**R : résumé**

**Semaine 1 :**

Source en ligne des documents nécessaires à la formation R : [GitHub](https://github.com/enormandeau/introR_2020)

Lien pour télécharger R (Windows) : [Cliquer sur Download R 4.0.2 for Windows](https://cran.r-project.org/bin/windows/base/)

Lien pour télécharger RStudio : [Cliquer sur « Download RSTUDIO for Windows”](https://rstudio.com/products/rstudio/download/#download)

1. **Répertoire de travail:**

Avant de pouvoir utiliser RStudio, l’on doit d’abord créer un répertoire de travail, un espace où tous les scripts et toutes les données iront se stocker. La commande générale est la suivante :

setwd(« Chemin\_vers\_l’annuaire »)

L’on doit écrire cette commande dans la console. La partie « Chemin\_vers\_répertoire » est celle qui doit changer selon l’utilisateur. Par exemple, s’il est décidé que le répertoire de travail se trouvera dans le disque de stockage « C : », dans la section « Utilisateurs », dans la sous-section « mfect », dans la sous-section « Documents » et dans le dossier « R », le chemin vers l’annuaire de travail sera le suivant :

« C :/Users/mfect/Documents/R»

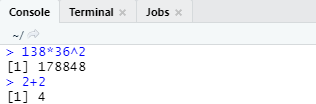
Et la commande de changement de répertorie de travail sera :

setwd(« C:/Users/mfect/Documents/R»)

Faire bien attention de mettre les apostrophes, de mettre des barres obliques orientées vers l’avant (contrairement à Windows qui préfère les orientées vers l’arrière) et de mettre des majuscules là où il doit y en avoir, car RStudio est sensible à la casse. S’il y a une erreur dans l’orthographe, il y aura une erreur dans RStudio et le répertoire de travail ne sera pas créé. Par ailleurs, il y a possibilité de créer « manuellement » le répertoire de travail à partir de RStudio, dans la section « Session », sous-section « Set Working Directory », sous-section « Choose Directory » (raccourci : Ctrl+Shift+H). Il sera dès lors possible de choisir le répertoire de travail avec une interface et la souris.

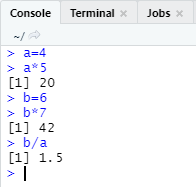
1. **R comme calculatrice (moins intéressant) :**

R peut être utilisé comme une calculatrice, bien que ce ne soit pas son usage principal. Il ne suffit que de taper les opérations désirées comme si l’on utilisait la calculatrice de Windows (voir la section « calculatrice » du dictionnaire de R pour connaître la signification des symboles) :

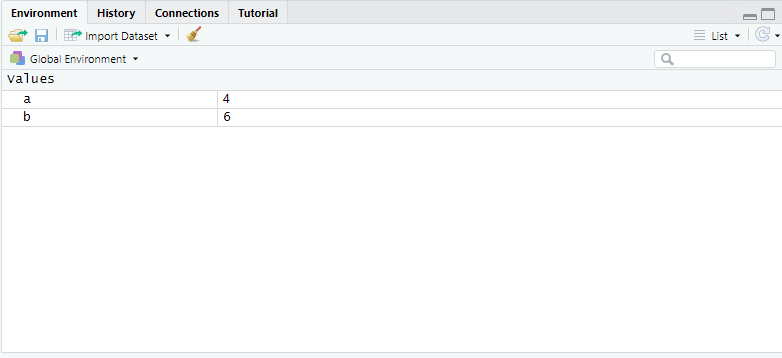


1. **Attribuer un nom à une valeur :**

L’on peut utiliser l’algèbre avec RStudio et attribuer un nom à une valeur. L’on peut ensuite faire des opérations avec ces noms. Cela se fait comme présenté :



La valeur associée au nom se retrouve dans la section « environnement » de RStudio :



* 1. **Nettoyer la mémoire de R:**

Lorsque l’on veut supprimer toutes les variables de la mémoire de R, l’on utilise la commande suivante :

rm(list=ls())

