T. D. n° 0
Initiation à R

1 Introduction : Qu'est-ce-que R ?

- R est un logiciel permettant de faire des analyses statistiques et de produire des graphiques.
- Nous allons utiliser R comme une boîte à outils pour faire des analyses statistiques.
- Mais R est également un langage de programmation complet. C'est cet aspect qui fait que R est différent des autres logiciels statistiques.
- Les informations sur R sont disponibles sur la page d'accueil du projet :
  `http://www.r-project.org/`
  C'est le premier résultat pour la recherche de la lettre R avec le moteur de recherche google et la meilleure source d'informations sur le logiciel R. Vous pourrez y trouver les différentes distributions du logiciel, de nombreuses bibliothèques de fonctions et des documents d'aide.
- Enfin, R est un clone gratuit du logiciel S-Plus commercialisé par MathSoft et développé par Statistical Sciences autour du langage S (conçu par les laboratoires BELL).

2 Comment se procurer le logiciel R ?

Le logiciel R est gratuit. La page officielle du logiciel est :
`http://www.r-project.org/`
Si vous avez un ordinateur à la maison, vous pouvez le télécharger à l'adresse :
`http://mirror.ibcp.fr/pub/CRAN/`

3 Remarques d'ordre général sur R :

- Bien sûr il existe une version française du logiciel R.
- R fonctionne avec plusieurs fenêtres sous Windows. En particulier, nous distinguons la fenêtre `R Console`, fenêtre principale où sont réalisées par défaut les entrées de commandes et sorties de résultats en mode texte. À celle-ci peuvent s'ajouter un certain nombre de fenêtres facultatives, telles que les fenêtres graphiques et les fenêtres d'informations (historique des commandes, aide, visualisation de fichier, etc...), toutes appelées par des commandes spécifiques via la console.
- Le menu `File` ou `Fichier` contient les outils nécessaires à la gestion de l'espace de travail, tels que la sélection du répertoire par défaut, le chargement

de fichiers sources externes, la sauvegarde et le chargement d'historiques de commandes, etc.
- Le menu `Edit` ou `Edition` contient les habituelles commandes de copier-coller, ainsi que la boîte de dialogue autorisant la personnalisation de l'apparence de l'interface.
- Le menu `Misc` traite de la gestion des objets en mémoire et permet d'arrêter une procédure en cours de traitement.
- Le menu `Packages` automatise la gestion et le suivi des librairies de fonctions, permettant leur installation et leur mise à jour de manière transparente au départ du site `CRAN (Comprehensive R Archive Network)` `http://cran.r-project.org/` ou de toute autre source locale.
- Enfin, les menus `Windows` ou `Fenêtres` et `Help` ou `Aide` assument des fonctions similaires à celles qu'ils occupent dans les autres applications Windows, à savoir la définition spatiale des fenêtres et l'accès à l'aide en ligne et aux manuels de références de `R`.
- Ce qui est entré par l'utilisateur figure en rouge, et la réponse de `R` est en bleu.
- Les nombres entre crochets au début de chaque ligne donnent l'indice du premier nombre de la ligne.

4 Objectif de ce T.D.

Ce T.D. a pour objectif de vous montrer les commandes de base de `R` (ouverture, fermeture, sauvegarde, aide,...) et de vous faire manipuler des tableaux de données (saisie sous `R`, importation de fichier de données,...).

5 Pour commencer avec `R`

a) **Démarrer `R` :**
Vous lancez le logiciel `R` en cliquant sur l'icône `R`. Le symbole `>` signifie que `R` est prêt à travailler. Il ne faut pas taper ce symbole au clavier car il est déjà présent en début de ligne sur la `R Console`. C'est à la suite de ce symbole `>` que vous pourrez taper les commandes `R`. Une fois la commande tapée, vous devez toujours la valider par la touche Entrée.

b) **Quitter `R` :**
Pour quitter `R`, vous utilisez la commande :
```
> q()
```
La question `Save workspace image? [y/n/c]` est posée. `R` propose de sauvegarder le travail effectué. Trois réponses possibles :
— `y` (pour yes, oui),
— `n` (pour no, non),
— `c` (pour cancel, annuler).

