TP 1 : démarrer avec R

Nathan Uyttendaele, Johan Segers et Antoine Soetewey

LINGE1222 Analyse statistique multivariée

1 C'est quoi, "R" ?

R n'est rien d'autre qu'un langage de programmation. C'est, au moment d'écrire ce document, le langage *de facto* en statistique, bien que pas le seul langage de programmation employé par les statisticiens.

Pour faire du R, nous avons besoin de deux choses :

- un éditeur de texte dans lequel rédiger notre code
- un endroit où exécuter ce code

Comme éditeur de texte, nous recommandons **Notepad++** sous Windows et **Sublime Text** sous Mac (les deux éditeurs sont gratuits). Ensuite, installez le logiciel gratuit fourni par le **CRAN** (Comprehensive R Archive Network) qui servira à exécuter votre code par copier-coller depuis l'éditeur texte. Lien :

https://www.freestatistics.org/cran/

Si vous utilisez un ordinateur dans une salle informatique de Louvain-la-Neuve, notez que le logiciel fourni par le CRAN, sobrement appelé R, sera normalement déjà installé sur ce dernier. Il n'y a par contre pas de garantie que l'éditeur de texte Notepad++ soit installé (dans le pire des cas, on peut utiliser Notepad qui est, lui, toujours installé).

Dans le cadre du cours LINGE1222 depuis 2018, nous faisons toutefois du R en utilisant la version gratuite du très populaire logiciel **R Studio**. Ce logiciel a l'avantage d'offir à la fois un éditeur de texte performant pour rédiger son code et un endroit où exécuter le code rédigé dans cet éditeur. Important : R Studio nécessite l'installation préalable du logiciel R fourni par le CRAN afin de pouvoir correctement fonctionner. Juste installer R Studio sur votre ordinateur personnel est insuffisant.

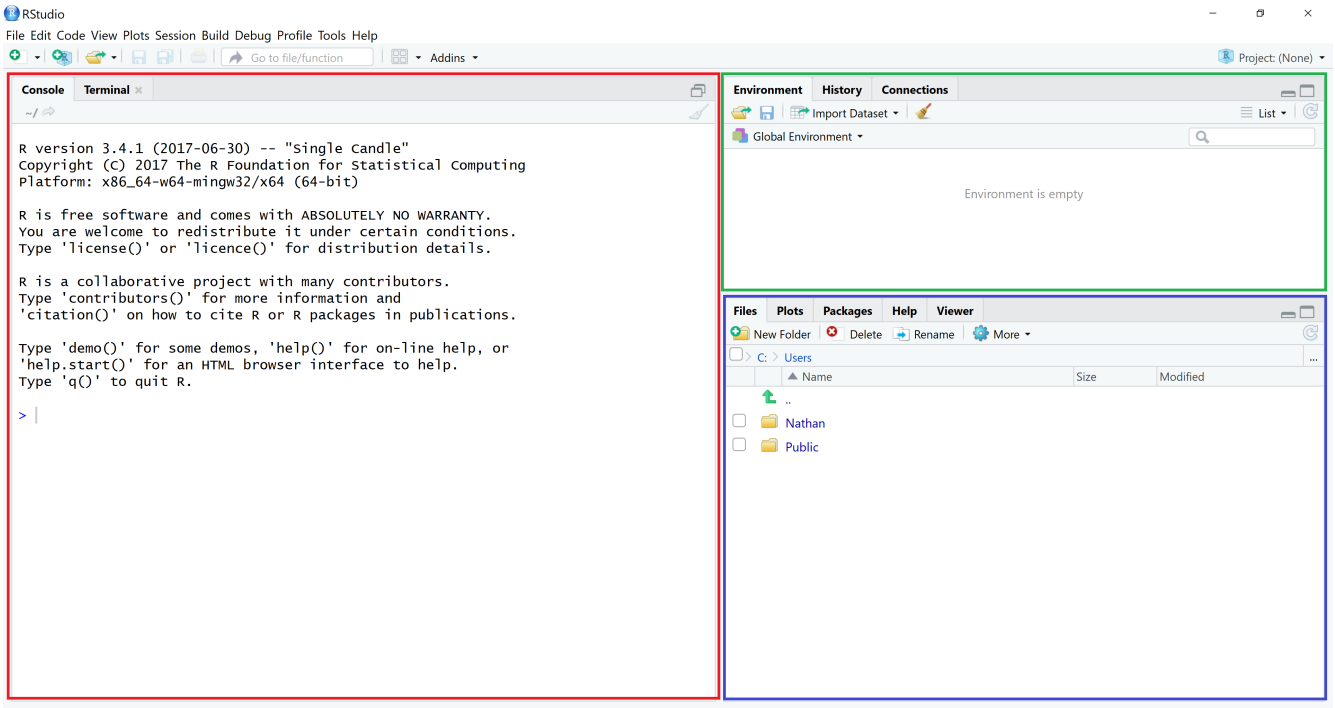
Si vous utilisez un ordinateur dans une salle informatique de Louvain-la-Neuve, notez que R Studio sera normalement déjà installé sur ce dernier, il vous suffit d'ouvrir le programme pour commencer à travailler. Si vous utilisez un ordinateur personnel, rendez-vous sur

https://www.rstudio.com

R Studio est disponible sous Windows, MAC et Linux.

2 R Studio

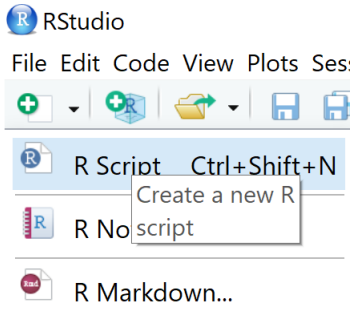
Par défaut, la fenêtre R Studio comporte trois sous-fenêtres, indiquées en rouge, vert et bleu dans la capture d'écran ci-après.



