrouve face à un « prompt »(>) qui attend des commandes. On parle de logiciel fonctionnant en « ligne de commande »par opposition à un logiciel « clic-bouton ».

Le caractère # permet d'insérer un commentaire. Toute ligne précédée du caractère # n'est pas interprétée par le logiciel. Il est essentiel de commenter son code au maximum.

On peut naviguer dans l'historique des lignes de commande au moyen des touches « flèche haut »et « flèche bas ».

D) Ouvrir un script

Lorsque les commandes sont tapées directement dans la console, il est difficile de sauvegarder toutes les commandes tapées. On recommande donc de créer un script, c'est-à-dire un programme, dans lequel on tapera toutes les commandes à exécuter. Pour ouvrir un script, aller dans `File` puis `New File` puis `R Script`. Une fenêtre apparaît dans laquelle il faudra écrire les commandes à exécuter.

Exemple : copier dans le script les lignes suivantes :

```
2+2
4/2
x=c(2,5,9)
```

Pour exécuter une ligne de code dans la console, il faut suffit de placer le curseur sur cette ligne et cliquer sur l'icône `Run` dont le raccourci est `Ctrl + Entrée`. Il n'est pas nécessaire de sélectionner la ligne, sauf si on souhaite exécuter plusieurs lignes à la fois. Dans ce cas, il faut sélectionner toutes les lignes à exécuter avant de cliquer sur `Run`.

Il faut penser à sauvegarder régulièrement le script tout au long de chaque séance de travail puis à la fin de la séance. Le fichier sauvegardé porte l'extension `.R`.

Exemple : exécuter les commandes copiées dans le script. On observe que l'objet `x` a été créé dans l'espace de travail (fenêtre `Environment`) alors que les autres commandes ont simplement produit un résultat dans la console.

E) Sauvegarde de l'espace de travail

Pour sauvegarder les objets créés au cours d'une session, aller dans la fenêtre `Environment`, puis `Save workspace as`. Sauvegarder le fichier dans le dossier `TpStat`. Le fichier sauvegardé portera l'extension `.RData`. Lors de la prochaine séance, pour récupérer les objets que vous avez créés, aller dans la fenêtre `Workspace`, puis cliquer sur `Load Workspace` et choisir le fichier.

Comme pour le script, il est recommandé de sauvegarder l'espace de travail régulièrement, en particulier après la création de nouvelles variables ou avant le lancement d'un gros calcul.

Récapitulatif : à la fin de la séance, vous devrez donc avoir deux fichiers dans votre dossier `TpStat` :

1. un fichier au format `.R` contenant le script (vos lignes de commande)
2. un fichier au format `.RData` contenant les objets que vous avez créés (le résultat des lignes de commande)

F) Aide en ligne

Pour accéder à l'aide en ligne, on peut directement utiliser l'onglet `Help` de la fenêtre en bas à droite.

1.2 Opérations élémentaires

Attention, **R** est sensible à la casse, il faut donc bien distinguer les lettres majuscules et minuscules.

A) Logique par objet de R

Dans **R**, on manipule des objets. Les deux principaux types d'objets que nous allons utiliser sont les vecteurs et les `data.frames`.

- **Vecteur :** c'est l'objet de base dans **R**. La fonction `c()` permet de combiner les arguments pour créer un vecteur. Dans **R**, l'affectation des valeurs se fait avec `=` ou `<-`.
  Exemple : on crée deux vecteurs numériques `x` et `y` et on les affiche :

```
x = c(2,1,1,3,2)
x
y <- c(1:5)
y
```

  Exécuter les instructions suivantes et noter leurs effets :

```
length(x)
y[3]
y[-3]
```

- **Tableau de données** (`data.frame`) : c'est tableau à double entrée (individus en ligne et variables en colonne) pouvant contenir des éléments de différents types (c'est-à-dire les variables peuvent être quantitatives et/ou qualitatives). Ce sont les objets les plus utilisés en analyse de données.

  Exécuter les instructions suivantes et noter leurs effets :

```
F <- data.frame(x=rep(1,4),y=1:4,z=c(1,5,2,6))
F
F$x
x
names(F)
names(F) <- c("A","B","C")
F
row.names(F)
head(F,3)
tail(F,3)
F[2,]
F[2:4,]
F[c(2,4),]
F$B[2]
```

B) Instructions logiques

Afin de comparer deux objets, il existe différentes opérations logiques : `>`, `<`, `>=`, `<=`, `!=` (« différent de ») et `==` (« égal à », à ne pas confondre avec le symbole d'affectation `=`). Plusieurs instructions logiques peuvent être combinées grâce aux opérateurs : `&` (et), `|` (ou). Le résultat d'une instruction logique est soit `TRUE` (vrai) soit `FALSE` (faux).

Exécuter les instructions suivantes et noter leurs effets :

```
x<2
x==2
x!=2
x!=2 | y<=3
x==1 & y>2
which(x!=2)
x[x!=2]
F[F$B==2 & F$C==2,]
F[F$B==2 | F$C==2,]
```

C) Opérations sur un vecteur numérique

Les variables quantitatives sont représentées par des vecteurs de type *numeric* ou *integer* (entier).

