

TP 1 : Mise en route

1 Conseils importants

Merci de lire la fiche de TP en détail. Avant d'appeler le responsable du TP, vérifier **systématiquement** :

- que vous avez lu le message d'erreur (et traduit s'il est partiellement en anglais),
- que vous avez bien lu la fiche en détail,
- que votre code est bien formaté (espace, retour à la ligne, etc.) et qu'il n'y a pas de fautes de frappe,
- et que vous avez cherché dans les pages d'aide de R.

Ne pas copier-coller le code inclus dans le PDF. Utilisez vos doigts et le clavier pour le saisir ! On apprend en le tapant.

Quand vous ne comprenez pas une commande compliquée, ou qu'il y a une erreur sur une commande compliquée, essayez de la décomposer pour voir ce que fait chaque partie. . .

2 Un petit tour de RStudio

2.1 La console R et ses limites

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. 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 RStudio comme suit.

- Ouvrir RStudio
- Dans la partie console à gauche, taper par exemple `3 + 5` (puis entrée). Il vous répond...
- Taper `x <- 2.718` pour créer la variable `x`. Et taper ensuite `x` pour voir son contenu.
- Essayer deux autres calculs simples.

► La console possède un historique. Elle permet de recopier sous le curseur les commandes déjà saisies pour les modifier et les ré-évaluer.

- Jouer avec les touches flèches haut et flèches bas lorsque votre curseur est dans la console R.
- Corriger la première commande `3 + 5` en `3.4 + 5.5` et évaluer le résultat.
- Quitter RStudio : s'il vous pose une question comme « *Save workspace image to ...* », répondez *Save*.
- Relancer RStudio.
- Constaté, avec les flèches, que vous retrouvez l'ensemble des commandes que vous avez tapé avant de quitter R.
- Taper `x` (puis entrée) pour afficher le contenu de la variable `x`. Que constate-t-on ?

La ligne de commande n'est pas un outil pratique dès que l'on veut faire un peu de code. Prenons l'exemple du code ci-dessous, qui est une boucle qui affiche tous les nombres entiers de 1 à 10.

```
for (i in 1:10) {  
  print(i)  
}
```

► Essayer de taper ces trois lignes de codes dans la console R.

- Vous devez constater que, dès que l'on tape une parenthèse ouvrante, RStudio ajoute une parenthèse fermante. Ce qui évite d'oublier de la fermer.
- La même chose est vraie pour l'accolade ouvrante, qui entraîne automatiquement une accolade fermante. Il faut

donc la supprimer avant d'aller à la ligne.

- Notez que, au moment où vous aller à la ligne, l'invite de commande, qui est symbolisé par un `>` dans la console R, se change en `+`.
- R a détecté que l'accolade ouvrante de la première ligne n'est pas fermée, et attend la fin complète de la commande avant de l'évaluer.

Lorsque vous cherchez à évaluer une commande et que rien ne se passe, pensez à vérifier que l'invite de commande n'est pas un `+`. Dans ce cas, cela veut dire qu'il y a un ou plusieurs symboles ouvrants qui ne sont pas refermés, comme

- des parenthèses, des crochets ou des accolades : `() [] { }`,
- des guillemets (*quote* en anglais) simples ou doubles : `" " ' ', ...`

On veut maintenant corriger le code saisi dans la console pour que l'on obtienne le résultat de

```
for (i in 1:10) {
  print(i, digits = 3)
}
```

- Essayer de deviner ce que fait ce code avant de l'évaluer.
- Essayer d'utiliser l'historique pour modifier les trois lignes de codes précédemment saisies et obtenir le résultat de ces trois lignes de code.
 - Constater qu'il est compliqué de retrouver les trois lignes de code dans l'ordre,

2.2 Utilisation de scripts

Les scripts R sont des fichiers au format texte, dont l'extension est `.R`, et qui permettent d'enregistrer des lignes de commandes. On peut ainsi ré-évaluer ces lignes de commandes, les copier-coller puis les modifier, etc. Cela permet également de retrouver quelques temps après l'ensemble des commandes nécessaires pour obtenir un résultat. C'est très utile pour refaire une étude statistique en complétant des données, modifiant les légendes des graphiques, utilisant d'autres tests statistiques, etc. C'est aussi très utile pour retrouver comment on peut obtenir tel type de résultats avec R, à partir d'exemples que l'on a déjà fait.

Conséquence : ne jamais saisir le code dans la console directement, comme on l'a fait exceptionnellement aujourd'hui, sauf pour faire quelques tests.

Dans un fichier script, tout ce qui est sur la ligne après un symbole dièse (`#`) est considéré comme du commentaire, et n'est pas évalué.