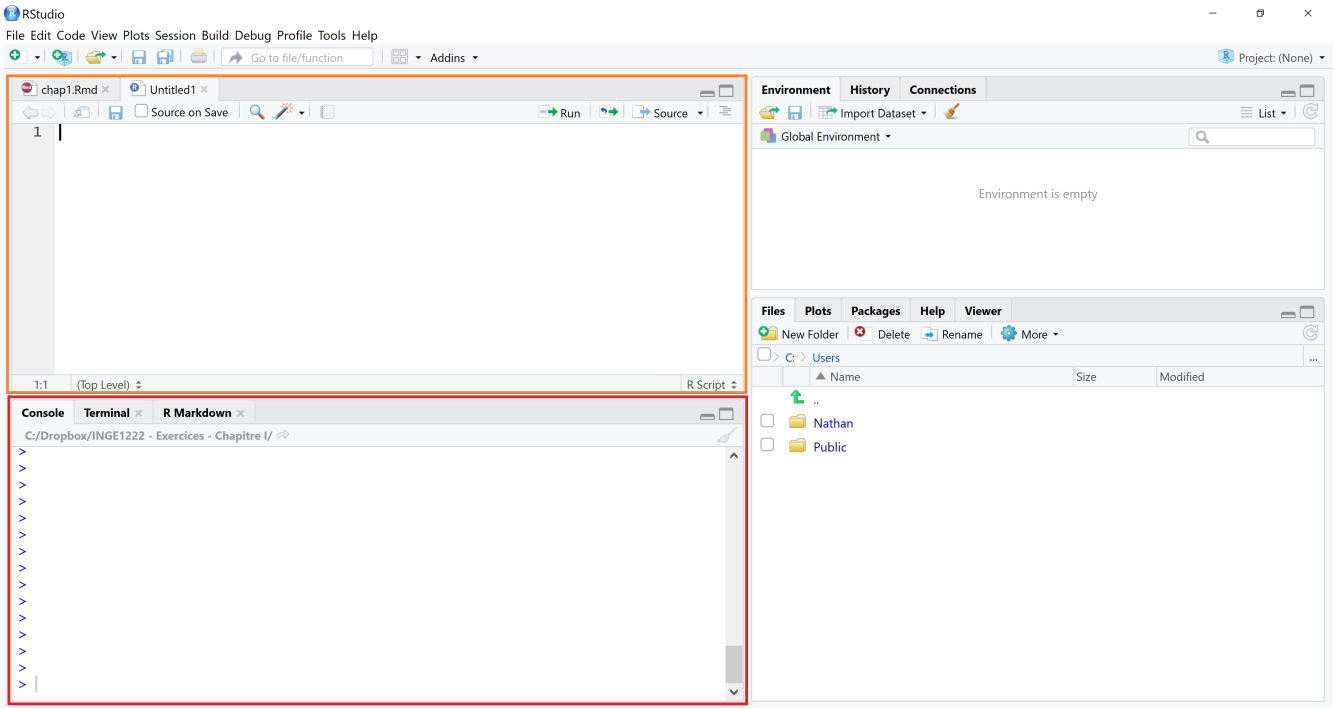
La fenêtre en rouge est de loin la plus importante : il s'agit de l'endroit où vous pourrez exécuter du code R.

Par défaut l'éditeur de texte fourni avec R Studio ne s'ouvre pas de façon systématique. Pour l'ouvrir, faire

`File > New File > R Script` ou cliquer sur le bouton marqué d'une petite croix verte dans le coin supérieur gauche, ensuite sur `R Script` (voir image ci-après).



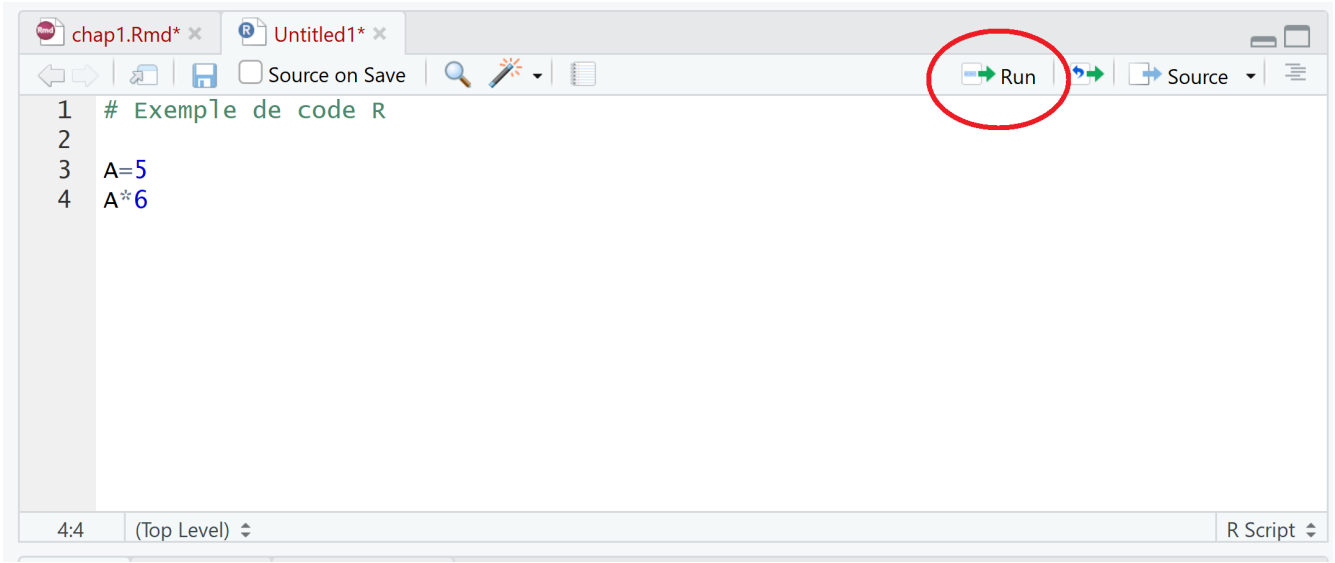
Le résultat est une nouvelle sous-fenêtre (en orange dans la capture d'écran ci-après) dans laquelle vous allez pouvoir rédiger votre code. Le code sera exécuté dans la sous-fenêtre en rouge.



La grosse différence entre l'éditeur de texte (en orange) et la console (en rouge) est que le code écrit dans l'éditeur de texte peut être sauvé et à nouveau exécuté ultérieurement. La fenêtre en rouge constitue un historique des codes qui ont été exécutés ainsi que les résultats de ces codes.

