Exercices corrigés
Prise en main du logiciel R

MASTER INFORMATIQUE MEDICALE
IFRISSE

Il est recommandé de faire les exercices en rouge

Ce polycopié rassemble différents exercices vous permettant d'apprendre à utiliser le langage de programmation R. Ces exercices sont de difficultés variables. Il est important de réaliser les exercices les plus simples avant de débuter les plus difficiles.

Manipulation des objets R (vecteurs, facteurs, listes, tableaux, et matrices)

Exercice 1

Soit `a=c("lannister", "targaryen", "baratheon", "starck", "greyjoy")`

1. Quelle est la longueur du vecteur ? *indice : `length`*
2. Essayez de faire `a[1:3]`. Qu'obtenez-vous ?
3. Créez un nouveau vecteur `b`ne contenant que lannister et starck.
4. Essayez de faire a[-1]. Qu'obtenez-vous ?
5. Triez par ordre alphabetique. *indice : `sort`* ?

```
a=c("lannister", "targaryen", "baratheon", "starck", "greyjoy")
length(a)
a[1:3]
b = a[c(1,4)]
a[-1]
sort(a)
```

Exercice 2

1. Créez un vecteur `a` contenant tous les entiers de 1 à 100. *Indice: `:`.*
2. Ajoutez les valeurs 200, 201, 202 au vecteur `a`.
3. Créez un vecteur `b` contenant tous les entiers pairs de 2 à 100. *Indice: `seq`.*

Indice: `seq`.

```
a = c(1:100)
c(a, 200, 201, 202)
b = seq(from = 2, to = 100, by = 2)
```

Exercice 3

1. Créer le vecteur `vec1` contenant tous les multiples de 2 compris entre 1 et 50.
2. Créer le vecteur `vec2` contenant 3 fois chacun des 10 chiffres (soit 0, 0, 0 ; 1, 1, 1 ; 2, 2, 2 ; 3, 3, 3 ; etc.). *Indice: `rep()`*
3. Créer le vecteur `vec3` contenant une fois la lettre A, deux fois la lettre B, trois fois la lettre C ... et 26 fois la lettre Z. Quelle est la longueur de cette suite ? *Indice: `LETTERS`; `length()`*

Pensez à bien regarder l'aide des fonctions. Par exemple, pour la fonction `rep` regardez bien la différence entre les paramètres `each` et `times` ; vous pouvez copier-coller les lignes d'exemple pour bien comprendre chaque paramètre.

```
#1
vec1 = seq(from=2, to=50, by=2 )

#2
vec2 = rep(c(0:9), each=3)

#3
vec3 = rep(LETTERS, times = c(1:26))
length(vec3)
```

Exercice 4

La commande `paste` permet de concataner du texte.
Essayez `paste("chr", 1, sep="")`.
Créez, en une seule ligne de commande, le vecteur `vec4` contenant les noms suivants : `chr1`, `chr2`, ... , `chr22`, `chrX`, `chrY`. *Indice : `paste()`*

```
paste("chr", 1, sep="")
vec4 = paste("chr", c(1:22,"X","Y"), sep="")
```

Exercice 5 : Facteurs

1. Définissez un facteur `fac = factor(c("a","b","b","b","a","b","a","a"))`.
2. Calculez le nombre de `"a"` et de `"b"` dans `fac` en utilisant les fonctions `which` et `length` et des opérateurs binaires (`==`).
3. Que permet de faire la fonction `table` ? Appliquez la à `fac`. **Retenez bien la fonction `table` très très utile !**

```
fac = factor(c("a","b","b","b","a","b","a","a"))
length(which(fac == "a"))
length(which(fac == "b"))
table(fac)
```

Exercice 6 : Matrices

1. Exécutez la commande `a = rep(c(0,1), 50)`. Qu'a-t-on fait ?
2. Utilisez `a` pour construire une matrice `A` à 10 lignes et 10 colonnes. *indice : matrix*
3. Affichez les dimensions de cette matrice. *Indices : `dim`, `ncol`, `nrow`*
4. Utilisez la fonction `t` sur cette matrice pour créer une matrice `B`. Que s'est-il passé ?
5. Les commandes `A[1:5, ]` et `B[, 1:5]` permettent de récuperer respectivement les 5 premières lignes de `A` et les 5 premières colonnes de `B`. Inspirez-vous de ces commandes pour récupérez dans `A` les lignes ne contenant que des 1 et dans `B` les colonnes ne contenant que des 0.

```
#1
a = rep(c(0,1), 50)

#2
A = matrix(a, ncol = 10, nrow = 10)
```

```r
#3
dim(A)
ncol(A)
nrow(A)

#4
B = t(A)

#5
line1 = A[seq(2,10,2),]
col0  = B[,seq(1,10,2)]
```

Exercice 7 : `liste` et tableaux de données (`data.frame`)