- À l'aide de `File > New File > R Script`, créer un nouveau script R et remettre la première boucle de code avec `print(i)` au début du script.
 - Au moment où vous ouvrez ce nouveau fichier de script, la colonne de gauche se divise en deux : la console en bas, le script en haut.
 - Constater que le code est mis en couleur en fonction de la syntaxe. Cela aide à corriger les erreurs...
 - Constater que le nom du script (dans le nom de l'onglet) « `Untitled1` » est en rouge, suivi d'une étoile. Cela veut dire qu'il y a eu des modifications depuis la dernière sauvegarde.
 - Enregistrer le script dans le fichier nommé `mon_premier_script.R`.
 - Insérer une ligne supplémentaire en haut du fichier de script où il y a écrit : « `# Première boucle en R` ».

Il y a plusieurs façons d'évaluer le code dans ce script.

1. Pour évaluer tout le script, taper dans la console `source("mon_premier_script.R")`
Alternativement, on peut aussi cliquer sur le bouton « Source » de l'onglet contenant le script pour produire cette ligne de commande et l'évaluer.
2. Sélectionner une partie du code dans le script, par exemple `1:10` et cliquer sur le bouton « Run » en haut de la fenêtre de script.
3. Poser votre curseur sur une ligne, sans avoir de sélection, et appuyer sur le bouton « Run ». Toute la ligne est copiée et évaluée dans la console. Notez que le curseur change ensuite de ligne dans le script.

Enfin, plutôt que d'appuyer sur bouton « Run » avec la souris, on peut utiliser un raccourci clavier. Celui-ci est Ctrl+Entrée sous Windows, Cmd+Entrée sous Mac. . . **Pour se souvenir** du raccourci clavier, glisser le pointeur / la flèche de la souris sur le bouton « Run » sans cliquer, et attendre qu'une bulle d'aide apparaisse. Le raccourci clavier est à la fin de la bulle d'aide.

- ▶ Tester toutes ces possibilités sur votre fichier de script.
- ▶ Ajouter la seconde boucle dans le script, précédée d'une ligne blanche, et d'une ligne de commentaire qui indique que c'est votre seconde boucle. Tester de nouveau différentes possibilités.

2.3 Un test pour les séances futures avec des packages

On peut ajouter différentes fonctionnalités à R en installant, puis en activant différents « *packages* ». L'onglet « Packages » du cadre inférieur droit de RStudio sert à manipuler ces packages.

Il faut cocher ou décocher une case dans la liste des packages pour charger ou décharger un package. Notez que cela engendre une commande dans la console R qui est évaluée.

Le bouton « *Install* » de cet onglet permet d'installer de nouveaux packages en les téléchargeant depuis internet.

- ▶ Ouvrir cet onglet et constater que vous voyez apparaître une liste de package déjà installé avec la version de base de R.
- ▶ Charger le package MASS, évaluer la commande `ginv(10)` dans la console, le décharger et recommencer cette évaluation. Que se passe-t-il ?
- ▶ Essayer d'installer le package `tidyverse` en utilisant ce bouton. Si cela ne fonctionne pas, essayer le package `magrittr`. Signaler à l'enseignant le résultat de ces commandes.

2.4 Mise en forme du code

Il est très important, voire fondamental, de bien présenter son code. On lit et on relit souvent son code (pour chercher une erreur, retrouver une méthodologie, etc.). Il doit donc non seulement être compréhensible par l'ordinateur, mais aussi par un être humain.

Avez-vous remarqué mes espaces autour des symboles `+` ou `<-` et la façon dont la boucle est écrite ? Ces règles de mise en forme sont tellement importantes que de grandes compagnies ont défini des standards. Par exemple, Google :

<https://google.github.io/styleguide/Rguide.xml>

fournit un tas d'exemples et de contre-exemples de ce qu'il faut et ne faut pas faire. Je vous invite à lire à connaître ces règles et les respecter. **Elles vous aideront énormément à corriger vos erreurs.**

Par exemple, il est très compliqué à la lecture de comprendre ce que fait la ligne de code ci-dessous.

```
for(i in 1:10){ii<-i/3+1;print(ii,digits=i)}
```

Il est beaucoup plus simple de lire le code suivant.

```
for (i in 1:10) {
  ii <- i / 3 + 1
  print(ii, digits=i)
}
```

Remarquer l'indentation, c'est-à-dire les deux espaces en début de ligne, à l'intérieur de la boucle. Cela aide l'oeil à retrouver le début et la fin de la boucle.

- ▶ Lire et comprendre les règles de Google jusqu'à la partie « Organization ».
- ▶ Corriger le code que vous avez taper dans le script pour respecter ces règles.

Il existe un complément à ces règles Google, pour utiliser des packages récents de manipulation de données. Voir <http://style.tidyverse.org> . Pour l'instant, on se contentera des règles de Google.

3 Premier projet

Bien souvent, on utilise le même logiciel pour faire plusieurs choses pendant la même période. Par exemple, vous allez utiliser R dans plusieurs UE de votre année de M1 en même temps.