Pour exécuter du code rédigé dans l'éditeur texte (fenêtre orange), deux options :

- Vous tapez votre code et vous appuyez ensuite sur le bouton "run" ou vous utilisez le raccourci clavier CTRL+Enter. La ligne de code où se trouve votre curseur sera alors exécutée.
- Vous tapez votre code et sélectionnez dans l'éditeur texte la partie que vous souhaitez exécuter et vous appuyez ensuite sur le bouton "run" ou utilisez le raccourci clavier CTRL+Enter. Tout le code sélectionné sera exécuté.



Il est également possible de rédiger une ligne de code directement dans la console ; appuyer sur Enter exécutera cette ligne de code.

3 Quelques exemples de code

Le présent document vise à vous aider à démarrer votre apprentissage dans R. Mais il ne constitue absolument pas une introduction qui s'auto-suffit.

R est un langage utilisé par des centaines de milliers de personnes sur la planète et de nombreux tutoriaux, guides et ouvrages introductifs existent en ligne.

Soyez débrouillard : demandez de l'aide à Google dès que vous êtes coincé et rappelez-vous qu'une recherche en anglais donne considérablement plus de résultats qu'une recherche en français. Des liens utiles seront aussi placés sur la page Moodle du cours LINGE1222.

Ci-après, quelques exemples de code (sur fond gris) et le résultat (quand il y en a un) une fois ce code exécuté.

Addition de deux nombres :

```
1+1
```

```
[1] 2
```

Multiplication de deux nombres :

```
5*5
```

```
[1] 25
```

Calcul de $\frac{1}{\sqrt{50\pi}}\, e^{-\frac{(10-11)^2}{50}}$:

```
1/sqrt(50*pi)*exp(-(10-11)^2/50)
```

```
[1] 0.07820854
```

Stocker des valeurs :

```
A=5
B=6
```

Note : quand vous exécutez un code qui ne fait que stocker des valeurs quelque part, il n'y a typiquement pas de résultats à l'écran après exécution du code. Les valeurs ont été stockées, point.

Note 2: Pour stocker des valeurs, vous pouvez utiliser le signe "=" comme ci-dessus, ou bien le signe "<-". Ecrire A = 5 et A <- 5 revient exactement à la même chose.

Multiplier le contenu de l'objet `A` avec le contenu de l'objet `B` :

```
A*B
```

```
[1] 30
```

Stocker 3 valeurs dans un objet `A`. Notez que l'objet `A` ainsi créé est considéré comme un vecteur vertical (3×1) et peut être utilisé pour faire du calcul matriciel (exemple plus loin).

```
A=c(1/2, -1, 0)
```

Créér une matrice carrée 3×3 et la stocker dans un objet `ma_matrice` (vous pouvez donnez le nom que vous souhaitez aux objets que vous créez dans R !) :

```
ma_matrice=matrix(c(-1,2,0,0,0,0,5,2,0),nrow=3,ncol=3)
```

Afficher le contenu de l'objet `ma_matrice`, fraîchement créée :

```
ma_matrice
```

```
     [,1] [,2] [,3]
[1,]   -1    0    5
[2,]    2    0    2
[3,]    0    0    0
```

Obtenir une fiche d'aide sur la fonction `matrix()` qui permet de créer des matrices :

```
?matrix
```

Note : l'exécution de cette ligne de code ouvre une fiche d'aide dans la sous-fenêtre en bas à droite de R Studio. Ci-après, une capture d'écran de ce que ça donne pour la fonction `matrix()`.

Faire le produit matriciel de la version transposée de `A` avec la matrice `ma_matrice` (3×3) :

```
t(A)%*%ma_matrice
```

```
     [,1] [,2] [,3]
[1,] -2.5    0  0.5
```

On voit que le résultat est un objet matriciel de dimension (1x3), ce qui est bien la

dimension attendue.

Inverser une matrice :

```
solve(ma_matrice)
```

Note : toutes les matrices ne sont pas inversibles. ma_matrice ne l'est pas. Le code ci-dessus va donc générer une erreur si vous l'exécutez.

Générer 10 observations aléatoires depuis une loi normale $\mathcal{N}(0, 1)$:

```
rnorm(10)
```

```
 [1] -1.04617620 -1.59075354 -1.69821856 -0.18762191  0.02969810
 [6]  0.06554792 -0.26652168  0.21331694  2.37235268 -0.41988629
```

Générer 10 nouvelles observations aléatoires depuis une loi normale $\mathcal{N}(0, 1)$ et stocker ça dans l'objet `A` :

```
A=rnorm(10)
```

Multiplier chacune de ces observations aléatoires par 10 :

```
A*10
```

```
 [1]   5.101339 -14.105844   8.648019 -16.254305 -12.151883  19.501926
 [7] -20.791170   5.739425  -6.293783  -7.561860
```

