Université de Montpellier
2018/2019

HLMA408: Traitement de données
Emmanuel Bonnet, Joseph Salmon

TP n° 1 : Prise en main de RStudio

Merci de lire la fiche de TP en détail. Avant d'appeler le responsable du TP, vérifier que vous avez bien lu la fiche en détail, et chercher dans les pages d'aide de R.

Les points à faire, ainsi que les questions attendant une réponse, sont marqués d'un triangle rouge ▸.

1 Présentation de RStudio

Vous pouvez démarrer R tout seul sans fenêtre graphique.

RStudio est un environnement de développement pour R que nous allons utiliser. Il permet de travailler plus efficacement avec R que la simple ligne de commande dans une console, en utilisant différentes informations rangées dans des fenêtres graphiques.

▸ Testons R seul comme suit.

— Ouvrir une console (qui s'appelle encore terminal). Taper R. (Puis Entrée)

— Le logiciel R attend alors vos commandes.

— Taper par exemple 3 + 5 (puis entrée). Il vous répond...

— Essayer un second calcul simple.

— Pour quitter, tapez q() et répondez n.

— Fermer le terminal.

▸ Démarrer RStudio depuis les menus GNU/Linux.

1.1 Agencement de RStudio

— En bas à gauche, la "console". Elle attend vos commandes R après le prompt >. Comme dans le cas précédent, vous pouvez lui demander ce que vaut 3+5.

— En haut à gauche, la fenêtre "éditeur". Chaque fichier est dans un onglet. Cela permet de taper des commandes R, de les enregistrer dans des fichiers (appelés fichiers de script) et de les retrouver ensuite. Vous pouvez ensuite les envoyer dans la console R, voir plus loin.

— En haut à droite, le workspace qui liste les données dans la mémoire de R, et un onglet "History" où vous voyez l'historique des commandes que vous avez envoyées à R.

— En bas à droite, dans quatre onglets,

- le système de fichiers (Files)
- les graphiques produits par R (Plots)
- les bibliothèques de commandes disponibles (Packages), que l'on peut activer ou désactiver ici
- les pages d'aide (Help)

1.2 Répertoire de travail

Le répertoire de travail (ou "Working directory") est l'endroit du le système de fichiers de l'ordinateur où R lit et écrit des informations par défaut.

Lorsque vous démarrez un TP sous R, pensez à choisir le bon répertoire de travail. Pour cela, on peut utiliser la commande "Set as working directory" du menu "More" de l'onglet "Files" après s'être placé dans le bon répertoire. La commande `setwd` fait la même chose. Par exemple `setwd("~/HLMA408/TP1")` permet de choisir le sous-répertoire TP1 du répertoire HLMA408 de votre compte sous Linux, noté ~. Une dernière méthode consiste à utiliser la commande "Set Working Directory" qui est dans le menu

"Session" ou dans le menu "Tools" suivant les versions de RStudio. Enfin, la commande `getwd()` permet de savoir quel est le répertoire de travail.

Pour chaque fiche de TP, nous utiliserons un script R. Je vous conseille de créer un sous-répertoire de HLMA408 pour chaque fiche de TP. Dans chacun des ces sous-répertoires, vous pourrez stocker :

— le sujet de TP au format PDF,

— les fichiers de données,

— le ou les scripts R qui contiennent les commandes que vous aurez exécutées au cours du TP,

— les fichiers de graphiques que vous aurez enregistrés (au format png ou pdf).

Enfin, notez que tout ce qui suit un # dans un fichier script de R est interprété comme un commentaire.

▸ Avec la commande `getwd()`, trouvez quel est le répertoire de travail.
▸ À l'aide de l'onglet "Files",

— Créer un répertoire s'appelant `HLMA408` dans votre compte.

— Entrer dans ce répertoire.

— Créer un sous-répertoire de ce dernier s'appelant `TP1`.

— Entrer dans ce sous-répertoire.

— Dans le menu "More", choisissez "Set as working directory".

Ainsi, `~/HLMA408/TP1` est désormais le répertoire de travail de R.

▸ Vérifier le résultat à l'aide de la commande `getwd()` et essayez aussi la fonction `setwd` dans la console R !

