

# 1 L'environnement 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

```
1 ###Calcul###
2
3 # Une opération simple
4 10 + 3
5 # Une Opération plus complexe
6 10 / (3+8) * 78
7 # les différents opérateur sont :
8 # multiplication *
9 # addition +
10 # division /
11 # soustraction -
12
13
14 ###Stockage###
15
16 #stockage d'une valeur dans une variable
17 nombreX <- 50.8
18 #Accès à la valeur stockée dans la variable nombreX
19 #Utiliser l'auto-complétion. Commencer par écrire nomb puis appuyer sur la
    touche de tabulation.
20 nombreX
21
22
23
24 # Utilisation de plusieurs variables
25 # stockage de la variable nombreY (le symbole <- est identique à =)
26 nombreY <- 7.4
27 # Utiliser l'auto-complétion, si plusieurs solutions existent vous pouvez
28 # continuer de taper le nom de la variable pour que le choix soit plus
29 # restreint puis utiliser les flèches du clavier pour choisir la bonne
30 # variable. Une fois sélectionnée, appuyer soit sur entrée.
31 nombreX + nombreY
32 # Stockage du résultat de l'opération dans la variable sommeXY
33 # Conseil : utiliser l'historique en appuyant sur la 'flèche haut'
34 # on peut retrouver des commandes déjà écrites.
35 sommeXY <- nombreX + nombreY
36 # Opération avec variable et constante déjà existante.
37 sommeXY
```

### 2.2 Créer des objets et les utiliser

#### 2.2.1 Vecteurs

```
1 ####Création###
2
3 #création d'un vecteur numérique
4 monVecteur1 <- c(20, 45, 78, 12)
```

```

5
6 # Une suite de nombre de 1 à 30
7 maSuite <- seq(from = 1, to = 30)
8
9 #Une suite de nombre paire
10 maSuitePaire <- seq(from = 2, to = 20, by = 2)
11
12
13 # Nous avons utiliser la fonction seq()
14 # Pour trouver à quoi serve les arguments taper la commande
15 ?seq
16
17 # La documentation de cette fonction s'affiche sur la panneau en bas à
    droite de RStudio
18
19 #La fonction rnorm permet de créer des vecteurs avec des nombres aléatoires
20 monVecteurAlea <- rnorm(45)
21
22 #Création d'un vecteur avec chaine de caractère.
23 monVecteurA <- c("Mut_1", "Mut_2", "Mut_3")
24
25
26 ###S'informer sur les objets###
27
28 ##Taille de l'objet##
29
30 ##Exercice 1 #####
31 # Indiquer la longueur de tous les vecteurs créés au moyen de la fonction
    legnth()
32 #
33 # Longueur des vecteurs :
34 #
35 #
36 #
37 #
38 ##
39 #Remarque :
40 # Pour connaitre tous les vecteurs créés il faut regarder l'onglet
    Workspace
41 # dans le panneau en haut à droite de R studio.
42 # Il est également possible d'utiliser la fonction ls() dans la console R.
43
44 #####
45
46 ##Accéder aux éléments##
47
48 #Accès à l'élément d'indice 1
49 monVecteur1[1]
50 #Accéder aux éléments d'indice 3 , 4 puis 5.
51 # Remarques.
52
53
54 #Accès aux éléments d'indice 1 à 3
55 monVecteur1[1:3]
56
57 ##Calcul##
58
59 #opération sur le vecteur
60 monVecteur1 + 5.5
61
62 #addition de deux vecteurs

```

```

63 monVecteur2 <- c(10, 100, 5, 2)
64 monVecteur1 + monVecteur2

```

### 2.2.2 Matrices et data frame

```

1  ###Creation###
2
3  ##Matrice avec des nombres aléatoires##
4  maMatrice <- matrix(rnorm(100), ncol = 10)
5
6  ##Expliquer cette ligne de commande##
7  #A quoi correspondent les différents mots composants cette ligne de commande
8  .
9  ##S'informer sur la matrice
10 #voir les n premières lignes d'une matrice
11 head(maMatrice)
12
13 #Combien de lignes la fonction head() affiche t'elle par défaut
14 #Comment afficher plus ou moins de ligne ? Utiliser la documentation de la
15   fonction head pour trouver.
16 #Comment voir la matrice à partir des dernieres lignes ?
17
18 #Dimension de la matrice : nombre de lignes et nombre de colones
19 dim(maMatrice)
20 #Quel type d'objet renvoie la fonction dim()
21
22 #La fonction ncol() permet de nous renseigner sur le nombre de colonnes.
23 #Trouver une fonction similaire pour trouver le nombre de lignes
24 #Avec la fonction dim() afficher uniquement le nombre de lignes
25
26 ###Acceder aux valeurs###
27
28 #Une matrice est un ensemble de vecteur.
29 #Chaque colone est un vecteur ainsi que chaque ligne.
30
31 #Recuppperer des lignes, des colones des sous matrices.
32 #Recuperer la 9eme colonne.
33 col9 <- maMatrice[, 9]
34
35 #De la même façon récupérer une ligne au choix.
36
37 #Pour recuperer plusieurs lignes ou colonnes il faut indiquer les indices à
38   recuperer sous forme de vecteurs.
39
40 #Pour les colones 1,3 et 7
41 #On indique les indices des colones à reccuperer
42 colones <- c(1, 3, 7)
43 #Puis
44 mesColones137 <- maMatrice[, colones]
45 #Il est possible de tout faire en une seule ligne de commande
46 mesColones137 <- maMatrice[, c(1, 3, 7)]
47
48 ##Quelques fonctions à connaitre##
49 #Utiliser les fonctions suivante à la fois sur des vecteur comme col9 et sur
50   la matrice entière
51
52 # sum()