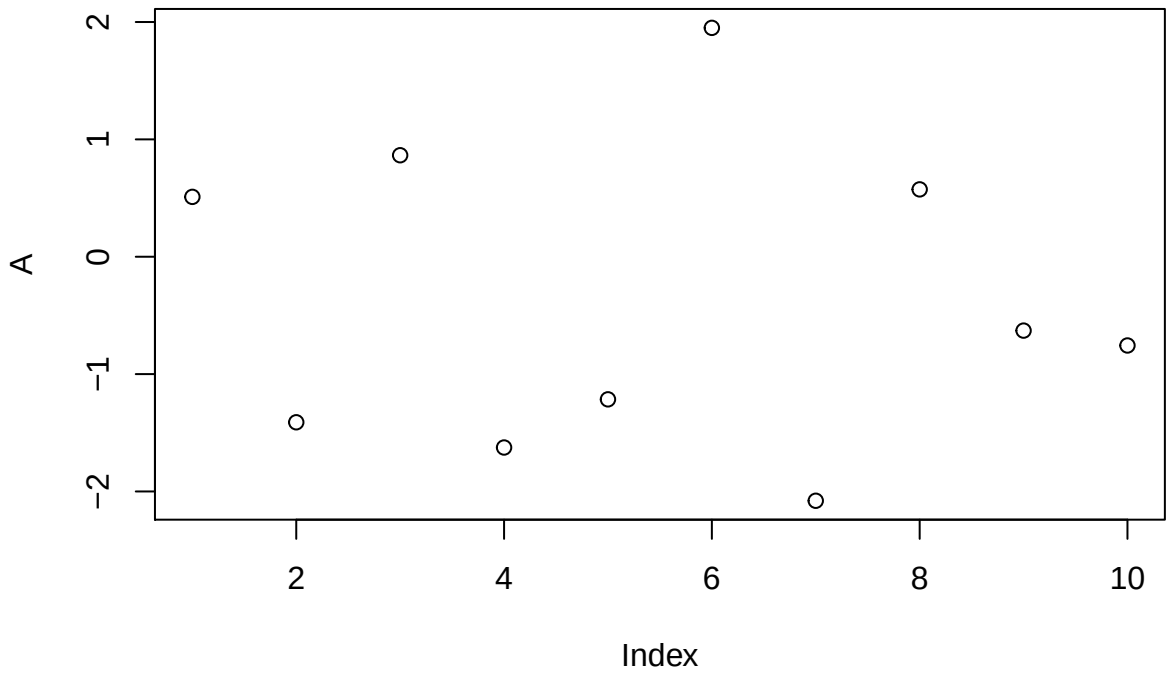
Obtenir une fiche d'aide sur la fonction `rnorm()` qui permet de générer des valeurs aléatoires depuis une loi normale $\mathcal{N}(0, 1)$:

```
?rnorm
```

(Ce bout de code affiche une fiche d'aide dans la sous-fenêtre en bas à droite dans R Studio.)

Tracer un graphique des 10 valeurs stockées dans l'objet `A` :

```
plot(A)
```

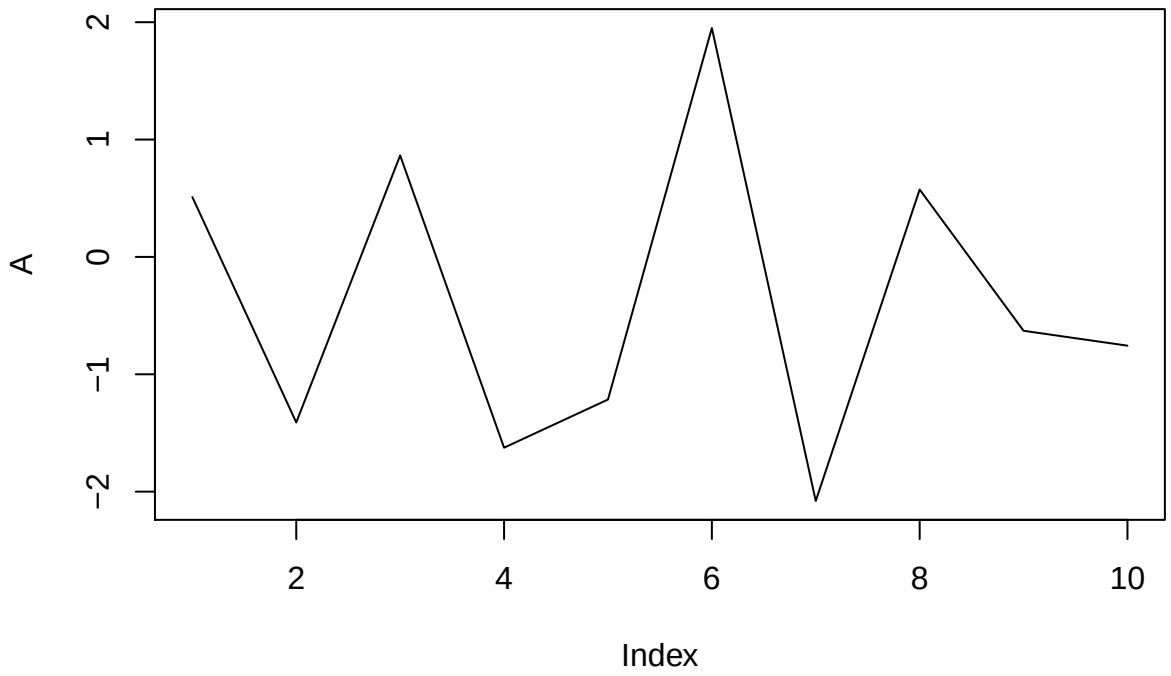


Obtenir une fiche d'aide sur la fonction `plot()` qui permet de tracer des graphiques :

```
?plot
```

On peut également modifier le graphique via les arguments de la fonction `plot()`. (Ces arguments sont détaillés dans la fiche d'aide.)

```
plot(A, type = "l")
```



4 Exportation et importation de données

Certainement l'une des opérations importantes à maîtriser est l'import de données depuis votre disque dur afin de les manipuler via le langage R.

L'importation de données reste une opération délicate car il existe de très nombreuses façons de stocker des données sur un disque dur : sous forme d'un fichier Excel, sous forme textuelle, sous forme d'un fichier SPSS, d'un fichier Stata, etc.

La manière d'importer des données n'est donc pas toujours la même car elle dépend de comment (format) ces données sont stockées sur le disque dur.

Attention : magré la variété des formats, il existe toutefois une convention universelle. Les colonnes d'un jeu de données doivent représenter des **variables** et les lignes des **individus**.

Un petit exemple d'import/export est donné ci-après, mais gardez à l'esprit que des tutoriaux détaillés existent en ligne. Une recherche dans Google avec les mots **"How to import data in R"** devrait donner pas mal de ressources.

Commençons par artificiellement créer un jeu de données comportant 3 individus (lignes) et deux variables (colonnes).

```
a=c(45, 60.48, -78.14)
b=c(178.001, 179.45, 158.2)
mes_donnees=data.frame(variable1=a, variable2=b)
```

Comme on ne fait que stocker des valeurs dans l'objet `mes_donnees`, l'exécution de cette ligne de code ne donne aucun résultat particulier à l'écran (voir aussi les exemples donnés plus haut dans le document).

Si on veut visualiser le contenu de l'objet `mes_donnees`, il faut explicitement appeler cet objet après création en exécutant la ligne de code suivante :

```
mes_donnees
```

```
  variable1 variable2
1     45.00   178.001
2     60.48   179.450
3    -78.14   158.200
```

Exportons à présent le contenu de l'objet `mes_donnees` sur le disque dur sous forme d'un fichier texte. La première chose à faire est de décider OÙ sur votre disque dur vous souhaitez exporter le contenu de l'objet `mes_donnees`.