▸ Créer un script utilisant la commande "New R script" du menu "File".
L'enregistrer dans un fichier appelé `TP1.R` dans le répertoire `~/HLMA408/TP1`
Ajouter une première commande en haut du fichier pour fixer le répertoire de travail à `~/HLMA408/TP1`

▸ Ajouter deux premières lignes au script R qui commencent par un # chacune. Dans la première, taper un titre. Dans la seconde, taper votre nom. Exemple :

```
# HLMA 408 TP 1 : Mon premier script R
# Prénom Nom
```

On trouvera les données du TP ici :
http://josephsalmon.eu/enseignement/datasets/babies23.data avec leur readme : http://josephsalmon.eu/enseignement/datasets/babies.readme.txt. Enfin un exemple de fichier R pour démarrer est disponible ici :
http://josephsalmon.eu/enseignement/Montpellier/HLMA408/exemple_session.R

1.3 Bibliothèques (*libraries*)

R est capable de faire de nombreuses analyses statistiques. Les différentes fonctions (ainsi que des jeux de données exemples) sont rangées dans des bibliothèques. Il faut charger ou décharger ces bibliothèques pour utiliser leurs fonctions ou leurs jeux de données.

Exemple. Nous allons utiliser la bibliothèque `MASS` qui contient des fonctions utiles et de nombreux jeux de données. Vous avez deux possibilités pour charger la bibliothèque :

— Soit dans l'onglet "Packages" vous cochez la ligne MASS

— Soit dans la console, vous tapez `library(MASS)`

Et pour la décharger :

— Soit dans l'onglet "Packages", vous décochez la ligne MASS

— Soit dans la console, vous tapez `detach(package:MASS)`

De nombreuses bibliothèques sont disponibles sur internet...

▸ Ajouter au script `TP1.R` une commande pour charger la bibliothèque MASS.

1.4 Scripts R et premier exemple de session

Nous avons déjà créer un premier script R. Rappelons que tout ce qui suit un # dans la ligne est du commentaire. Vous pouvez ensuite envoyer les commandes de ces scripts dans la console R à l'aide des boutons :

- "Run" pour envoyer la ligne où se trouve le curseur, ou toute la sélection si une partie du script est sélectionnée.
- "Source" pour envoyer tout le script à R.

▸ Télécharger le fichier `exemple_session.R` sur l'espace pédagogique et l'enregistrer dans votre répertoire `TP1` L'ouvrir avec RStudio et executer le code ligne à ligne à l'aide du bouton "Run" tout en lisant en détail les commentaires pour comprendre ce qu'il se passe.

▸ Bien prendre le temps de comprendre les résultats. Constater à la fin dans l'onglet "Plots" que vous pouvez naviguer entre l'affichage des différents graphiques à l'aide des flèches bleues.

▸ Si vous êtes complètement perdu, lire le document « R pour les débutants » d'Emmanuel Paradis que vous pouvez télécharger à l'adresse :
http://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf
et lire les sections 2 et 3.1.

▸ Une fois que vous avez fini de travailler avec le script d'exemple, vous pouvez effacer le contenu de la mémoire en cliquant sur le balai dans l'onglet "Workspace". Vous pouvez aussi fermer le fichier `exemple_session.R`, nous allons travailler sur d'autres données.

2 Importer et préparer des données

2.1 La fonction `read.table`

La fonction `read.table` permet d'importer des données enregistrées dans un fichier texte. Attention, un fichier texte n'est pas un fichier créé avec un traitement de texte. (Un fichier texte ne contient que des caractères à la suite des autres, mais pas de mise en forme du type gras, italique, taille de police, etc.)

La fonction `read.table` renvoie un objet de type `data.frame` qu'il faut assigner à l'aide de `<-` dans un objet. La commande type pour importer une table est

```
MettreIciUnNom <- read.table("chemin du fichier", pleins d'options)
```

La fonction `read.table` a de très nombreuses options pour expliquer à R comment sont organisées les données dans le fichier texte. Par exemple, l'option `header = TRUE` permet de dire à R que les noms des variables sont sur la première ligne du fichier. Je vous renvoie à la page d'aide de R et la section 3.2 du document d'E. Paradis pour plus de détail.