En tapant `c`, la procédure de fin de session sous `R` est annulée. Si vous tapez `y`, cela permet que les commandes tapées pendant la session soient conservées en mémoire et soient donc « rappelables » (mais vous ne pouvez pas les imprimer).

c) **Sauvegarder sous `R` :**
Si vous quittez `R` en choisissant la sauvegarde de l'espace de travail, deux fichiers sont créés :

(i) le fichier `.Rdata` contient des informations sur les variables utilisées ,

(ii) le fichier `.Rhistory` contient l'ensemble des commandes utilisées.

d) **Travailler avec `R` :**
Par exemple, tapez la commande suivante et validez :
```
> 2 + 5
```
Le résultat s'affiche sous la forme :
```
[1] 7
```
Le chiffre 1 entre crochets indique l'indice du premier élément de la ligne [1] , le second chiffre est le résultat de l'opération demandée.

e) **Consulter l'aide de `R` :**
Pour toutes les commandes, vous pouvez consulter une fiche de documentation en tapant, par exemple pour la commande `read.table` :
```
>?read.table
```
Faites défiler le texte avec la touche « Entrée » ou « Flèche vers la bas ». Une fois arrivé à « END », tapez `q`. Grâce à cette aide, il suffit de retenir le nom de la commande, mais pas toute la syntaxe.

Vous pouvez rappeler les commandes déjà exécutées (pendant cette séance) en utilisant la touche « Flèche vers le haut ».

6 Rentrer des données sous `R`

Différentes commandes sont disponibles pour saisir des données sous `R`.

6.1 Affectation

Un objet peut être créé avec l'opérateur « assigner » ou « affecter » qui s'écrit `<-` :
```
> n<-15
> N<-12
```
Pour vérifier le contenu d'un objet, vous tapez son nom.
Exemple pour `n` :
```
> n
```

1. Par exemple dans le résultat suivant l'indice de l'élément 123 est 1 et celui de 142 est 20

```
[1]   123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141
[20]  142 143 144 145
```

```
[1] 15
```

Remarques :

- `R` différencie les lettres minuscules et les lettres majuscules.
- Quand vous assignez un nom à un objet, l'affichage de cet objet n'est plus automatique, il faut le demander en tapant simplement le nom donné à l'objet.
- Le signe underscore a la même fonction que le signe `<-` ; il est donc déconseillé d'utiliser le signe underscore dans le nom des variables.
- En fait, il faut remarquer que le signe `=` marche également pour faire des affectations. Essayez :

```
> a=3
> a
3
```

Choisissez donc la manière que vous voulez pour affecter !

6.2 Suite

Premier exemple. Vous souhaitez créer la suite d'entiers de 1 à 12 :
Première façon

```
> suite <- 1:12
> suite
[1] 1  2  3  4  5  6  7  8  9  10  11  12
```

Seconde façon
La fonction `seq` crée une suite (séquence) de nombres et possèdent trois arguments : `from, to` et `by`.

```
> seq(from=1,to=12,by=1)
[1] 1 2 3 4 5 6 7 8 9 10 11 12
```

Remarque : Vous pouvez aussi écrire simplement :

```
> seq(1,12,1)
[1] 1 2 3 4 5 6 7 8 9 10 11 12
```

Second exemple. Vous souhaitez créer un vecteur formé par les éléments d'une suite arithmétique de premier terme `20`, de dernier terme `40` et de raison `5`, vous pouvez encore utiliser la fonction `seq` :

```
> seq(from=20,to=40,by=5)
[1] 20  25  30  35  40
```

6.3 Combinaison ou vecteur

Il est possible de saisir une série de valeurs numériques, caractères ou logiques.

Premier exemple.

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

`serie1` est un vecteur numérique. Comment le savoir ?
Tapez la commande `class` ou `mode` sur le nom de votre vecteur.

Nous reviendrons sur `mode` au paragraphe suivant.

```
> serie1
[1] 1.20  36.00  5.33  -26.50
```

Que remarquez-vous ?

Deuxième exemple.

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

`serie2` est un vecteur de chaînes de caractères

```
> serie2
[1] "bleu" "vert" "marron"
```

Remarque : Si un vecteur est composé de caractères et de nombres, le vecteur sera un vecteur de chaînes de caractères. Quand les composantes du vecteur sont des chaînes de caractères, il est obligatoire de les déclarer entre guillemets, sinon `R` ne reconnaît pas les composantes du vecteur :

```
> serie2<-c(bleu,vert,marron)
Error : Object "bleu" not found
```

Troisième exemple.