A tout instant, il y a dans R ce qu'on nomme un "dossier de travail". Pour connaître votre dossier de travail, exécutez la ligne de code

```
getwd() #wd étant l'abbréviation de "working directory"
```

(Notez que tout ce qui se trouve après un # sera considéré comme un commentaire et ne sera donc pas executé comme du code.)

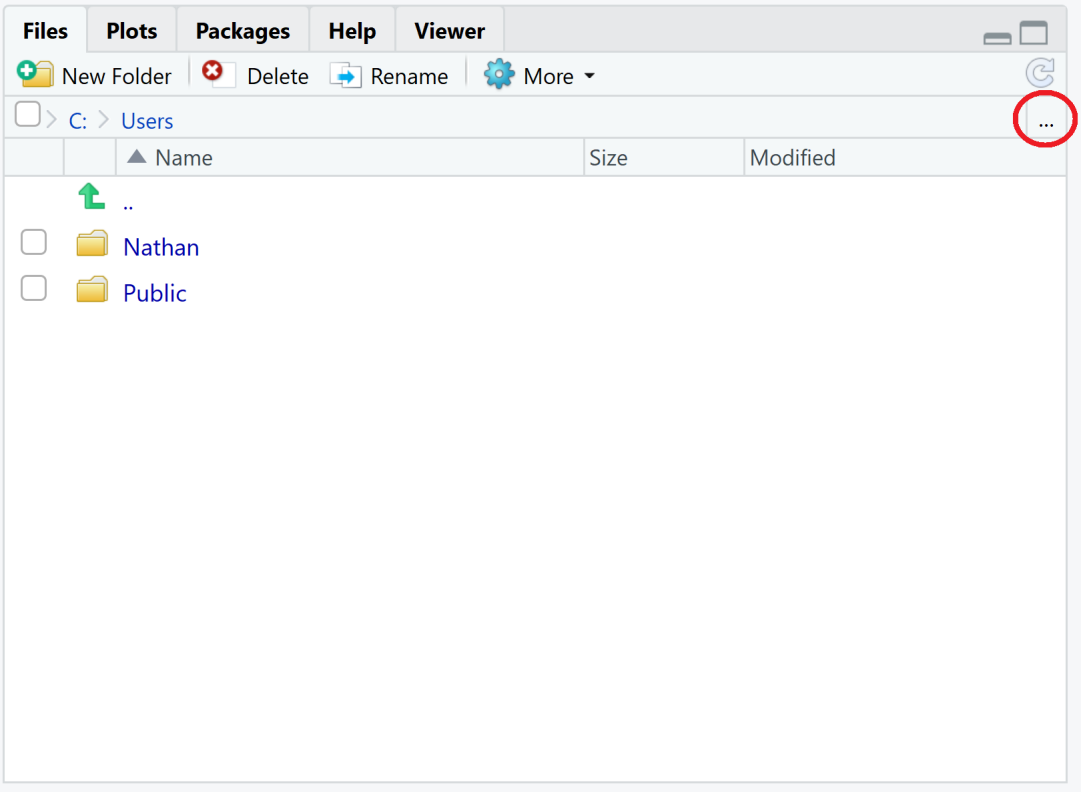
Vous devriez obtenir comme résultat quelque chose comme "C:/Users/*votre_nom*".

Pour changer votre dossier de travail, deux options. Soit vous utilisez la fonction `setwd()`, soit vous utilisez l'interface de R Studio.

Disons que je souhaite que mon dossier de travail devienne "C:/Dropbox/LINGE1222". Il suffit d'entrer la commande suivante :

```
setwd("C:/Dropbox/LINGE1222")
```

Via l'interface de R Studio, il s'agit d'utiliser la sous-fenêtre en bas à droite et de cliquer sur le bouton "..." (voir capture d'écran ci-après)

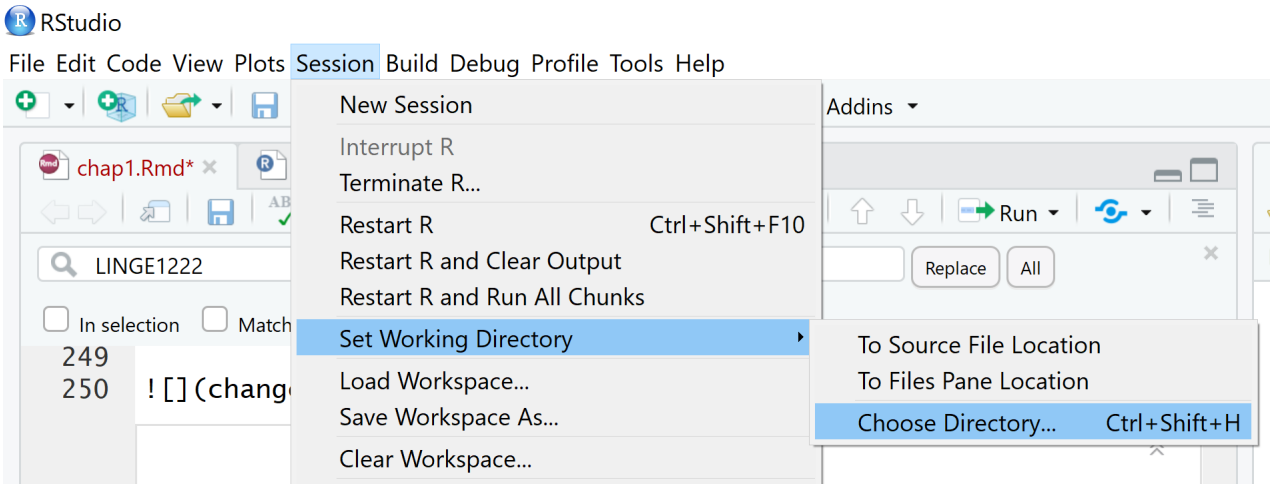


On navigue alors jusqu'au dossier d'intérêt, on valide et ensuite, nous sommes toujours dans la sous-fenêtre en bas à droite, on clique sur :