Je vous propose de créer un projet par UE. Un projet est enregistré dans un répertoire/dossier de votre compte informatique. Vous pourrez mettre dans ce répertoire : les sujets de TP, les scripts et notebook que vous allez écrire, les fichiers qui contiennent les données, les fichiers qui contiennent des variables R, l'historique de la console, etc.

3.1 Création

Pour manipuler un projet R, vous pouvez :

1. Utiliser le bouton « Project : (None) » en haut à droite de la fenêtre RStudio, puis cliquer sur « New Project... », « Open Project... » ou « Close Project » suivant que vous vouliez créer un nouveau projet, ouvrir un projet existant, ou fermer le projet que vous utilisez actuellement.
2. Utiliser les commandes « New Project... », « Open Project... » ou « Close Project » du menu « File » de RStudio.

► Créer un nouveau projet :

- en répondant positivement aux questions qui concernent l'enregistrement de ce qui existe déjà,
- de type « *New Directory* », *New Project*,
- dont le nom du répertoire (*Directory name*) est `M1_LogicielR`
- comme sous-répertoire de ce que vous voulez dans votre compte sur les ordinateurs (utiliser le bouton « Browse » pour parcourir le système de fichier jusqu'à ce que vous soyez où vous souhaitez ajouter un répertoire)
- sans cocher les options parlant de « *git* » et « *packrat* » si elles existent dans votre version de RStudio.

► Constater qu'à la place du bouton « Project : (none) » en haut à droite, vous avez maintenant un bouton qui s'appelle « `M1_LogicielR` ».

► Constater que tout a été ré-initialisé. La console est vide, il n'y a plus de script ouvert,...

► En profiter pour regarder, tout en haut de la console, quelle est la version de R actuellement installée sur les ordinateurs de la salle. Signaler ce numéro de version à l'enseignant.

Pour information, la dernière version de R en septembre 2021 est la 4.1.1 (*Kick Things*). Sur ma machine portable, je travaille au moment de taper ce sujet avec la version 4.1.0 (*Camp Pontanezen*), que je mettrai à jour dans peu de temps. . .

On peut recommencer une nouvelle fois.

► Fermer ce projet que vous venez de créer et constater et que vous retrouvez tout ce que vous aviez perdu.

► Créer, avec les mêmes options que précédemment un projet dont le nom du répertoire est nommé `M1_Statistique` pour l'UE de statistique, dans le même répertoire que celui où vous avez posé `M1_LogicielR`.

► Revenez au projet `M1_LogicielR`.

3.2 Répertoire de travail

L'onglet « Files » dans le cadrant en bas à droite permet de naviguer dans le système de fichier. Lorsqu'un projet est ouvert, le répertoire affiché est celui du projet. On constate qu'il ne contient qu'un fichier dont l'extension est `.Rproj`, qui porte le même nom que le projet, et qui enregistre des informations relatives au projet en question.

Lorsque R est lancé (comme c'est le cas dans la partie « Console » de RStudio), il utilise un « répertoire de travail », que l'on peut éventuellement changer.

Le répertoire de travail (ou "Working directory") est l'endroit du système de fichiers de l'ordinateur où R lit et écrit des informations par défaut. Lorsque l'on utilise un projet RStudio, le répertoire de travail est exactement le répertoire du projet.

Lorsque vous démarrez RStudio sans projet, ou que vous démarrez R autrement, pensez à choisir le bon répertoire de travail. Pour cela, on peut utiliser la commande « Set as working directory » du bouton « More » de l'onglet « Files » après s'être placé dans le bon répertoire. La commande `setwd` fait la même chose. Par exemple `setwd("~/MesTPInfos/M1_LogicielR")` permet de choisir le sous-répertoire `M1_LogicielR` du répertoire `MesTPInfos` de la racine de votre compte sous Linux ou Mac, noté `~`. Sous Windows, il faut également donner le chemin complet du répertoire, avec les règles de Windows. 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.

- ▶ Avec la commande `getwd()`, trouvez quel est le répertoire de travail. Vous devez voir apparaître le chemin complet du répertoire du projet.
- ▶ Avec la commande « Recent Files » du menu global « File » de RStudio, rouvrir le script `mon_premier_script.R`. Et à l'aide de la commande « Save as... » de ce même menu, enregistrer une copie de ce fichier dans le répertoire du projet `M1_LogicielR`.
- ▶ Changer de répertoire de travail et constater les différences.
 - À l'aide de l'onglet « Files », créer un répertoire s'appelant `A_supprimer` dans votre répertoire du projet `M1_Logiciel`.
 - Essayez les différentes techniques de pour fixer le répertoire de travail à ce répertoire à supprimer, ou bien au répertoire qui contient le projet.
 - Vérifiez avec `getwd()` dans la console que vous êtes au bon endroit.
 - Suivant le répertoire de travail dans lequel je me trouve, dans quel cas la commande `source("mon_premier_script.R")` fonctionne ? Et pour `source("../mon_premier_script.R")`, sachant que `../` désigne le répertoire parent ?
 - Comprendre le message d'erreur et savoir l'identifier à l'avenir.
- ▶ À ce stade, en utilisant le navigateur de fichier de votre système, je vous conseille (de renommer ou) d'effacer la version du script `mon_premier_script.R` qui ne se trouve pas dans le répertoire du projet `M1_Logiciel`.

