1 L'environement RStudio

1.1 Éléments de RStudio

Présentation des différentes fenêtres de RStudio et personnalisation.
Par défaut La fenêtre de RStudio est composé de 4 panneau principaux
De gauche à droite puis de haut en bas

2 Premiers Pas

2.1 R est une calculette

```
# Une opération simple
10 + 3
# Une Opération plus complexe
10 / (3+8) * 78
# les différents opérateurs sont :
# multiplication *
# addition +
# division /
# soustraction -
```

EXERCICE : Calculer la différence entre votre dernière année à l'université Et votre année d'obtention du bac et divisé par la différence entre 2013 et votre année de naissance. Multiplier par 100 pour obtenir un pourcentage du temps passé à l'université.

```
###Stockage d'un valeur###

#stockage d'une valeur dans une variable
nombreX <- 50.8
#Accès à la valeur stockée dans la variable nombreX
#Utiliser l'auto complétion. Commencer par écrire nomb puis appuyer sur la
    touche de tabulation.
nombreX

# Utilisation de plusieurs variables
# stockage de la variable nombreY (le symbole <- est identique à =)
nombreY <- 7.4
# Utiliser l'auto complétion, si plusieurs solutions existent vous pouvez
# continuer de taper le nom de la variable pour que le choix soit plus
# restreint puis utiliser les flèches du clavier pour choisir la bonne
# variable. Une fois sélectionnée, appuyer soit sur entrée.
nombreX + nombreY
# Stockage du résultat de l'opération dans la variable sommeXY
# Conseil : utiliser l'historique en appuyant sur la 'flèche haut'
# on peut retrouver des commandes déjà écrites.
sommeXY <- nombreX + nombreY
# Opération avec variable et constante déjà existante.
sommeXY
```

EXERCICE : Répéter l'exercice précédant mais en utilisant des variables. Plusieurs étapes peuvent être nécessaires.

2.2 Créer des objets et les utiliser

2.2.1 Vecteurs

```r
####Création###

#création d'un vecteur numérique
monVecteur1 <- c(20, 45, 78, 12)

#A savoir
c(20, 45, 78,

# Une suite de nombre de 1 à 30
maSuite <- seq(from = 1, to = 30)

#Une suite de nombre paire
maSuitePaire <- seq(from = 2, to = 20, by = 2)


# Nous avons utiliser la fonction seq()
# Pour trouver à quoi serve les arguments taper la commande
?seq

# La documentation de cette fonction s'affiche sur la panneau en bas à
    droite de RStudio
```

EXERCICE En utilisant la fonction *seq()*, Créer un vecteur de nombres impaires de 7 à 77.

```r
#La fonction rnorm permet de créer des vecteurs avec des nombres aléatoires
monVecteurAlea <- rnorm(45)

#Création d'un vecteur avec chaine de caractère.
monVecteurA <- c("Mut_1", "Mut_2", "Mut_3")
#La fonction rep permet de répéter une chaine de caractères un certain
    nombre de fois.

rep("Mut", 4)

# La fonction paste permet de coller des chaines de caractères en les sé
    parant par un séparateur.
paste("Mut", 1, sep = "_")
```

EXERCICE En utilisant les fonctions *rep()* et *paste()* et *seq()* créer un vecteur identique à *monVecteurA*, mais allant de 1 à 20.

```r
##Accéder aux éléments##

#Accès à l'élément d'indice 1
monVecteur1[1]
#Accéder aux éléments d'indice  3 , 4 puis 5.

#Accès aux éléments d'indice 1 à 3
monVecteur1[1:3]

# Pour connaitre la longueur d'en vecteur on utilise la fonction length()
length(monVecteur1)
```

EXERCICE Trouver une manière d'acceder au dernier élément d'un vecteur.

```r
##Calcul##

#opération sur le vecteur
monVecteur1 + 5.5
```

```r

#addition de deux vecteurs
monVecteur2 <- c(10, 100, 5, 2)
monVecteur1 + monVecteur2
```