`More > Set As Working Directory`

Une autre alternative pour choisir son répertoire de travail est d'utiliser le bouton "Session" en haut de la fenêtre principale de R Studio :

`Session > Set Working Directory > Choose Directory`

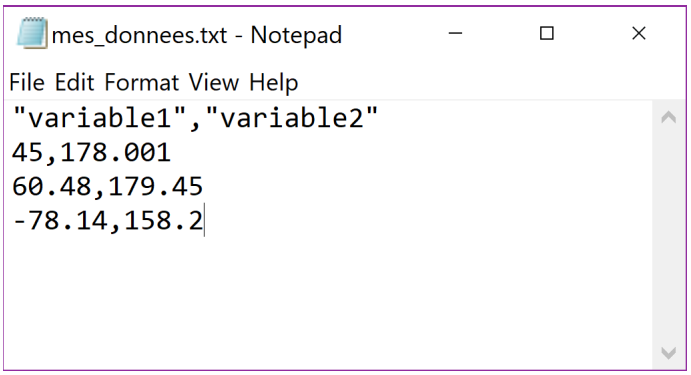


Cette manipulation réalisée, vous pouvez sauver le jeu de données en entrant la commande

```
write.csv(mes_donnees,file="mes_donnees.txt",row.names=FALSE)
```

Le fichier `mes_donnees.txt` devrait apparaître sur le disque dur dans le dossier de travail.

Notez la structure textuelle de ce fichier, ouvert ci-après dans un éditeur de texte très basique :



On voit que :

- Il est composé de 4 lignes de texte.
- la première ligne contient les noms des colonnes (les noms des variables).
- des valeurs successives en interne d'une ligne sont séparées par une virgule.
- le séparateur décimal est le point.

Nous allons à présent importer le contenu de ce fichier texte. Bien sûr, l'objet `mes_donnees` est toujours présent dans R de sorte que l'import du contenu du fichier `mes_donnees.txt` depuis le disque dur n'a aucun sens (on pourrait juste manipuler l'objet `mes_donnees` qui est toujours présent dans R même après export de son contenu). Nous allons toutefois importer le contenu du fichier `mes_donnees.txt` pour montrer comment cela fonctionne.

Pour importer des données, l'option la plus simple est d'utiliser la fonction `read.csv()`, comme ci-après :

```
donnees_importees=read.csv(file="mes_donnees.txt", sep=",", dec=".", header=TRUE)
View(donnees_importees)
```

Quelques explications de la fonction `read.csv()` :

`file="mes_donnees.txt"`

On indique ici le nom du fichier que l'on souhaite importer. **Ce fichier doit se trouver dans notre dossier de travail**.

`sep=","`

Permet d'indiquer que ce qui sépare des valeurs en interne d'une ligne du jeu de données à importer est la virgule.

`dec="."`

Permet d'indiquer que le séparateur décimal dans le jeu de données sur le disque dur est ici le point.

`header=TRUE`

Permet d'indiquer que la première ligne du jeu de données n'est pas un individu mais contient les noms des variables (ce n'est pas toujours le cas, à vous de le savoir !)

`row.names = 1`

Un autre argument utile qui permet de spécifier que la première (d'où = 1) colonne correspond au nom des observations. Voir exercice plus bas avec le dataset Eurojobs pour une application de cet argument.

`View(donnees_importees)`

Permet de visualiser les données après import pour s'assurer que tout s'est bien passé. Un import sans message d'erreur ne signifie pas forcément que l'import s'est fait correctement. Il faut vérifier à l'oeil !

Une fois l'import réalisé, vous pouvez manipuler l'objet `donnees_importees` comme bon vous semble. Par exemple, pour calculer la moyenne de la première colonne de ce jeu de données, le code est

```
mean(donnees_importees[,1])
```

```
[1] 9.113333
```

Une alternative, comme on connait le nom de la première colonne de ce jeu de données, c'est d'utiliser un signe dollar pour faire référence à cette colonne :

```
mean(donnees_importees$variable1)
```

```
[1] 9.113333
```

Cette deuxième méthode est fortement conseillée par rapport à la première dans le cas où vous comptez modifier la structure de votre base de donnée. En effet, si vous ajoutez une colonne, la numérotation change. Par conséquent, on se réfère généralement aux variables par leur nom. De plus, c'est toujours plus facile de comprendre et interpréter du code où le nom de la variable est écrit.

A retenir : lorsque vous importez un jeu de données qui n'est rien d'autre qu'un fichier texte, il faut toujours prendre la peine d'ouvrir ce fichier texte avec un éditeur de texte pour comprendre sa

structure interne. Ensuite seulement, tentez de l'importer dans R Studio.

Aussi : le fichier possède parfois l'extension .csv au lieu de l'extension .txt. Ceci ne fait aucune différence : il s'agit bien d'un simple fichier texte.

5 Les packages

L'une des raisons du succès de R est que n'importe quel utilisateur peut rédiger ses propres codes (sous forme de fonctions) et les uploader sur un serveur central auquel n'importe quel utilisateur peut à son tour accéder. **Le partage de codes est au coeur de la philosophie de R.**

Par défaut dans R, il est possible de créer une matrice, de l'inverser, de générer des valeurs depuis un loi normale, d'additionner deux nombres, etc. Mais parfois ce que l'on désire faire ne fait pas partie des fonctionnalités de base dans R. Ce que R fait par défaut n'est pas toujours non plus satisfaisant : la fonction `plot()` par exemple, qui permet de tracer des graphiques dans R, est pour beaucoup d'utilisateurs trop limitée.

Bonne nouvelle toutefois : peu importe votre problème, il est probable qu'un autre utilisateur l'a déjà résolu et a laissé du code derrière lui.