3.3 R Notebook

Attention : Pour utiliser les fonctionnalités de cette partie, il est possible que RStudio vous demande d'installer ou de mettre à jour des packages. Il faut faire ces installations et mise à jour pour que cela fonctionne.

On peut ajouter des notebook dans le projet. Ce sont des fichiers qui mélangent du texte, en français/anglais. . . , et du code.

Le fichier commence par une entête, délimitée par deux lignes contenant trois tirets `---`. En dessous ce trouve le corps du Notebook, avec

- du texte sur fond blanc,
- des morceaux (chunk en anglais) de code R sur fond gris.

- ▶ Utiliser la commande « File » > « New File... » > « R Notebook » des menus principaux de R pour créer un nouveau notebook.
- ▶ Constater que ce notebook contient déjà des lignes d'exemple.
- ▶ Enregistrer ce fichier sous le nom `mon_premier_notebook.Rmd` dans le répertoire du projet de notre UE.

Remarquer que le bloc de code est entouré de deux délimiteurs, sur leurs propres lignes. Le délimiteur ouvrant est formé de trois guillemets à l'envers, suivi de `r` entre accolades, soit `'{r}'`. Peuvent s'ajouter des options dans l'accolation pour nommer le chunk, gérer l'évaluation par R, le format des sorties, etc. Le délimiteur fermant est formé de trois guillemets à l'envers, soit `'{r}'`, sur leur propre ligne.

En haut à droite de chaque chunk ce trouve un triangle vert qui permet d'évaluer l'ensemble du code R dans ce code. Lorsque le code produit un résultat, ce résultat est ajouté au fichier notebook à la suite du bloc. Dans les options du bouton « Run » de la fenêtre du notebook, vous trouverez différentes commandes pour executer des lignes ou des chunks (celui où est le curseur, tous ceux qui précèdent, qui suivent, tous les chunks, etc.). On accède aux options en cliquant sur le petit triangle à côté du bouton « Run ».

Enfin, le bouton « Preview », en haut du Notebook, permet d'avoir une vue mise en forme du notebook au format html.

- ▶ Cliquer sur bouton Preview de `mon_premier_notebook.Rmd` pour regarder le résultat.
- ▶ Executer le chunk d'exemple et refaire une Preview.
- ▶ Comment obtient-on du texte en italique ? Et un lien hypertexte ?
- ▶ Que se passe t-il si on met deux étoiles autour d'un mot (par exemple `**execute**`) ?

4 Quitter RStudio

Pour quitter RStudio, vous pouvez utiliser la commande « *Quit RStudio* » du menu File. Répondre positivement lorsqu'il vous propose d'enregistrer quelque chose.

Vous pouvez ensuite zipper ou archiver le répertoire qui contient le projet RStudio et vous l'envoyez par mail pour pouvoir continuer sur un autre ordinateur.

5 Pour la prochaine séance de TP

On veut créer en R une fonction qui prenne un nombre x entre 0 et 1 en entrée, et renvoie un vecteur v , dont les coordonnées sont le développement en base 10 avec une précision de 8 chiffres après la virgule. Autrement dit,

$$\left| x - \sum_{k=1}^8 10^{-k} v_k \right| < 10^{-8}.$$

Par exemple, si $x = 0.3145$, le vecteur sera $v = (3; 1; 4; 5; 0; 0; 0; 0)$ car $0.3145 = \frac{3}{10} + \frac{1}{100} + \frac{4}{1000} + \frac{5}{10000}$.

Si `floor` est la fonction qui désigne la partie entière, le pseudo-code qui permet de faire ce qui est souhaité est donné ci-dessous.

Pour i allant de 1 à 8 :

Calculer $x \leftarrow 10x$
 Calculer $v_i \leftarrow \text{floor}(x)$
 Remplacer $x \leftarrow x - v_i$

FinPour

- ▶ Comprendre la justesse du pseudo-code ci-dessus. Pour cela, on pourra montrer que, après le i^{e} passage dans la boucle, on a

$$\left| x - \sum_{k=1}^i 10^{-k} v_k \right| < 10^{-i}.$$

- ▶ Implémenter ce code tel quel en R, en utilisant une boucle.
- ▶ Tester ce code sur quelques nombres entre 0 et 1.