```

```

52 # mean()
53 # median()
54 # length()
55 # summary()
56
57 #Calculer la moyenne d'un vecteur de votre choix sans utiliser la fonction
    mean()
58
59 ###Nom des colones##
60 #la fonction colnames() permet d'obtenir le nom des colones d'une matrice ou
    d'un data.frame
61 colnames(maMatrice)
62
63 #Que retourne cette fonction ?
64
65 #Donner un nom au colones
66 colnames(maMatrice) <- LETTERS[1:ncol(maMatrice)]
67 colnames(maMatrice)
68
69 ##Type et classe d'objet##
70 typeof(maMatrice)
71 class(maMatrice)
72
73 #Transformer une matrice en data.frame
74 monDataFrame <- as.data.frame(maMatrice)
75
76 #Autre moyen d'accéder au colone avec un data.frame
77 coloneC <- monDataFrame$C

```

### 3 Erreurs, importer exporter des données

#### 3.1 Importer et exporter des données

```

1 # Importation de données
2
3 #La fonction essentiel read.delim
4 #Sans aucun argument
5 mesEchantillons <- read.delim("samples.txt")
6
7 #Avec des arguments
8 mesEchantillons <- read.delim("samples.txt", row.names = 1, dec = ",",
    stringsAsFactor = FALSE, header = TRUE)
9
10 #Observer le nom des colones
11
12
13 mesEchantillons <- read.delim("samples2.txt", row.names = 1, dec = ",",
    stringsAsFactor = FALSE, header = TRUE)
14
15 #Possible d'importer des données avec R studio mes tous les arguments ne
    sont pas disponibles
16
17 #Sauvegarde dans un fichier
18 #La fonction essentiel write.table
19 write.table(x = mesEchantillons, file = "mesEchantillons.txt", col.names =
    NA , row.names = TRUE, quote = FALSE, sep = "\t")

```

#### 3.2 5 erreurs courantes

```

1 #Ligne de commande erroné.
2 matriceBug <- read.delim("Samples.txt" , rown.names = 1 head = true)
3
4 #Erreur 1 :
5 #Erreur 2 :
6 #Erreur 3 :
7 #Erreur 4 :
8 #Erreur 5 :
9
10 #Bonne ligne de commande :

```

## 4 Graphiques

```

1 #la fonction plot()
2 x<-1:20
3 plot(x, x^2)
4
5 #Il existe plusieurs fonction de base pour les graphiques.
6 # plot()
7 # hist()
8 # barplot()
9 # barplot2()
10 # boxplot()
11 # Utilisation basique
12 boxplot(maMatrice)
13
14
15 #Arguments communs à toutes les fonctions
16 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)
17
18
19 # Sauvegarde dans un fichier image
20 # Dans l'onglet Plots : Export-'Save Plot As Image'
21 # File name : boxPlot
22 # La même chose avec la commande :
23 jpeg("boxPlot.jpeg")
24 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)
25
26 #fermer la fenêtre graphique en cours et enregistre le fichier pour l'
    occasion
27 dev.off()
28
29 #Il existe 71 paramètres pour affiner les graphiques
30 #liste des 71 paramètre de la fonction par()
31 par()

```

## 5 Écrire et utiliser une fonction et l'utiliser dans un script

### 5.0.1 Préparation

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

```

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

```

```

2
3  #On utilise le nom des argument comme variable pour faire des calcul,
   appeler des fonctions.
4  resultatTemporaire <- argument1  + argument2
5
6  #On continue différent traitement avec d'autre variable créer dans la
   fonction
7  resultatTraiter <- uneAutreFonction(resultatTemporaire)
8
9  #On retourne le resultats avec la fonction return()
10 return(resultatTraiter)
11
12
13 }

```

**Enoncé** : 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. :

```

1 ingredientsPateAPizza(nbPizza = 1)
2   Farine   Eau Levure
3 [1] 500 250 20

```

Les ingrédients pour une pizza

- farine : 500
- eau : 250
- levure : 20

**Aide** : Pour nommer les éléments d'un vecteur on utilise la fonction *names()* qui fonctionne de la même façon que *colnames* pour les matrices et *data.frame*.

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

```

1 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.