TP 1 : INTRODUCTION À R ET ANALYSE DESCRIPTIVE UNIDIMENSIONNELLE

Ce TP a pour objectif de vous montrer les commandes de base de R , de vous faire manipuler des tableaux de données et de vous faire assimiler les représentations numériques et graphiques des statistiques descriptives univariées.

1 Introduction au logiciel R (30-40 min.)

1.1 Présentation

- R est un logiciel permettant de faire des analyses statistiques et de produire des graphiques, mais c'est également un langage de programmation complet.
- R est un clône gratuit du logiciel S-Plus commercialisé par MathSoft et développé autour du langage S.
- Les informations sur R sont disponibles sur la homepage du projet : http://www.r-project.org/
- Son développement est libre et structuré autour de packages téléchargeables gratuitement.
- Attention, cette liberté indique que les utilisateurs sont "testeurs" des packages proposés. Ainsi, il faut privilégier les packages de base, existants depuis longtemps et dont les auteurs assurent un suivi régulier.
- Un point de départ est le manuel d'Emmanuel Paradis, "R pour les débutants" et disponible dans la rubrique "Documentation" du CRAN (http://cran.r-project.org/).
- Plusieurs milliers de pages d'enseignement en français de statistiques sous R sont disponibles ici : http://pbil.univ-lyon1.fr/R/.

1.2 Pour commencer avec R

- La fenêtre R Console est la fenêtre principale par défaut où sont réalisées les entrées de commandes et sorties de résultat en mode texte. Le symbole > signifie que R est prêt à travailler. C'est à la suite de ce symbole que vous pouvez taper les commandes R .
- Il est cependant fortement recommandé de composer le code R dans une fenêtre spécifique du logiciel : la fenêtre de script.
  Dans le menu Fichier , les entrées "Nouveau script" ou "Ouvrir un script" permettent de créer un nouveau script de commandes R ou d'accéder à un ancien script sauvegardé lors d'une session précédente d'utilisation du logiciel.
  Pour exécuter des instructions à partir de la fenêtre de script il suffit de procéder par copier-coller ou de se servir de raccourci clavier "ctrl + R".

Pour sauvegarder un script, il suffit, lorsque la fenêtre de script est active, de sélectionner l'entrée "Sauver" du menu `Fichier` .
Dans un script, le symbole # signifie le début d'un commentaire et n'est donc pas pris comme une commande à exécuter.
Il n'y a pas besoin du symbole > comme dans la fenêtre `R` Console.

- Consulter l'aide de `R` :
  Ne pas hésiter à s'y promener pour se familiariser avec son utilisation. Plusieurs manières de faire :

  1. Si on veut accéder au desciptif et options d'une commande :
     ? + *nom de la commande* OU help(*nom de la commande*)

     ```
     > ?mean
     > help(mean)
     ```

  2. Si on ne connait pas la commande, on cherche par mot clé :
     help.search("*mot clé*")

     ```
     > help.search("mean")
     ```

     Les résultats s'affichent alors sous la forme suivante :
     Nom du Package::Nom de la commande Descriptif de la commande

     ```
     base::mean                    Arithmetic Mean
     ```

  3. Utiliser le menu "Aide".

- Sauvegarder sous `R` :
  Vous pouvez sauvegarder deux types de fichiers :

  1. le fichier `.Rdata` contient des informations sur les variables utilisées,
  2. le fichier `.Rhistory` contient l'ensemble des commandes utilisées.

1.3 Rentrer et manipuler des données

- L'AFFECTATION

```
> n <- 15
> N <- 12
> n
[1] 15
> N
[1] 12
```

Remarques :

1. `R` différencie les lettres minuscules et les lettres majuscules
2. Vous pouvez rappeler les commandes déjà exécutées pendant cette séance en utilisant la touche ↑ dans la fenêtre `R` Console.
3. Le signe = marche également à la place du signe ← dans les versions récentes de `R` .

