1 L'environement RStudio

1.1 Données du TP

Réccuperer les fichiers pour le TP en suivant les indication suivantes :
- Créer un nouveau projet
- Choisir version de contrôle
- Choisir Git
- écrire l'addresse suivante : https ://github.com/eHirchaud/formationR

Un dossier nommé formationR sera créé dans votre dossier personel

1.2 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

Le panneau à droite ...(images)

2 Utilisation de R comme calculatrice avec objet simple

Objectif : Comprendre le concept de variable, stockage d'une valeur dans une variable, calcul simple, variable prédéfinie. Connaitre quelques symboles particuliers. Utiliser l'auto-complétion et l'historique.

2.1 Exercice : calcul simple avec les variables

Le symbole > en début de ligne (chevron), permet d'indiquer une ligne de commande : il n'est pas à recopier. Le symbole # permet d'indiquer que la ligne est un commentaire. Cette ligne est là comme information pour l'utilisateur et n'est pas interprétée par R. Tester dans la console ces différents exemples :

```
1
2  # Une addition
3  > 10 + 3
4  # une division
5  > 10 / 3
6  # les différents opérateur sont :
7  # multiplication *
8  # addition +
9  # division /
10 # soustraction -
11
12 # Racine carrée :
13 > 25^0.5
14
15 #Série croissante de chiffres
16 2:5
17
18 #stockage d'une valeur dans une variable
19 > nombreX <- 50.8
20 #Accès à la valeur stockée dans la variable nombreX
21 #Utiliser l'auto-complétion. Commencer par écrire nomb puis appuyer sur la
       touche de tabulation.
22 > nombreX
23
24 #Calcul avec la variable
25 > nombrex + 25
26
27 # Pourquoi l'opération ne marche-t-elle pas ? Corriger et exécuter la ligne
       de commande.
28
29
```

```
# 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
> rayon <- nombreX
> rayon
> pi
> A <- pi * rayon^2 #air d'un cercle en utilisant la variable pi déjà défini
    dans R
# D'autres constantes existent : letters, LETTERS, month.name
```

3 Vecteurs

Un vecteur est un objet qui contient plusieurs éléments.

Objectif : Savoir créer un vecteur, comprendre le comportement d'une opération de vecteur. Accéder à une valeur particulière

3.1 Exercice : operations sur les vecteurs

```

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

#Quelle est la longueur (nombre d'indice) de ce vecteur ?

#-----------------------------------------


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

#Que se passe-t-il lorsque que l'on dépasse la longueur du vecteur ?

#-----------------------------------------------

#Accès aux éléments d'indice de 1 à 3
> monVecteur1[1:3]
#opération sur le vecteur
> monVecteur1 + 5.5

#Quel est le comportement de l'addition d'un nombre sur un vecteur ?

#-------------------------------------------------

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

```

#Quel est le comportement de l'addition de deux vecteurs entre eux ?

-----------------------------------------------

# Addition de deux vecteurs de taille différente :
> monVecteur3 <- c(1, 2, 3)
> monVecteur1 + monVecteur3

> monVecteur4 <- c(50.2, 45.5)
> monVecteur4 + monVecteur1

# Quelle est le comportement de l'opération?

# -------------------------------------------------

#Création d'un vecteur avec chaine de caractère.
> monVecteurA <- c("A", "B", "C")
> monVecteurB <- c(A, B, C)

#Pour quelle raison  monVecteurB n'a pas pu être créer ?

# -------------------------------------------------
```

4 Utiliser des fonctions

Objectif : Utiliser la documentation d'une fonction, exécuter une fonction avec un ou plusieurs argument(s). Les fonctions se terminent par des parenthèses. Elles retournent un résultat qui peut être varié (un objet modifié, une information...) Le résultat de la fonction peut être stocké dans un objet pour être utilisé par la suite. A l'intérieur des parenthèses, la fonction peut comporter des arguments séparés par des virgules.

```
# Utilisation de la documentation :
> ?read.delim
# ou
>  help(read.delimn)

#Fonctions sans argument
> getwd()
> ls()
> dir()

#A quoi sert chacune de ces 3 fonctions? A quelle fenêtre de Rstudio la
    fonction ls() correspond ? Et la fonction dir() ?


#-------------------------------------------------

#Fonctions avec arguments

#Calcul de la median de monVecteur1
> median(x = monVecteur1)
#equivalent à
> median(monVecteur1)

#Somme de tous les éléments de monVecteur1
> sum(monVecteur1)
#Somme de chaque élément des vecteurs.
> sum(monVecteur1, monVecteur2, monVecteur3, monVecteur4)
#Certains arguments possèdent des valeurs par défaut. Ils n'ont pas né
    cessairement besoin d'être modifiés.
```

```
#Regarder la documentation de median ou sum pour trouver un tel argument.