Lorsqu'un utilisateur uploade une fonction à destination des autres utilisateurs de R, il doit le faire sous la forme d'un *package*. Lorsque vous souhaitez utiliser cette fonction, vous devez télécharger et installer le package contenant cette fonction. Il suffit d'exécuter la commande suivante (une connection internet est nécessaire) :

```
install.packages("nom_du_package_d_interet")
```

Ci-après, exemple de téléchargement et installation du package `ggplot2`, très populaire car permettant d'accéder à de nombreuses fonctions utiles pour tracer des graphiques plus avancés, comme la fonction `ggplot()`.

```
> install.packages("ggplot2")
Installing package into 'C:/Users/Nathan/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.4/ggplot2_2.2.
1.zip'
Content type 'application/zip' length 2784309 bytes (2.7 MB)
downloaded 2.7 MB

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Nathan\AppData\Local\Temp\RtmpOiLOsq\downloaded_packages
>
```

Une fois le package installé, attention ce n'est pas fini. Il faut charger le package et seulement après qu'il ait été chargé vous pourrez utiliser toutes les fonctions qu'il contient.

Pour charger un package, la commande à exécuter est :

```
library(nom_du_package_d_interet)
```

Un package dont on aura besoin dans le cours LINGE1222 est le package `FactoMineR`, très utile pour l'analyse en composantes principales. Pour utiliser les fonctions disponibles dans ce package, il faudra donc exécuter les commandes suivantes:

```
install.packages("FactoMineR") # ne devra être exécutée qu'une seule fois
library(FactoMineR) # devra être exécutée à chaque fois que vous ouvrez R Studio
```

Vous n'avez pas la moindre idée de quelles sont les fonctions se trouvant dans un package donné ? Google sera votre ami.

6 Exercices à réaliser en TP

Les exercices ci-après sont destinés à être réalisés lors du premier TP du cours LINGE1222. Les jeux de données à importer sont tous disponibles sur la page Moodle du cours.

Rappelez-vous : si vous ne savez pas quelle commande R permet de réalisez la manipulation demandée dans l'exercice, Google sera votre ami. Voir aussi la documentation sur la page Moodle du cours.

6.1 Manipulation de données (I)

Importer la base de données Eurojobs (voir Moodle).

- La moyenne et la médiane de la variable Agr sont-elles identiques ? Qu'est-ce que cela implique ?
- Quelle est la variance et quel est le 60ème quantile de la variable Man ?
- Quelle est la corrélation entre les variables SI et SPS ? Entre les variables SPS et SI ?

Solutions :

```
#Import des données
Eurojobs <- read.csv(file="Eurojobs.csv", sep=",", dec=".", header=TRUE, row.names = 1)
# row.names = 1 spécifie que la première colonne correspond au nom des observations
```

```
mean(Eurojobs$Agr)
```

```
[1] 19.13077
```

```
median(Eurojobs$Agr)
```

```
[1] 14.45
```

La moyenne est nettement plus élevée que la médiane ce qui indique probablement la présence de quelques très grandes valeurs pour cette variable dans la base de données.

```
var(Eurojobs$Man)
```

```
[1] 49.10874
```

Attention: dans R, la variance est calculée avec un dénominateur $n-1$ et pas avec un dénominateur n comme dans le cours, où n est la taille de l'échantillon. Si on souhaite la version où on divise par n, on peut faire par exemple comme suit:

```
mean((Eurojobs$Man - mean(Eurojobs$Man))^2)
```

```
[1] 47.21994
```

si on utilise la formule $\mathbb{E}((X - \mathbb{E}(X))^2)$ ou bien

```
mean(Eurojobs$Man^2) - (mean(Eurojobs$Man))^2
```

```
[1] 47.21994
```

si on utilise la formule $\mathbb{E}(X^2) - (\mathbb{E}(X))^2$.

```
quantile(Eurojobs$Man, 0.6)
```

```
 60%
28.5
```

```
cor(Eurojobs$SPS, Eurojobs$SI)
```

```
[1] 0.5721728
```

```
cor(Eurojobs$SI, Eurojobs$SPS)
```

```
[1] 0.5721728
```

La corrélation étant une mesure symétrique, les deux corrélations sont les mêmes.

6.2 Manipulation de données (II)

Importer la base de données `rangs.csv` (voir Moodle).

Cette base de données contient plusieurs variables élaborées afin de mesurer la qualité scientifique de 50 universités. Les variables sont

- Alumni : score basé sur le nombre d'alumnis ayant obtenu un prix Nobel ou une médaille Field;
- HiCi : score basé sur le nombre de membres du personnel académique repris dans la liste des *highly cited researchers Sciences* entre 2002 et 2006;
- Award : score basé sur le nombre de membres du personnel académique ayant obtenu un prix Nobel ou une médaille Field;
- NS : score basé sur le nombre d'articles publiés dans *Nature* et *Sciences* entre 2002 et 2006;
- SCI : score basé sur le nombre d'articles indexés dans *Science Citation Index-expanded* et dans *Social Science Citation Index 2006*;
- Size : moyenne pondérée des cinq mesures précédentes divisées par le nombre d'équivalents temps-plein du personnel académique de l'institution.

Répondre brièvement aux questions suivantes :

- Quelle est la médiane du score *Alumni* ?
- Quelle est la valeur du $30^{ème}$ quantile de la variable *SCI* ?
- Quelle est la corrélation entre les variables *Alumni* et **Award** ?
- Quelle variable présente la plus forte variance ?

Solutions :

```
rangs <- read.csv("rangs.csv", sep = ";", dec = ",", header = TRUE)
```

```
median(rangs$Score.on.Alumni)
```

```
[1] 28.8
```

```
quantile(rangs$Score.on.SCI, 0.3)
```

```
  30%
54.04
```

```
cor(rangs$Score.on.Alumni, rangs$Score.on.Award)
```

```
[1] 0.7539759
```

La variable Award possède la plus grand variance :

```
lapply(rangs[7:13], var)
```

```
$Score.on.Alumni
[1] 536.4677

$Score.on.Award
[1] 638.3379

$Score.on.HiCi
[1] 212.06

$Score.on.N.S
[1] 221.9539

$Score.on.SCI
[1] 159.83

$Score.on.Size
[1] 216.6588

$Total.Score
[1] 207.205
```

Nous aurions trouvé la même réponse si nous avions calculé les variances pour chaque variable une à une et puis comparé celles-ci. La fonction `lapply()` permet cependant d'appliquer une fonction sur plusieurs objets. Dans notre cas, nous appliquons la variance sur les variables de 7 à 13 en une fois.