- LES SUITES
  - Nombres entiers :

```
> suite1 <- 1:12
> suite1
```

  - Séquence de nombres :

```
> suite2 <- seq(from=1,to=12,by=1)
> suite2
> suite3 <- seq(from=20,to=40,by=5)
> suite3
```

- LES VECTEURS
  - Vecteur numérique :

```
> serie1 <- c(1.2,36,5.33,-26.5)
> serie1
```

  - Vecteur de chaînes de caractères :

```
> serie2 <- c("bleu","vert","marron")
> serie2
```

  - Vecteur logique :

```
> serie3 <- c(T,T,F,F,T)
> serie3
```

- MANIPULER LES VECTEURS
  - Affichage d'un ou plusieurs éléments

```
> serie1[3]
> serie1[3:4]
```

  - Concaténer des vecteurs

```
> x <- c(2.3,3.5,6,14,12)
> y <- c(3.2,5,0.7,1,3.5)
> z <- c(x,y)
> z
```

  - Extraction des données d'un vecteur

```
> x[c(2,5)]
> x[-c(2,3)]
> x>4
> x[x>4]
```

  - Faire des calculs sur les composantes d'un vecteur

```
> 20+x*5
> (x+y)/2
```

  - Remplacer des données dans un vecteur

```
> x[3] <- 35
> x[3]
> x==1
> x[x==1] <- 25
> x
> x[x>=5] <- 20
> x
```

1.4 Lire des données dans un fichier

Les données sont souvent saisies à l'aide d'un éditeur de texte ou d'un tableur. Il est donc nécessaire de les transférer sous R en indiquant l'endroit où sont stockés les fichiers de données.
Pour connaitre le répertoire courant, il suffit de taper l'instruction suivante :

```
> getwd()
```

Pour changer le répertoire courant, il suffit d'entrer le chemin d'accès correspondant :

```
> setwd("C:/...")
```

Télécharger sur la plateforme e-UAPV les fichiers au format .txt et .csv disponibles. Observer la construction de ces fichiers dans un éditeur de texte et analyser la façon dont ils sont appelés dans R .

```
> tab <- read.table("table1.txt")
> tab
> tab[,1]
> tab[1,1]
> tab[1:2,2]

> tab <- read.table("table2.txt",header=TRUE)
> tab

> tab <- read.table("table3.txt",dec=",")
> tab

> tab <- read.table("table4.txt",dec=";")
> tab

> read.table(file.choose())
```

Il existe deux types de fichier .csv : anglo-saxon avec un "." comme séparateur décimal et une "," comme séparateur de colonne et français avec une "," comme séparateur décimal et un ";" comme séparateur de colonne. Le premier se lit avec l'instruction

```
> read.csv(file.choose())
```

et le second avec l'instruction

```
> read.csv2(file.choose())
```

Remarque : Si la librairie "xlsReadWrite" et la commande read.xls permet de charger des documents au format excel, nous conseillons de convertir dans Excel les fichiers .xls en fichiers .txt ou .csv et d'utiliser les commandes précédentes.

2 Analyse descriptive unidimensionnelle (1h20-1h30)

2.1 Données "Iris"

Le logiciel R est un ensemble de bibliotheques de fonctions appelées packages. Chaque bibliothèque contient des jeux de données. Pour connaître par exemple les jeux de données contenus dans le package "base", écrire l'instruction suivante :

```
> data(package = "base")
```

Le résultat apparaît dans une fenêtre R data sets. En voici un extrait :

```
Data sets in package 'datasets':
AirPassengers.....Monthly Airline Passenger Numbers 1949-1960
BJsales.....Sales Data with Leading Indicator
BJsales.lead (BJsales)..Sales Data with Leading Indicator
BOD.....Biochemical Oxygen Demand
...
iris.....Edgar Anderson's Iris Data
```

1. Noter la présence du fichier "iris". Les données de ce fichier sont célèbres. Elles ont été collectées par Edgar Anderson (1935). Le fichier donne les mesures en centimètres des variables suivantes :

   (i) longueur du sépale (Sepal.Length),
   (ii) largeur du sépale (Sepal.Width),
   (iii) longueur du pétale (Petal.Length)
   (iv) largeur du pétale (Petal.Width)
   pour trois espèces d'iris :
   (i) Iris setosa,
   (ii) Iris versicolor
   (iii) Iris virginica. Sir R.A. Fisher (1936) a utilisé ces données pour construire des combinaisons linéaires des variables permettant de séparer au mieux les trois espèces d'iris.