#importation de données issues d'un fichier
#L'auto-complétion est aussi très utile pour connaitre les arguments d'une fonction, une fois la première parenthèse ouverte, appuyer sur la touche tab, la liste des arguments apparait et appuyer sur entrée pour valider l'argument sélectionné. Répéter pour chaque argument.
> maTable <- read.table(file = "Rmatrice.txt", header = TRUE, row.names = 1, sep = "\t")
#équivalent à
> maTable <- read.table("Rmatrice.txt", header = TRUE, row.names = 1, sep = "\t")
#Le premier argument est souvent la variable ou le fichier qui sera traité c'est pourquoi l'on omet souvent le nom de l'argument (souvent file pour un fichier x pour une variable).
```

5 Matrice

Objectif : importer des données issues d'un fichier, sauvegarder des objets dans un fichier, calcul sur les matrices. Les matrices (matrix) sont des objets assimilés à des tableaux, leur contenu est uniquement numérique. La fonction read.delim est identique à read.table mais certains arguments n'ont pas les mêmes valeurs par défaut. (Si vous regardez la documentation de read.delim vous remarquerez qu'il s'agit de la même que read.table)

```
# Importation de données
> maMatrice0 <- read.delim("Rmatrice.txt")

# Obtenir de l'information
  # Voir les premiéres lignes de la matrices
> head(maMatrice0)
# Dimensions de la matrice (lignes, colones)
> dim(maMatrice0)
# Nombre de lignes
> nrow(maMatrice0)

#Quelle est le nombre de lignes et de colonnes de la matrice ?

#-----------------------------------------------

#Trouver une façon d'obtenir le nombre de lignes et une fonction équivalente à nrow pour le nombre de colonnes.

> maMatrice <- read.delim("Rmatrice.txt", row.names = 1)
> head(maMatrice)

#Que signifie l'argument row.names ? :

#-------------------------------------------------


# Effacer un objet
> rm(maMatrice0)

# Propriété de la matrice :
> colnames(maMatrice) \#nom des colonnes
> maMatrice[, 1] # première colonne
> maMatrice$NT_1 # éléments de la colonne ayant pour nom NT_1
> head(maMatrice[, 1]) #premiers éléments de la 1ere colonne
> ech1 <- maMatrice[, 1]
> echNT1 <- maMatrice$NT_1 #premiers éléments de la colonne ayant pour nom NT_1
```

```
36
37 # Calcul avec les matrices
38 > maMatriceLog <- log(maMatrice) #Log tous les éléments de la matrice
39 > head(maMatriceLog)
40 #Fonction apply permet de travailler sur les colonnes ou sur les lignes.
41
42 # Calcul uniquement sur les colonnes avec lafonction apply
43 > geneMedian <- apply(maMatrice, 2, median)
44 # Le deuxième argument '2' indique que le calcul se fait sur les colonnes.
45 # Le troisième argument 'median' signifie que l'on applique (apply) le
     calcul de la médiane sur toutes colonnes.
46 > geneMedian
47 > length(geneMedian)
48
49 #Question : De la même façon créer un profil médian (échantillon médian).
50 # A compléter
51 > profilMedian <- apply(
52
53 #Sauvegarde dans un fichier
54 > write.table(x = maMatriceLog, file = "logmat.txt", col.names = NA , row.
     names = TRUE, quote = FALSE, sep = "\t")
```

6 Graphiques

Objectifs : savoir utiliser des fonctions graphiques simples, savoir sauvegarder une image.