2.2.2 Matrices et data frame

```r
###Création###

##Matrice avec des nombres aléatoires##
maMatrice <- matrix(rnorm(100), ncol = 10)

##S'informer sur la matrice
#voir les n premières lignes d'une matrice
head(maMatrice)

#Combien de lignes la fonction head() affiche - t'elle par défaut
#Comment afficher plus ou moins de ligne ? Utiliser la documentation de la fonction head pour trouver.
#Comment voir la matrice à partir des dernières lignes ?

#Dimension de la matrice : nombre de lignes et nombre de colonnes
dim(maMatrice)
#Quel type d'objet renvoie la fonction dim()

#La fonction ncol() permet de nous renseigner sur le nombre de colonnes.
#Trouver une fonction similaire pour trouver le nombre de lignes

#Avec la fonction dim() afficher uniquement le nombre de lignes

###Accéder aux valeurs###

#Une matrice est un ensemble de vecteur.
#Chaque colonne est un vecteur ainsi que chaque ligne.

#Récupérer des lignes, des colonnes des sous matrices.
#Récupérer la 9ème colonne.
col9 <- maMatrice[, 9]

#De la même façon récupérer une ligne au choix.

#Pour récupérer plusieurs lignes ou colonnes il faut indiquer les indices à récupérer sous forme de vecteurs.

#Pour les colonnes 1,3 et 7
#On indique les indices des colonnes à récupérer
colonnes <- c(1, 3, 7)
#Puis
mesColonnes137 <- maMatrice[, colonnes]
#Il est possible de tout faire en une seule ligne de commande
mesColonnes137 <- maMatrice[, c(1, 3, 7)]

##Quelques fonctions à connaitre##
#Utiliser les fonctions suivante à la fois sur des vecteur comme col9 et sur la matrice entière

# sum()
# mean()
# median()
# length()
```

```r
# summary()

#Calculer la moyenne d'un vecteur de votre choix sans utiliser la fonction mean()

###Nom des colonnes##
#la fonction colnames() permet d'obtenir le nom des colonnes d'une matrice ou d'un data.frame
colnames(maMatrice)

#Que retourne cette fonction ?

#Donner un nom aux colonnes
colnames(maMatrice) <- LETTERS[1:ncol(maMatrice)]
colnames(maMatrice)

##Type et classe d'objet##
typeof(maMatrice)
class(maMatrice)

#Transformer une matrice en data.frame
monDataFrame <- as.data.frame(maMatrice)

#Autre moyen d'accéder aux colonnes avec un data.frame
colonneC <- monDataFrame$C
```

3 Erreurs, importer exporter des données

3.1 Importer et exporter des données

```r
# Importation de données

#La fonction essentiel read.delim
#Sans aucun argument
mesEchantillons <- read.delim("samples.txt")

#Avec des arguments
mesEchantillons <- read.delim("samples.txt", row.names = 1, dec = ",", stringsAsFactor = FALSE, header = TRUE)

#Observer le nom des colones


mesEchantillons <- read.delim("samples2.txt", row.names = 1, dec = ",", stringsAsFactor = FALSE, header = TRUE)

#Possible d'importer des données avec R studio, mais tous les arguments ne sont pas disponibles.

#Sauvegarde dans un fichier
#La fonction essentiel write.table
write.table(x = mesEchantillons, file = "mesEchantillons.txt", col.names = NA , row.names = TRUE, quote = FALSE, sep = "\t")
```

Exercice : retrouver les cinq erreurs de cette ligne de code.

```r
#Ligne de commande erronée.
matriceBug <- read.delin("sample.txt" ,  rown.names = 1  head = true)
```