2. Pour analyser le fichier iris, il faut le charger. Quelle est l'instruction qu'il faut taper pour charger ce fichier ?

3. Taper une à une chacune des instructions ci-dessous et noter le résultat obtenu.

   ```
   >iris
   >dim(iris)
   >names(iris)
   ```

   Quelle(s) différence(s) faites-vous avec la commande `> str(iris)` ? Tapez les commandes suivantes :

   ```
   >iris$Petal.Length
   >iris$Species
   ```

   Qu'observez-vous ?

4. La dernière colonne du fichier "iris" contient le nom des espèces réparties en trois catégories : setosa, versicolor et virginica. Pour accéder à celle-ci, il faut utiliser l'instruction `iris$Species`, comme à la question précédente. On dit alors que la dernière colonne contient une variable qualitative à trois modalités appelées "levels" par le logiciel. La fonction "levels" appliquée à la colonne `iris$Species` donne les modalités de la variable. En effet, il suffit de taper :

   ```
   > levels(iris$Species)
   ```

   Pour résumer l'information contenue dans cette variable, on utilise l'instruction :

   ```
   > summary(iris$Species)
   ```

   Quel est le résultat qui s'affiche ?

5. Cette information peut être obtenue en construisant un tableau (`table`) comptabilisant le nombre d'individus par modalité. Pour ce faire, taper l'instruction suivante :

   ```
   > table(iris$Species)
   ```

   Comparer avec le résultat obtenu à la question précédente.

6. Le logiciel `R` permet de réaliser des graphiques. Lorsqu'une instruction graphique est lancée, une nouvelle fenêtre "device" est ouverte. Les représentations graphiques liées aux variables qualitatives sont la représentation en secteurs ou camembert (pie) et la représentation en bâtons (barplot). Taper les lignes de commande suivantes :

   ```
   >pie(table(iris$Species))
   >barplot(table(iris$Species))
   ```

7. La troisième colonne du fichier "iris" contient la longueur du pétale. Il s'agit d'une variable mesurée qualifiée alors de variable quantitative. Pour résumer l'information contenue dans cette variable, nous utilisons la fonction `summary`. Taper la ligne de commande suivante :

   ```
   >summary(iris$Petal.Length)
   ```

   Quel résultat obtenez-vous ?

   ```
   Min.   1stQu.  Median  Mean   3rdQu.  Max:
   1.000  1.600   4.350   3.758  5.100   6.900
   ```

   Essayons de retrouver ces six valeurs de paramètres. Taper les lignes de commande suivantes :

   ```
   > min(iris$Petal.Length)
   > max(iris$Petal.Length)
   > mean(iris$Petal.Length)
   ```

   Pour calculer la moyenne nous aurions pu procéder autrement. Taper les lignes de commande suivantes :

```
> sum(iris$Petal.Length)
> length(iris$Petal.Length)
> sum(iris$Petal.Length)/length(iris$Petal.Length)
```

Obtenez-vous le même résultat que précédemment ? Maintenant occupons-nous de retrouver les valeurs des trois quartiles. Pour cela, taper la ligne de commande suivante :

```
> sort(iris$Petal.Length)
```

Que se passe-t-il ? Puis continuer par taper les lignes de commandes suivantes :

```
> ordLpetal <- sort(iris$Petal.Length)
> ordLpetal # commenter le résultat
> sum(ordLpetal)/length(ordLpetal)
> ordLpetal[38]
> (ordLpetal[75]+ordLpetal[76])/2
> ordLpetal[113]
```

8. Une des représentations adéquate est l'histogramme (`hist`). Regardez l'aide de `hist()`. Puis, taper la ligne de commande suivante :

   ```
   > hist(iris$Petal.Length,col=grey(0.6),main="Histogramme")
   ```

   `main` est l'option qui permet d'afficher un titre dans un graphique.