```
1 # Utilisation basique
2 > boxplot(maMatriceLog)
3 # Avec quelques paramètres
4 > boxplot(maMatriceLog, col = "blue" , las = "2")
5
6 # Sauvegarde dans un fichier image
7 # Dans l'onglet Plots : Export-> 'Save Plot As Image'
8 # File name :  boxPlot
9 # La même chose avec la commande :
10 > dev.print(jpeg, file = "boxPlot.jpeg")
11
12 # La fonction plot permet de tracer une série de données contre une autre (x
     ,y).
13 # Par exemple on sélectionne la première colonne.
14 > ech1 = maMatrice[, 1]
15
16 # En supposant que le profil median calculé précédemment se nomme
     profilMedian :
17 # Graphique simple (l'argument log permet d'obtenir une échelle en log pour
     les axes mentionnés)
18 > plot(x = profilMedian, y = ech1, log = "xy")
19
20 # Le même graphique avec des paramètres supplémentaires
21 > plot(x = profilMedian, y = ech1, log = "xy" , pch = 19, col = "darkgreen",
     xlab = "Profil Median", ylab = "Echantillon 1", main = "Profil median VS
     Echantillon 1")
```

7 Jeu des 5 erreurs : le débuggage

Le debuggage occupe une part non négligeable du temps pour l'utilateur de R. Avec l'experience, les erreurs se repèrent et se corrigent plus rapidement.

Objectif : Analyser des messages d'erreurs fréquent et les corriger.

But : Corriger la ligne de commande afin de ne plus avoir de message d'erreur.

Méthode : Ouvrir le fichier *bugTuto.R*. Exécutez la ligne de commande telle quelle puis corriger l'erreur annoncée, relancer la ligne de commande et ainsi de suite jusqu'à qu'il n'y ait plus de message d'erreur. Pour chaque erreur trouvée, écrire la cause de l'erreur dans le fichier bugTuto.R.

Conseil
- Utiliser l'historique pour ne pas avoir a réécrire toute la ligne et ajouter de nouvelles erreurs de frappes.
- Abuser de l'auto-completion pour éviter les erreurs de frappes ou de syntaxe.

8 Éxecuter un script

Differentes methodes pour executer un script Ouvrir le fichier 'scriptTuto.R'.

Pas à pas : Dans la fenêtre source, placer le curseur sur la première ligne : Cliquer sur l'icone « Run » ou le raccourci clavier Ctrl+Entrée

Par bloc :
- Sélectionner les lignes à exécuter avec la souris ou le clavier Maj+flèches.
- Puis de la même façon que pour le pas à pas cliquer sur « Run » ou Ctrl+Entrée.

Tout le fichier : Cliquer sur Source ou le raccourcis clavier Maj+Ctrl+S

9 Écrire et utiliser une fonction

Objectif : connaitre la syntaxe de la création d'une fonction. Savoir mettre en mémoire une fonction pour l'utiliser ('sourcer').

Fonction simple Ouvrir le fichier fonctionTuto.R et le fichier script2Tuto.R : Répéter l'exercice précédent avec le fichier script2Tuto.R. On obtient un message d'erreur : la fonction addition n'existe pas pour R. Pour lui faire connaitre il faut 'sourcer' le fichier qui contient la fonction.

Écrire une fonction similaire à addition du style : soustraction, multiplication... Utiliser la fonction écrite dans le fichier script2Tuto.R . Exécuter tout le script.

Pour éviter d'oublier de 'sourcer' le fichier, la méthode standard est d'écrire au début du fichier la commande :

```
> source("fonctionTuto.R")
```

Une autre option propre à RStudio est de cocher la case 'Source on Save' pour le fichier fonctionTuto.R. Cela a pour effet de 'sourcer' automatiquement le fichier fonctionTuto.R à chaque sauvegarde de ce fichier.

10 Bibliotheque (library)

Les bibliothèques sont des ensembles de fonctions et objets particuliers. La bibliothèque limma qui est adaptée à l'analyse de donnée de puce à ADN sera utilisée. Pour utiliser les fonctions de ces bibliothèques, il faut charger la bibliothèque. Pour cela il suffit d'écrire > library("limma") ou alors d'aller sur l'onglet Packages et de cocher la case à coté de limma. Si vous cliquez sur limma, vous aurez accès à toute la documentation de la bibliothèque.

11 Pour aller plus loin : les facteurs

Les facteurs correspondent au paramètre qualitatif : (fort, moyen, faible), (mutation1, mutation2, wild-type)

```
1 #Fonctions utilisées :
2 # rep(), which(), factor()
3 #Création des facteurs.
4 > echFacteur <- factor(c(rep("NT", 6), rep("CYT6H", 8), rep("CYT48H", 6)))
5
6 # Extraire les echantillons NT
7
```

```
> names(echFacteur) #nom des éléments de echFacteur
> colnames(maMatrice) # nom des colonnes de maMatrice
> all(names(echFacteur) == colnames(maMatrice)) # Répond TRUE si les noms de
    echFacteur et les noms des colonnes de maMatrice sont identiques.

#Construction de la commande par étape :

> echFacteur == "NT"  # Teste si chaque élément est égal à " NT "
> which(echFacteur == "NT") # Indices qui répondent à la condition.

#Commande finale :

> NT <- maMatrice[, which(echFacteur == "NT")] #Sélectionne les colonnes de
   la matrice correspondant aux indices ci-dessus.

#De la même façon, récupérer les échantillons CYT6H puis CYT48H
```