4 Graphiques

```r
#la fonction plot()
x<-1:20
plot(x, x^2)

#Il existe plusieurs fonction de base pour les graphiques.
# plot()
# hist()
# boxplot()
# Utilisation basique
boxplot(maMatrice)


#Arguments communs à toutes les fonctions
plot(x, x^2, xlim=c(0, 30), ylim=c(-100, 500), xlab="Variable x", ylab="
    Variable x au carré", main="Carré des valeurs de 1 à 20", cex.axis=1.5,
    cex.lab=1.5, cex.main=2, bty="l", pch=16)


# Sauvegarde dans un fichier image
# Dans l'onglet Plots : Export-'Save Plot As Image'
# File name :  boxPlot
# La même chose avec la commande :
png("boxPlot.png")
plot(x, x^2, xlim=c(0, 30), ylim=c(-100, 500), xlab="Variable x", ylab="
    Variable x au carré", main="Carré des valeurs de 1 à 20", cex.axis=1.5,
    cex.lab=1.5, cex.main=2, bty="l", pch=16, col = 2)

#Pour ajouter une légende
legend("topright", legend = "X", pch = 16, col = 2)

# Ferme la fenêtre graphique et enregistre le fichier.
dev.off()

# Il existe 71 paramètres pour affiner les graphiques
# Liste des 71 paramètres de la fonction par()
par()
```

5 Écrire et utiliser une fonction maison

5.0.1 Préparation

Il est plus facile d'écrire des fonctions dans un ou plusieurs fichiers. Créer un nouveau fichier et le nommer mesFonctions.R La synthaxe pour écrire une fonction est la suivante :

```r
maFonction <- function(argument1, argument2 = valeurParDefaut){

  #On utilise le nom des argument comme variable pour faire des calcul,
     appeler des fonctions.
  resultatTemporaire <- argument1  + argument2

  #On continue différent traitement avec d'autre variable créer dans la
     fonction
  resultatTraiter <- uneAutreFonction(resultatTemporaire)

  #On retourne le resultats avec la fonction return()
  return(resultatTraiter)


```

```
}
```

Ennoncé : Créer la fonction *ingredientsPateAPizza* qui prend comme argument le nombre de pizza, par défaut 1. Et retourne un vecteur avec les quantités pour X pizzas. :

```
ingredientsPateAPizza(nbPizza = 1)
  Farine  Eau Levure
[1] 500 250 20
```

Les ingrédients pour une pizza
- farine : 500
- eau : 250
- levure : 20

Tester sa fonction : La fonction est écrite dans le fichier fonction.R, mais elle est encore inconnue dans le *workspace*. Pour la mettre en mémémoire il faut executer la fonction *source()*.

```
source("fonction.R")
```

Si il n'y pas d'erreurs de syntaxe, la fonction sera reconnue par R ce que l'on remarque sur la fenêtre Workspace.
On peut maintenant utiliser notre fonction de la même façon que n'importe quelles autres.

Exercice : Créer la fonction *repIngred* qui prend comme arguments
- *vecIngred* Un vecteur de nombre dont chaque indice à le nom d'un ingrédients.
- *imageName* Une chaine de caractaire qui correspond au nom du fichier image.
- *recipiesName* Une chaine de caratraire qui correspond au nom de la recette.

6 Programation les bases

6.0.2 Les boucles

Le mot *for* permet de répeter une instruction un certain de nombre de fois.

```
x <- 1:20

for(i in 1:length(x)){

x + 100

}
```

Exercice : Dans un nouveau fichier de votre choix. Écrire une boucle *for* exécutant 10 fois la fonction *ingredientsPateAPizza*

6.1 Les conditions

les instructions *if* et *else* permettent d'execter des parties de codes sous certaines conditions.

```
if ( x > 0){
  print("x est positif")

}else{
  print("x est négatif")
}


if ( x > 0){
  print("x est positif")

}else{
  if(x != 0){
```

```
14      print("x est négatif")
15    }else{
16      print("x est nul")
17    }
18  }
```

Exercice : Dans la fonction *ingrédientsPateAPizza* ajouter une instruction *if else* qui vérifie que l'arguments nbPizza soit bien positif et non nul. Auquel cas un message du style "Veillez saisir un nombre de pizza positif" devra s'afficher.