6.3 Produit matriciel

On définit les matrices

$$\boldsymbol{A} = \begin{pmatrix} 3 & 0 \\ -1 & 2 \\ 1 & 1 \end{pmatrix}, \qquad \boldsymbol{B} = \begin{pmatrix} 1 & 4 & 2 \\ 3 & 1 & 5 \end{pmatrix}$$

Evaluer $\boldsymbol{AB}$ à la main et à l'aide de R Studio.

Solution :

```
A=matrix(c(3, -1, 1, 0, 2, 1), nrow=3, ncol=2)
B=matrix(c(1, 3, 4, 1, 2, 5), nrow=2, ncol=3)
A%*%B
```

```
     [,1] [,2] [,3]
[1,]    3   12    6
[2,]    5   -2    8
[3,]    4    5    7
```

6.4 Transposée et somme de matrices

On définit les matrices

$$\boldsymbol{A} = \begin{pmatrix} 4 & 2 \\ 0 & 9 \end{pmatrix}, \qquad \boldsymbol{B} = \begin{pmatrix} 7 & 0 \\ 3 & 1 \end{pmatrix}$$

Evaluer $\boldsymbol{A}' + \boldsymbol{B}$ à la main et à l'aide de R Studio.

Solution :

```
A=matrix(c(4, 2, 0, 9), ncol = 2, nrow=2, byrow = TRUE)
B=matrix(c(7, 0, 3, 1), ncol = 2, nrow=2, byrow = TRUE)
t(A)+B
```

```
     [,1] [,2]
[1,]   11    0
[2,]    5   10
```

6.5 Trace d'une matrice

On définit la matrice

$$\boldsymbol{Z} = \begin{pmatrix} 1 & 2 & 5 \\ 3 & 9 & 6 \\ 1 & 2 & 9 \end{pmatrix}$$

Calculer à la main et à l'aide de R Studio la trace de cette matrice.

Solution :

```
Z=matrix(c(1,2,5,3,9,6,1,2,9),ncol=3,nrow=3,byrow=TRUE)
sum(diag(Z))
```

```
[1] 19
```

6.6 Valeurs et vecteurs propres d'une matrice

On définit la matrice

$$\boldsymbol{A} = \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix}$$

- Déterminer à l'aide de R Studio les valeurs et vecteurs propres de cette matrice.
- Vérifier que la somme des valeurs propres est égale à la trace et que le produit est égal au déterminant de la matrice.
- Vérifier que les deux vecteurs propres soient orthonormés.

Solution :

```
A=matrix(c(2, 1, 1, 4), ncol = 2,nrow=2, byrow = TRUE)
res=eigen(A)
res
```

```
eigen() decomposition
$values
[1] 4.414214 1.585786

$vectors
           [,1]       [,2]
[1,] 0.3826834 -0.9238795
[2,] 0.9238795  0.3826834
```

```
sum(res$values)
```

```
[1] 6
```

```
sum(diag(A)) # trace de A
```

```
[1] 6
```

```r
prod(res$values)
```

```
[1] 7
```

```r
det(A) # déterminant de A
```

```
[1] 7
```

```r
t(res$vectors) %*% res$vectors
```

```
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

Remarque : l'objet `res` contient deux parties: les valeurs propres dans `res$values`, et les vecteurs propres dans `res$vectors`. Ces derniers sont fournis sous forme d'une matrice, dont chaque colonne est un vecteur propre. Dans la dernière ligne de code, on vérifie que cette matrice est orthogonale, ce qui confirme que les vecteurs propres sont orthonormés.

6.7 Indépendance linéaire de vecteurs

Lesquels des ensembles de vecteurs (lignes) ci-après, tous dans un espace en trois dimensions, sont linéairement dépendants?

1. $(4, -1, 2)$ et $(-4, 10, 2)$
2. $(-3, 0, 4)$, $(5, -1, 2)$ et $(1, 1, 3)$
3. $(8, -1, 3)$ et $(4, 0, 1)$
4. $(-2, 0, 1)$, $(3, 2, 5)$, $(6, -1, 1)$ et $(7, 0, -2)$

Solutions :

1. Deux vecteurs dans un espace en dimension 3 sont nécessairement linéairement indépendants à moins que l'un ne soit un multiple de l'autre, ce qui n'est pas le cas ici.
2. Trois vecteurs sont linéairement dépendants si l'un est une combinaison linéaire des deux autres. Si c'est le cas, le déterminant de la matrice formée par les vecteurs sera nul. Ici,

$$\det \begin{pmatrix} -3 & 5 & 1 \\ 0 & -1 & 1 \\ 4 & 2 & 3 \end{pmatrix} = 39 \neq 0$$

   ce qui implique que les vecteurs sont linéairement indépendants.

```r
B1=c(-3, 0, 4)
B2=c(5, -1, 2)
B3=c(1, 1, 3)
det(cbind(B1, B2, B3))
```

```
[1] 39
```

3. Deux vecteurs dans un espace en dimension 3 sont nécessairement linéairement indépendants à moins que l'un ne soit un multiple de l'autre, ce qui n'est pas le cas ici.

4. Un ensemble de quatre vecteurs dans un espace en trois dimensions est nécessairement linéairement dépendant.

6.8 Produit scalaire

On définit les vecteurs $\boldsymbol{a} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$ et $\boldsymbol{b} = \begin{pmatrix} -2 \\ 3 \end{pmatrix}$. Calculer le produit scalaire de ces deux vecteurs.

Solution :

```
a=c(3, 4)
b=c(-2, 3)
sum(a*b)
```

```
[1] 6
```

6.9 Distance euclidienne et angle

Dans un espace en dimension 4, on retrouve les points $\boldsymbol{t} = (2, 4, 0, -2)'$ et $\boldsymbol{u} = (-1, 0, 0, 2)'$.

- Calculer la distance euclidienne $d(\boldsymbol{t}, \boldsymbol{u})$ entre ces deux points.
- Calculer l'angle θ entre ces deux vecteurs.

Solution :

```
t=c(2, 4, 0, -2)
u=c(-1, 0, 0, 2)
sqrt(sum((t-u)^2)) # distance
```

```
[1] 6.403124
```

```
acos(sum(t*u) / sqrt(sum(t*t) * sum(u*u))) # angle (en radians)
```

```
[1] 2.150436
```