▸ Télécharger sur l'espace pédagogiques les fichiers `babies23.data` et `babies.readme.txt` et les enregistrer dans le répertoire du TP1.

▸ Lire le fichier `babies.readme.txt` pour comprendre la signification des différentes variables.

▸ Utiliser la fonction `read.table` pour créer un objet appelés `babies` dans lequel on importe les données du fichier `babies23.data`

▸ Ajouter cette commande au fichier de script `TP1.R` que l'on avait commencé de remplir dans la section 1.

▸ Afficher le tableau de données `babies` dans RStudio. Dans le fichier texte, il y avait deux colonnes (c'est-à-dire deux variables) qui s'appelaient `wt`. Comment R a résolu le conflit ?

2.2 Comment acceder aux lignes et aux colonnes de la table de donnees

Une table de données est un objet de type `data.frame` en R. Les différentes variables sont rangées dans des colonnes alors que les lignes correspondantes aux différentes observations. Les lignes et les colonnes sont numérotées à partir de 1. Elles peuvent avoir des noms.

On peut facilement extraire une sous-table de données ou bien un vecteur colonne depuis une table de données. Ainsi,

```
NomDeLaTable$NomDeLaColonne
```

permet d'obtenir la colonne dont le nom est `NomDeLaColonne` de la table dont le nom est `NomDeLaTable`. Cela renvoie un vecteur. (Attention, R est sensible à la différence majuscule/minuscule!)

On peut aussi utiliser les crochets `[ ]` pour extraire un nombre, un vecteur ou une partie de la table. La convention est `[ ligne, colonne ]`. Le mieux est de donner des exemples.

Supposons que l'on dispose d'une table de données (`data.frame`) qui s'appelle `hills` et qui a les propriétés suivantes :

— elle est formée de 35 lignes et 3 colonnes,

— les colonnes numérotées 1, 2 et 3 s'appellent respectivement `dist`, `climb` et `time`.

Alors

— `hills$climb` renvoie la colonne `climb` de la table hills sous forme de vecteur

— `hills[6, 2]` renvoie le nombre dans la ligne 6 et la colonne 2

— `hills[, 2]` renvoie la deuxième colonne
Le numéro de ligne n'est pas spécifié, il conserve donc toutes les lignes. Le résultat est un objet à un seul indice (qui correspond au numéro de ligne), donc un vecteur.

— `hills[6,]` renvoie la sixième ligne

— `hills[1:4, 2]` renvoie les nombres dans les lignes de 1 à 4 et sur la seconde colonne

— `hills[6, 1:2]` renvoie les nombres sur la ligne 6, dans les colonnes 1 et 2.

— `hills[1:4, 1:2]` renvoie une sous-table ne contenant que les lignes de 1 à 4 et les colonnes de 1 à 2.

- Essayer ces exemples sur la table de données `hills` de la bibliothèque MASS.
- Quelle est la valeurs dans la colonne 7, ligne 12 de la table `babies` ?
- Comment extrait-on la sous-table formée des lignes de 3 à 12 et des colonnes de 2 à 6 de la table `babies` ?
- Ajouter les commandes qui concernent la table `babies` au script `TP1.R`. Mettre une ligne de commentaire au dessus de chaque commande qui décrit succinctement sont objectif.

On peut aussi récupérer des colonnes ou des lignes dont les numéros ne se suivent pas. Voir les fiches suivantes de TP.

2.3 Mettre en forme les colonnes du jeu de données

Les commandes de R s'adaptent au type d'objet sur lesquels on les applique. Il est donc fondamental de mettre la table de données en forme correctement. Pour cela, il faut particulièrement :

— re-coder les valeurs manquantes avec le code spécial `NA`,

— préciser quelles sont les colonnes qui représentent des variables discrètes (ou facteurs).