Exécuter les instructions suivantes et noter leurs effets :

```
x+y
x-y
x*10
x/5
x^2
sqrt(x)
log(x)
sort(c(2,4,1))
```

Voici quelques premiers éléments sur les données manquantes, codées `NA` (pour *Not Available*) :

```
x[3]=NA;x
is.na(x)
any(is.na(x))
all(is.na(x))
```

Pour plus d'aide, taper `help(NA)`. Nous travaillerons sur les données manquantes ultérieurement.

D) Utilisation de packages

Les packages sont des ensembles de fonctions (et parfois de données) proposés par la communauté des utilisateurs de **R** et permettant d'accomplir des tâches spécifiques. L'utilisation d'une fonction incluse dans un package nécessite deux étapes préalables (réalisables via l'onglet `Packages` de la fenêtre en bas à droite de **RStudio**) :

- l'installation, avec la fonction `install.packages` (ou le bouton `Install` de la fenêtre `Packages`). Cette opération ne doit être accomplie qu'une seule fois.
- le chargement, avec la fonction `library` (ou en cochant la case du package dans la fenêtre `Packages`) pour charger le package et l'utiliser durant une session. *Cette opération est à renouveler à chaque lancement de* **R**.

Exemple avec le package `foreign` :

```
install.packages("foreign") # Installation du package
library(foreign) # Chargement du package
```

Il est recommandé de procéder régulièrement à une mise à jour des packages.

Sauvegarder l'environnement (fichier `.RData`) et le script (fichier `.R`) en leur donnant par exemple les noms *TP1.RData* et *TP1.R*.

2 Importer et exporter des données

2.1 Importer des données

Pour importer un fichier, une première solution est de cliquer sur le bouton *Import Dataset* et de choisir les options d'importation selon les caractéristiques de votre fichier.

En utilisant les fonctions, on distingue différentes façons d'importer un fichier de données dans **RStudio**, selon le type du fichier.

A) Données au format texte *(.txt)*

On va importer le fichier *employes.txt* se trouvant sous Moodle. On commence par le télécharger dans le répertoire *TpStat* créé précédemment. Pour cela, cliquer sur le fichier avec le bouton droit, choisir l'option *Enregistrer la cible sous* puis le répertoire *TpStat*. ***Ne pas ouvrir le fichier depuis Moodle !*** Dans **RStudio**, vider l'environnement grâce à l'icône balai.

Pour importer le fichier dans **RStudio**, on utilise la fonction `read.table` :
`employes=read.table("employes.txt",header=TRUE)`

Cette commande permet de créer l'objet *employes* qui est un `data.frame`. L'argument `header=TRUE` indique à **R** que le fichier à importer contient les noms des variables dans la première ligne.

Une fois le fichier importé, ne pas oublier de sauvegarder l'environnement (par exemple sous le nom `employes.RData`) et le script (fichier `TP1.R`).

Remarque : Si le fichier à importer ne se trouve pas dans le dossier défini par défaut pour la session (avec `setwd`), il faut préciser le chemin d'accès au répertoire dans lequel se trouve le fichier à importer, par exemple `employes=read.table("K:/Documents/employes.txt",header=TRUE)`

D'autres arguments de la fonction `read.table` peuvent être utiles dans l'importation de fichiers au format texte :

- `sep="\t"` qui précise que le séparateur (c'est-à-dire le caractère utilisé pour séparer les valeurs prises par les différentes variables) est la tabulation. Si les valeurs sont séparées par des points-virgules, on choisit : `sep=";"`.
- `dec="."` ou `dec=","` qui permet de spécifier la marque de décimale (le point ou la virgule).

B) Données au format texte où les valeurs sont séparées par une virgule *(.csv)*

Lorsque le fichier de données est un fichier texte où les différentes valeurs des variables sont séparées par une virgule (comma separated values, CSV), on utilise la fonction `read.csv` pour importer le fichier (voir l'aide de la fonction). Par exemple, si on dispose d'un fichier *fichier.csv* à importer dans le `data.frame` *nom_fichier*, on utilise la commande suivante :
`nom_fichier = read.csv("fichier.csv", sep=',', header=TRUE)`

C) Données dans d'autres formats

Le package `foreign` permet d'importer des fichiers d'autres types : SPSS (.sav), SAS (.sas7bdat), Stata (.dta), etc. Par exemple, si on dispose d'un fichier *fichier.sav* (fichier SPSS) à importer dans le `data.frame` *nom_fichier*, on utilise la commande suivante (après avoir chargé le package `foreign`) :
`nom_fichier = read.spss("fichier.sav",to.data.frame=TRUE)`

2.2 Exporter des données

Pour exporter des données depuis **R**, on peut utiliser la fonction `write.table`. Par exemple, pour exporter le `data.frame` appelé *donnees* dans le fichier texte *nom_fichier.txt*, on utilise la commande suivante : `write.table(donnees,file="nom_fichier.txt")`