L’on peut aussi cliquer sur l’icône du balai pour nettoyer la mémoire de R

Si l’on veut supprimer une variable spécifique, l’on utilise la commande générale suivante :

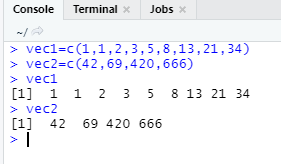
rm(« nom de la variable »)

Par exemple, si l’on veut supprimer la variable « a », la commande sera :

rm(a)

* 1. **Scalaires, vecteurs et matrices :**

L’on peut aussi attribuer un nom à un ensemble de données distribuées selon un certain ordre. Premièrement, l’on peut utiliser le vecteur, qui est un ensemble de données arrangées dans une « ligne ». Pour créer un scalaire, l’on utilise la commande « concatenate », qui signifie un enchaînement de données dans un espace à une dimension (R le lit comme « coller ensemble ») :

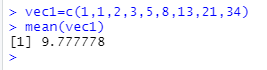


Dans cet exemple, l’on crée un vecteur nommé « vec1 » qui contient un ensemble de nombre. La lettre « c » est le diminutif de « concatenate », la commande utilisée pour créer un vecteur. La commande est suivie de l’ensemble de nombres. Les parenthèses délimitent la longueur de la chaîne de données. C’est le même processus utilisé pour « vec2 ».

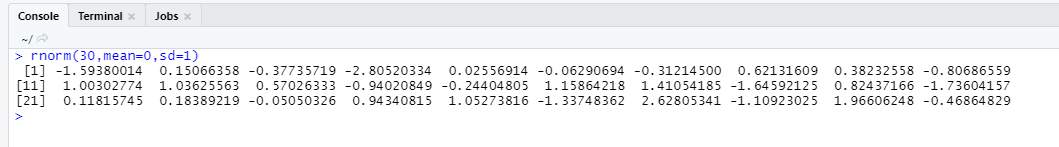
Le même raisonnement s’applique pour créer des matrices (tableaux), mais la commande utilisée, tout comme l’arrangement des données, change. Le principe de matrice et l’orthographe de cette dernière dans R seront abordés dans la semaine 2.

1. **Fonctions :**

Plusieurs fonctions sont implémentées dans R. Par exemple, si l’on veut calculer la moyenne d’un ensemble de données, l’on utilisera la commande « mean » pour la calculer :



Une autre fonction est l’échantillonnage de données « randomisées » à partir d’une distribution normale. La fonction « rnorm » permet une telle « randomisation » de données :

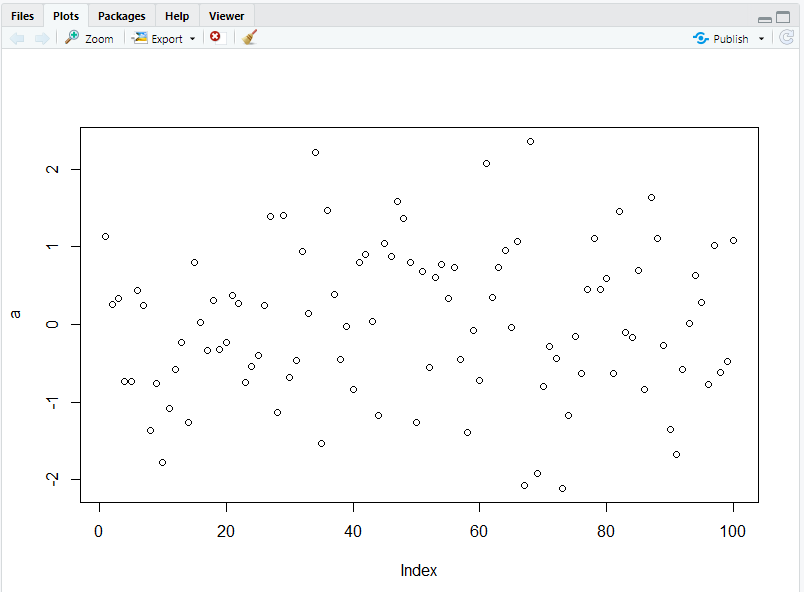
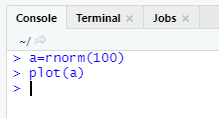


L’on peut aussi changer les paramètres de la moyenne et de l’écart-type de la distribution normale. Lorsque ces paramètres ne sont pas spécifiés, R met les valeurs par défaut.