La fonction `replace` est particulièrement efficace pour remplacer certaines valeurs par une autre dans un vecteur. Son prototype est

```
replace(UnVecteur, UneConditionSurLeVecteur, UneValeur)
```

Attention, la fonction `replace` ne modifie pas le vecteur que l'on passe comme premier paramètre. Mais il retourne une version modifiée. Si l'on veut que la version modifiée remplace la précédente, il faut faire quelque chose du genre

```
UnVecteur <- replace(UnVecteur, UneConditionSurLeVecteur, UneValeur)
```

Par exemple,

— `fume <-replace(fume, fume == 9, NA)` permet de remplacer les "9" par des `NA` dans le vecteur `fume`. **Attention, le test d'égalité se fait avec un double signe égal, comme ça : ==**

— `dates <-replace(dates, dates <0, NA)` permet de remplacer toutes les valeurs strictement négatives du vecteur dates par des `NA`.

— `jojo <-replace(jojo, jojo >= 100, 100)` permet de remplacer toutes les valeurs de `jojo` supérieures à 100 par 100.

Il existe de nombreux types de vecteurs en R. Les coordonnées du vecteur peuvent être :

— des nombres entiers ou réels → "numeric"

— des chaînes de caractères → "character"

— des modalités d'une variable discrète qui n'ont pas d'ordre → "factor"

— des modalités d'une variable discrète, en spécifiant l'ordre de chacune des modalités → "ordered"

Lorsque l'on importe des données avec la commande `read.table`, le logiciel R essaie de détecter le type de chacune des colonnes. Mais il ne devine jamais quelles sont les variables discrètes. Il faut donc les re-coder correctement. Les fonctions `as.numeric`, `as.character`, `factor` et `ordered` permette de convertir un vecteur dans l'un des quatre types décrits ci-dessus (suivant le nom de la fonction).

Exemples. Le vecteur `x` construit avec la commande `x <- c("a", "b", "c", "a")` est de type character. Si l'on veut dire à R qu'il s'agit de 4 observations d'une variables qui peut prendre les valeurs "a" ou "b" ou "c", il faut écrire

```
x <- factor(x)
```

et si l'on veut préciser l'ordre des trois modalités : "a" > "b" > "c", il faut créer un vecteur de type ordered et préciser à l'aide de l'option `levels=...` quel est sont les modalités de la plus petite à la plus grande :

```
x <- ordered(x, levels=c("c", "b", "a"))
```

Ainsi, pour re-coder correctement la colonne `smoke` de la table `babies` déjà importée il faut utiliser :

```
babies$smoke <- replace(babies$smoke, babies$smoke == 9, NA)
babies$smoke <- ordered(babies$smoke)
```

- Ajouter les deux lignes ci-dessus au script R qui s'appelle `TP1.R` et les faire tourner.
- Re-coder correctement la variable `sex` de la table `babies`. Ajouter ces commandes au script R.
- Même chose pour les variables `race`, `ed` et `marital`.

Il faudrait le faire pour toutes les variables, mais nous nous limiterons ici pour ne pas que le TP dure trop longtemps. Dans tous les cas, on se garde de re-coder correctement la variable `ded` qui serviva d'exemple de variable mal codée par la suite.

3 Analyse descriptive

3.1 Quelques fonctions pour resumer des donnees

Des commandes pour obtenir des résultats numériques

Commande	Description succincte
`summary`	Calcule des statistiques résumées sur un vecteur d'observations (s'adapte aux vecteurs "numeric", "factor", ...) ou sur une table de données, colonne par colonne
`mean`	Calcule la moyenne d'un vecteur
`sd`	Calcule l'écart-type d'un vecteur
`median`	Calcule la médiane d'un vecteur
`quantile`	Calcule les quantiles d'un vecteur d'observations.

Utiliser les pages d'aide de R pour comprendre plus en détail chacune de ces commandes.