1. Créez une liste `x` contenant :

- une variable aléatoire gaussienne de taille 10 appelée `a`
- un vecteur contenant uniquement des 1 de taille 10 également, appelé `b`. On peut accéder aux deux éléments de cette liste avec les commandes `x[[i]]` ou `x$nom_de_la_variable`. *Indices : `list`, `rnorm`.*

2. Créez un objet `y` qui est la transformation de cette liste en `data.frame`. On peut maintenant parcourir les éléments de chaque objet comme pour une matrice avec la commande `y[i,j]` ! *indice = `as.data.frame`*
3. Créez deux objets `z1` et `z2` contenant respectivement les 3 premières et les 3 dernières lignes de `y`. Quelle est la classe de ces deux objets ?
4. Rajoutez à la liste `x` un vecteur `chiffre` contenant les entiers de 1 à 26.
5. Essayez de transformer de nouveau `x` en `data.frame`. Que se passe-t-il ?

```r
#1
x = list(
  a = rnorm(10),
  b = rep(1, 10)
)
x[["a"]]
x$a

#2
y = as.data.frame(x)

#3
z1 = y[c(1:3) ,]
z2 = y[c(8:10),]
class(z1)
class(z2)

#4
x$chiffre = 1:26

#5
as.data.frame(x)
```

Exercice final classe objet : Student's Sleep Data

1. Executez la commande `data(sleep)`. Nous venons de charger en mémoire l'un des nombreux jeux de données distribués avec R ! Profitez de l'aide sur ce jeu de données pour en apprendre un peu plus (`?sleep`) ! Tous les jeux de données disponibles avec l'installation de base de R sont accessibles en tapant `data()`.
2. Quel est le type de l'objet `sleep` ?
3. Quelle fonction vous permet d'obtenir rapidement le nombre d'individus par groupe ? Executez la.
4. Combien y-a-t-il de valeurs négatives dans le groupe 1 ? *Indice : `which()`*
5. Soit s un vecteur représentant le sexe des individus : `s = rep(c("f","m","m","f","m","f","m","m","f","m"),2)`. Combinez l'objet `sleep` et le vecteur `s` dans une nouvelle matrice `sleep2`. *Indice : `cbind()`*
6. Quelles sont les noms des colonnes de la matrice `sleep2` ? Renommez la dernière colonne en 'sex'. *Indice : `colnames()`*
7. Combien y-a-t-il de femmes et d'hommes dans chacun des groupes ?

```
#1
data(sleep)
#2
typeof(sleep); class(sleep)
#3
table(sleep$group)
#4
length(which(sleep$extra < 0))
#5
s = rep(c("f","m","m","f","m","f","m","m","f","m"),2)
sleep2 = cbind(sleep, s)
#6
colnames(sleep2)
colnames(sleep2)[4] = "sex"
#7
table(sleep2$s, sleep2$group)
```

Lire et sauvegarder des données

Exercice 8 : Lire les données d'un fichier : fonction `read.table`

Il est possible de lire les données stockées dans des fichiers sous format `txt` grÃ¢ce, entre autres, aux fonctions suivantes: `read.table()`, `read.csv()`, `read.csv2()` et `scan()`. Par ailleurs, la fonction `read.xls()` (resp. `write.xls()`) du package `gdata` fournit les outils pour lire (resp. écrire) des fichiers au format Excel. Il existe aussi la fonction `read.xlsx` (resp. `write.xlsx`) du package `xlsx`.

(Récupérez les fichiers demandés sur le site xxxxxxx). Vous pouvez ouvrir au préalable ces différents fichiers dans un éditeur de texte afin d'identifier le séparateur de colonnes, le symbole de décimale, comment sont définies les valeurs manquantes etc ...

1. Importez dans une variable nommée `A` le jeu de données nommé `auto2004_original.txt`. **Indice : le séparateur de colonne 'tabulation' correspond à '‸ en informatique*
2. Importez dans une variable nommée `B` le jeu de données `auto2004_sans_nom.txt`.
3. Importez dans une variable nommée `C` le jeu de données `auto2004_virgule.txt`.
4. Importez dans une variable nommée `D` le jeu de données `auto2004_don_manquante.txt`. Combien de valeurs manquantes sont contenues dans le fichier ?
5. Importez dans une variable nommée `E` le jeu de données `auto2004_don_manquante(99999).txt`.
6. Quel est le mode des objets créés par la fonction `read.table()` ?