N.B. : À l’intérieur de fonctions, il est facultatif de mettre des espaces

1. **Graphiques :**

L’on peut faire des graphiques dans R avec la fonction « plot() ». L’on doit spécifier les valeurs qui doivent être inclues dans le graphique. Le graphique sera visible dans la fenêtre « plots » de RStudio. Puisqu’il serait inefficace d’écrire chaque valeur qui doit être inclue dans un graphique, c’est là que les variables deviennent intéressantes. Voici un exemple :



Plusieurs paramètres peuvent être modifiés à l’intérieur de la fonction « plot() », mais les deux paramètres principaux sont les données situées sur l’axe des X et l’axe des Y – la base de données utilisée pour faire le graphique. R fonctionne de la manière suivante :

Plot(« X », »Y », …)

L’on doit d’abord spécifier les valeurs en X, puis les valeurs en Y. La suite comprend l’esthétique du graphique, qui sera abordée plus tard.

**Semaine 2 :**

1. **Script :**

Lorsque l’on veut enregistrer et exécuter un ensemble de commandes dans R, l’on utilise un script. C’est l’outil principal de R. Pour tester une partie d’un script, l’on sélectionne les lignes de codes à tester et l’on fait « Ctrl + Enter » pour les envoyer dans la fenêtre de la console, où les lignes seront exécutées une à la suite de l’autre. Pour exécuter la totalité d’un script, l’on fait « Ctrl + Shift + S ». L’on peut aussi utiliser la commande « Source (« nom ». R) » pour exécuter un script. Il est très important de noter que lorsqu’on veut enregistrer et exécuter un script, l’on doit ajouter « .R » à la fin du nom du script. C’est de cette manière que R reconnaît le type de dossier et le lit.

1. **La fonction « aide » :**

Pour lancer une recherche à partir de R sur un concept plus difficile à comprendre ou pour en apprendre plus sur une fonction, l’on peut utiliser la fonction Help(« Name »). Par exemple, si l’on voulait connaître les paramètres possibles d’un graphique, l’on ferait :

Help(par)

1. **Indexer une valeur :**

Si l’on veut modifier une valeur spécifique dans un vecteur ou dans une matrice, l’on utilise les symboles « [] » pour isoler certaines données dans un ensemble de données. Par exemple, si l’on voulait changer la troisième valeur d’un vecteur pour 12, l’on procèderait ainsi :

Vec1[3]=12

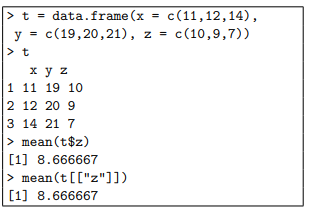
R cible ainsi la valeur originale et la remplace pour la valeur désirée. Même processus pour les matrices et les data frames, mais il faut d’abord spécifier la rangée, puis la colonne, ou vice-versa (l’un est le contraire de l’autre). L’on peut aussi modifier toute une colonne dans une data frame, comme dans une matrice:

Matrice1[,2]=vec1

Puisque l’on a laissé la spécification de la rangée, R prend pour acquis que l’on veut changer les données de toutes les rangées situées également dans la deuxième colonne. Pour les data frames, l’on utilise plutôt le signe « $ » après le nom de la data frame, comme démontré dans la section 8, question de simplifier les lignes de code.