- Que renvoie la commande `summary(babies$wt)` ?
- Quelle est la différence entre les résultats des deux commandes ci-dessous ? Pourquoi ?

```
summary(babies$ed)
summary(babies$ded)
```

- Que renvoient les commandes ci-dessous ?

```r
quantile(babies$wt)
quantile(babies$wt, probs = c(0.1, 0.3, 0.7, 0.9))
quantile(babies$wt, probs = seq(from=0.05, to=0.95, by=0.05))
```

3.2 Quelques fonctions graphiques

Des commandes pour obtenir des graphiques

Commande	Description succincte
`plot`	Permet de faire des nuages de points, des courbes en donnant une liste de points serrés par lesquels passe la courbe, des diagramme en barres, etc.
`hist`	Permet de faire des histogrammes en densité d'effectifs ou densité de fréquence
`kde`	Permet de calculer un estimateur à noyau de la densité dans la bibliothèque `"ks"` et d'en tracer par la suite le résultats avec la fonction `plot()`
`vioplot`	Permet de tracer un ou plusieurs diagrammes en violon en parallèle dans la bibliothèque `"vioplot"`
`pie`	Permet de tracer un diagramme en secteurs
`qqnorm`	Permet de tracer un diagramme quantile-quantile de comparaison de la distribution des observations d'un vecteur à la distribution de la loi normale
`qqline`	Permet d'ajouter la meilleure droite sur un graphique obtenu avec `qqnorm`
`abline`	Permet d'ajouter à un graphique ouvert le dessin d'une droite en donnant sa pente et son ordonnée à l'origine
`points`	Permet d'ajouter à un graphique ouvert des points
`lines`	Permet d'ajouter à un graphique ouvert une courbe donnée par une suite de points...
`legend`	Permet d'ajouter une légende à un graphique ouvert
`title`	Permet d'ajouter un titre à un graphique ouvert

- Ajouter toutes les commandes ci-dessous au script `T1.R` avant de les faire tourner.
- Comprendre ce que dessine les commandes suivantes :

```r
plot(babies$wt)    plot(babies$wt.1)    plot(babies$dwt)
plot(babies$smoke)
```

- Faire un histogramme de la variable `wt` à l'aide de la commande `hist` en passant l'option `freq = FALSE`
- Que font les deux commandes ci-dessous ?

```r
install.packages("ks")
library(ks)
fhat <- kde(babies$wt[babies$smoke == 0])
plot(fhat,col="green",
     xlab="Poids (en onces)", main="Estimation de la densité chez les non fumeuses")

fhat2 <- kde(babies$wt[babies$smoke == 0])
plot(fhat2,col="red",
     xlab="Poids (en onces)", main="Estimation de la densité chez les fumeuses")


plot(fhat,col="green",
     xlab="Poids (en onces)", main="Estimation de la densité")
plot(fhat2,col="red",add=T)
```

- Que font les deux commandes ci-dessous ?

```r
qqnorm(babies$wt[babies$smoke == 0])
qqline(babies$wt[babies$smoke == 1], col=2, lty=2)
```

- Que faut-il changer pour obtenir une ligne de couleur différente ? et en trait plein ?
- Que font les commandes ci-dessous ?

```
install.packages("vioplot")
library(vioplot)
vioplot(babies$wt[babies$smoke == 0])
vioplot(babies$wt[babies$smoke == 1])
vioplot(babies$wt[babies$smoke == 0],
       babies$wt[babies$smoke == 1],names = c("Non fumeuses", "Fumeuses"))
title("Comparaison de distribution des poids à la naissance")
```

Enregistrer le dernier graphique dans un PDF.

- Avec la fonction `pie`, vous pouvez représenter la variable `smoke` ainsi :

```
pie(table(babies$smoke))
```

Remarquez qu'il faut d'abord appliquer la fonction `table` qui calcule les effectifs de chacune modalité dans le vecteur `babies$smoke`.

- Que font les commandes ci-dessous ?

```
ed.labels <- c("less than 8th grade",
              "8th -12th grade - did not graduate",
              "HS graduate--no other schooling", "HS+trade",
              "HS+some college", "College graduate",
              "Trade school HS unclear", "Trade school HS unclear")
pie(table(babies$ed), labels=ed.labels)
title("Mother's education level")
```

Enregistrer ce graphique dans un PDF.

- Et que pensez-vous du graphique obtenu avec la commande ci-dessous ?

```
plot(table(babies$ed), type = "h", ylab="Counts")
```

4 Quitter RStudio

- Quitter RStudio en choisissant la commande "Quit RStudio du menu "File" et répondre négativement à la question posée.