Indice : `help("read.table")`, `help("is.na")`

```
#1
A = read.table(file="auto2004_original.txt", sep="\t", header = TRUE)

#2
B = read.table(file="auto2004_sans_nom.txt", sep="\t", header = FALSE)

#3
C = read.table(file="auto2004_virgule.txt", sep="\t", header = TRUE, dec = ",")

#4
D = read.table(file="auto2004_don_manquante.txt", header = TRUE, sep="\t", na.strings = "")
nb = length(which(is.na(D) == TRUE))

#5
E = read.table(file="auto2004_don_manquante(99999).txt", header = TRUE, sep="\t", na.strings = "99999")

#6
class(E)
```

Exercice 9 : Enregistrer des données

Créer la matrice suivante :

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

1. Ecrire la matrice `A` dans un fichier nommé `matrice.txt`. Que remarquez-vous?
2. Ajouter des arguments à la commande précédente pour retirer des noms aux lignes et aux colonnes du fichier créé.
3. Sauver la matrice `A` au format `.Rdata` dans le fichier `matriceA.Rdata` grâce à la fonction `save`.
4. Que donne la commande `C = load("matriceA.Rdata")` ?
5. Sauver toutes les variables dans un fichier nommé “données.Rdata”

```
A = matrix(seq(12), ncol = 4, byrow = TRUE)

#1
write.table(A, file = "matrice.txt")

#2
write.table(A, file = "matrice.txt", row.names = FALSE, col.names = FALSE)

#3
save(A, file="matriceA.Rdata")

#4
C = load("matriceA.Rdata")
### la matrice A est rechargée.
### la variable C vaut "A"

#5
save(list = ls(), "données.Rdata")
```

Fonctions graphiques

Exercice 10 : quelques graphiques des base en R

1. Chargez le jeu de données `iris`déjà présent dans R.
2. Pour chaque espèce de fleurs, indiquez le nombre de lignes.
3. A partir du résultat précédent, réalisez une représentation en camembert (pie chart), puis en bâtons (barplot). *Indice* ***`pie()`***
4. Tapez la ligne de commande suivante : `summary(iris)`.

Quel résultat obtenez-vous ?

5. Une représentation adéquate est la boîte à moustache (boxplot). Créez un boxplot pour les 4 variables numériques du jeu de données `iris`. *Indice* ***`boxplot()`***
6. Réalisez le même graphique en ajoutant un titre et en supprimant les valeurs extrêmes.
7. Représentez le pie chart et le boxplot sur la même fenêtre graphique. *Indice* ***`par()`****, option mfrow.*
8. Exportez le graphique précédent en pdf sur votre machine. *Indice* ***`pdf()`*** *;* ***`dev.off()`***
9. Représenter un scatter plot simple avec en abscisse la longueur des pétales et en ordonnées leur largeur. *Indice* ***`plot`***
10. Afin de découvrir les différents paramètres de la fonction `plot`, refaite le même graphique qu'à la fonction précédente en :
    - augmentant la taille des points *Indice :* ***`cex`***
    - remplaçant les points par des triangles de couleur rouge *Indice* ***`pch`*** *option*
    - ajoutant une ligne horizontale en pointillé à y=1 *Indice :* ***`abline`***

```r
#1
data(iris)

#2
p = table(iris$Species)

#3
pie(p)
barplot(p)

#4
summary(iris)

#5
boxplot(iris[,1:4])

#6
boxplot(iris[,1:4], title="Iris Boxplot", outline=FALSE)

#7
par(mfrow=c(1,2))
boxplot(iris[,1:4], title="Iris Boxplot", outline=FALSE)
pie(p)

#8
pdf("plotIris.pdf")
par(mfrow=c(1,2))
```

```r
boxplot(iris[,1:4], title="Iris Boxplot", outline=FALSE)
pie(p)
dev.off()

#9
plot(iris$Petal.Length, iris$Petal.Width)

#10
plot(iris$Petal.Length, iris$Petal.Width, cex=5)
plot(iris$Petal.Length, iris$Petal.Width, pch=17, col="red")
abline(h=1, lty=2)

#9
pairs(iris[,1:4], col=as.numeric(iris$Species), pch=16)

#10
pdf("pairs_iris.pdf")
pairs(iris[,1:4], col=as.numeric(iris$Species), pch=16)
dev.off()
```

Exercice 11 : distribution et density plot

1. Charger le jeu de données `airquality`.
2. Prendre connaissance des données (dimension, type).
3. Construire un histogramme de la variable `Ozone`. Représenter l'histogramme en terme de probabilité de densité plutôt qu'en terme de fréquence (axe Y). Ajouter un titre, modifier le noms des axes, et colorer les barres de l'histogramme en gris et les traits de l'histogramme en bleu. Augmenter la taille du pas de l'histogramme à 30.
4. Regarder l'aide de la fonction `density()`. Appliquer cette fonction à la variable `Ozone`. Cela retourne-t-il une erreur ? Pourquoi ? Corriger la en spécifiant une option.
5. Ajouter la courbe de densité précédemment générer à l'histogramme. *Indice* ***`lines()`***

```r
data(airquality)
hist(airquality$Ozone)
hist(airquality$Ozone, freq=FALSE, main="Histogramme Ozone", xlab="Ozone", ylab="Densité", col = "gray"

hist(airquality$Ozone, freq=FALSE, main="Histogramme Ozone", xlab="Ozone", ylab="Densité", col = "gray"
      breaks=30)

d = density(airquality$Ozone, na.rm=TRUE)
lines(d)
```