9. Réaliser le même type d'analyse sur chacune des trois autres variables quantitatives : largeur du pétale, longueur du sépale et largeur du sépale. Notez que vous n'avez pas toutes les instructions à réécrire en utilisant le système de flèches du clavier. $\uparrow$ et $\downarrow$ vous permettent de retrouver les fonctions que vous avez utilisées. $\leftarrow$ et $\rightarrow$ vous permettent de vous déplacer dans la fonction et donc, dans changer certains paramètres.

Un glossaire qui peut servir de résume

Fonction	Description
`q()`	Quitte le logiciel
`?commande`	Demande la fiche de documentation de la **commande**
`s <-valeur`	Initialise la variable `s` avec la **valeur**. **Exemple** : `n<-5`
`n : m`	Crée une suite de nombres entiers de `n` à `m`.
`seq(n,m,i)`	Crée une suite de nombres de `n` à `m` en incrémentant par `i` **Exemple** : `seq(2,4,0.5)` produit la suite 2, 2.5, 3, 3.5, 4
`c(s1,s2,...,sk)`	Crée une suite en collant les $s_1, s_2, ..., s_k$ dans l'ordre. **Exemple** : `c(2:5,7,seq(8,9,0.5))` produit `2,3,4,5,7,8,8.5,9`
`rep(s,n)`	Crée une suite contenant `n` fois `n` . **Exemple** : `rep(c(1,2,3),2)` donne `1,2,3,1,2,3`
`scan()`	Saisir "au clavier" un jeu de données numériques

`s[I]`	Crée une suite composée des éléments de la suite `s` indexées par `I`. Ici `I` peut être de la forme :
	`I` est entier. **Exemple** : `s[3]` renvoie le 3ème élément de `s`
	`I` est une suite. **Exemple** : `s[3:5]` renvoie les 3e, 4e et 5e éléments de `s`
	`I` est une condition. **Exemple** : `s[t>3]` renvoie les éléments de la suite `s` correspondants aux éléments de la suite `t` qui sont supérieurs à 3

`s[-I]`	Crée une suite composée des éléments de la suite `s` qui sont complémentaires à ceux indexés par `I`.
`mode(s)`	Affiche le mode (numérique, caractère,...) de la variable `s`
`length(s)`	Affiche le nombre d'éléments contenus dans la variable `s`
`names(s)`	Renvoie les noms des éléments de `s`. Si `s` est une table, renvoie les noms des colonnes. **Exemple** : `names(s)<-nom` renomme les éléments de `s` en utilisant les valeurs de la suite `nom`.
`sort(s)`	Trier les composantes d'un vecteur par ordre croissant
`rev(sort(s))`	Trier les composantes d'un vecteur par ordre décroissant
`file.choose`	`file.choose()` - sélectionne un fichier stocké sur l'ordinateur
`read.table`	`read.table("file")` - lit le fichier de données `file` ne contenant pas les noms des variables en première ligne. `read.table("file",header=T)` - lit le fichier de données `file` contenant les noms des variables en première ligne. **Exemple** : `t <- read.table$("table.txt",header=T)`
`read.csv`	`read.csv("file")` - lit un fichier de données `file` au format CSV
`read.xls`	`read.xls("file")` - lit un fichier de données `file` au format excel
`write.xls`	`write.xls("file")` - écrit un fichier de données `file` au format excel
`tab$col`	Renvoie la suite composée des éléments de la colonne `col` de la table `tab`. **Exemple** : `e<-amis$email` initialise la variable `e` avec les valeurs de la colonne `email` de la table `amis`.

Attention pour utiliser les fonctions `read.xls` et `write.xls`, il faut au préalable charger le package `xlsReadWrite`.