1. **Data frame :**

Les data frames sont similaires aux matrices, à l’exception que les différentes colonnes peuvent avoir différents modes, contrairement aux matrices où toutes les données doivent être des nombres. L’orthographe d’un data frame est différent de celui d’une matrice :

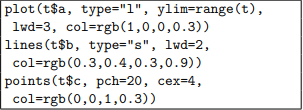


La data frame se construit en colonne. La longueur de chaque colonne ne doit pas nécessairement être équivalente et peut être vérifiée avec un paramètre lors du montage de la data frame. Pour isoler une colonne, l’on ajoute le $ après le nom de la data frame, suivi du nom de la colonne. L’on peut aussi utiliser les symboles [], mais il faut les dédoubler afin que R puisse bien identifier la colonne. Afin de faciliter les choses et d’alléger les lignes de code, R fonctionne aussi avec $ pour les data frames. Dans le domaine de l’agronomie, ce sont les data frames qui sont majoritairement utilisées pour monter les graphiques.

1. Listes :

Les listes ne sont pas ce qu’il y a de plus intéressant dans R, car ce sont essentiellement un « ramassis » de données sans ordre ou structure particulier. Les listes sont construites en « noms de dossier », et la longueur de chaque dossier peut différer, tout comme le mode des données à l’intérieur des dossiers. C’est le même principe qu’une liste d’épicerie ou une « bucket list ». L’on peut même inclure des listes dans des listes.

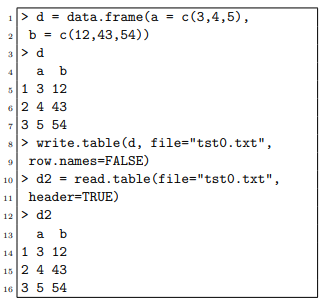
1. Graphiques :

La fonction de base pour créer des graphiques avec R est « plot(arguments) ». Je suggère de taper « plot » sans parenthèses dans la console afin de voir les différents arguments que l’on peut ajouter dans cette fonction (voir le dictionnaire pour un avant-goût). Voici un exemple :

Parallèlement, la fonction pour créer un histogramme, qui peut être utile (je crois) en agronomie, est « hist(arguments) ».

Pour ce qui est des graphiques en général, la meilleure solution pour s’adapter aux différences entre R et Excel, c’est d’en faire le plus possible dans R et compléter avec Excel pour peaufiner la mise en forme et la présentation, car, bien qu’Excel soit plutôt intuitif, faire des graphiques est une tâche plutôt longue et est difficilement « automatisable ». Il y a d’ailleurs la possibilité d’enregistrer des fichiers R de telle sorte qu’Excel soit capable de les lire; la procédure est expliquée dans la partie 11.

1. Lire et écrire des fichiers de données

Il y a plusieurs façons d’écrire de l’information dans l’environnement de R et de lire l’information d’autres fichiers. Par exemple, si l’on créé un tableau dans R, on peut écrire ce tableau dans un fichier de texte brut avec la commande write.table(« nom du tableau »,file= « nom du fichier.txt », row.names= FALSE). Ici, l’argument row.names est mis à la valeur FAUX, de sorte qu’il n’y ait pas de nom de rangée qui soit écrit dans le fichier. Pour lire ce fichier, l’on peut utiliser la commande read.table(file=  « nom du fichier.txt », header=TRUE). Ici, l’argument header permet de lire aussi l’entête des colonnes. Voici la version graphique de cet exemple :

Pour écrire des tableaux qui pourront être lus par Excel, l’on utilise la même commande que celle précédemment donnée, mais l’on change le type de fichier .txt pour un fichier .csv, qui est un type qu’Excel est capable d’interprété. Voici l’exemple :

1. Données non-disponibles :

En agronomie, on travaille presqu’exclusivement avec des données réelles, c’est-à-dire qu’il y a des données non-disponibles à travers les tableaux de données. R a quelque peu de difficulté à calculer avec des données qui sont essentiellement inexistantes. En fait, R traite les données inexistantes comme des NA, des données dont leur valeur est indéterminée et peut être n’importe quelle valeur en même temps. Les difficultés apparaissent lorsque R essaie d’attribuer une valeur à ces données; on peut l’interpréter comme si R le lisait comme un 0 et une valeur infinie en même temps. On peut demander à R de ne pas prendre en compte les données non-disponible avec l’argument « na.rm=TRUE » : R demande s’il doit enlever les NA; on répond oui.