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

`serie3` est un vecteur logique

```
> serie3
[1] TRUE  TRUE  FALSE  FALSE  TRUE
```

Quatrième exemple.

Lors d'une étude statistique, il peut arriver que certaines données ne soient pas disponibles : nous disons que **la donnée est manquante**. Pour saisir une donnée manquante, vous devez utiliser le symbole `NA` (Not Available) que l'objet soit numérique, caractère ou logique :

```
> serie4<-c(1.2,36,NA,-26.5)
```

la 3ième valeur est laissée en valeur manquante

```
> serie4
[1] 1.20  36.00  NA  -26.50
```

7 Manipuler des vecteurs

Plusieurs opérations sont possibles sur les vecteurs : concaténation, extraction, calculs, répétition, légende et tri.

7.1 Éléments d'un vecteur

Il est possible de demander l'affichage d'un (ou de plusieurs) élément(s) d'un vecteur en spécifiant entre crochets, en plus du nom du vecteur, l'indice de l'élément du vecteur. **Exemple :** Pour afficher le troisième élément de `serie1`, vous tapez la ligne de commande suivante :

```
> serie1[3]
[1] 5.33
```

Exemple : Pour afficher le troisième et le quatrième éléments de `serie1`, vous tapez la ligne de commande suivante :

```
> serie1[3:4]
[1] 5.33   -26.50
```

7.2 Concaténer deux vecteurs

Il est possible de concaténer deux vecteurs (formés de variables de même type) pour en former un nouveau :

```
> x<-c(2.3,3.5,6,14,12)
> y<-c(3.2,5,0.7,1,3.5)
> z<-c(x,y)
> z
[1] 2.3  3.5  6.0  14.0  12.0  3.2  5.0  0.7  1.0  3.5
```

7.3 Extraire des données d'un vecteur

Il est possible d'extraire des données à partir d'un vecteur selon trois façons :

- **Première façon :** Utiliser un vecteur pour préciser le numéro d'ordre des composantes à extraire. Ainsi pour extraire la deuxième et la cinquième composantes du vecteur `x` :
  ```
  > x[c(2,5)]
  [1] 3.5 12.0
  ```
- **Deuxième façon :** L'utilisation du signe tiret permet de supprimer des composantes, par exemple pour supprimer la deuxième et la troisième composantes du vecteur `x` :
  ```
  > x[-c(2,3)]
  [1] 2.3 14.0 12.0
  ```
- **Troisième façon :** Utiliser un vecteur formé de valeurs logiques. **Exemple :** Pour obtenir un vecteur ne contenant que les composantes supérieures à 4, vous pouvez utiliser la commande :
  ```
  > x[x>4]
  [1] 6 14 12
  ```

Si vous disposez de deux vecteurs ayant le même nombre de composantes, vous pouvez demander à afficher les valeurs de l'un pour lesquelles les valeurs de l'autre sont supérieures (ou inférieures) à une certaine valeur. Par exemple, les vecteurs `x` et `y` sont composés de `5` valeurs. Vous pouvez demander d'extraire de `y` les valeurs de `y` pour lesquels `x` est supérieur à `4` en utilisant la ligne de commande suivante :

```
> y[x>4]
[1] 0.7 1.0 3.5
```

7.4 Faire des calculs sur les composantes d'un vecteur

`R` peut faire des calculs sur l'ensemble des composantes d'un vecteur :

```
> 20+x*5
```

```
[1] 31.5  37.5  50.0  90.0  80.0
> (x+y)/2
[1] 2.75  4.25  6.50  12.00  12.75
```

7.5 Remplacer des données dans un vecteur

Il est possible de remplacer certaines composantes d'un vecteur par de nouvelles valeurs.
Considérez une suite de valeurs numériques notée `x` :

```
> x<-1:10
```

Premier exemple :
Si vous voulez remplacer la troisième valeur de `x` par 35, vous utiliserez alors la ligne de commande suivante :

```
> x[3]<-35
```

puis vous demanderez à `R` d'afficher le résultat

```
> x
[1] 1  2  35  4  5  6  7  8  9  10
```

Deuxième exemple :
Si vous voulez remplacer la valeur `1` par la valeur `25`, vous utiliserez alors la ligne de commande suivante :

```
> x[x==1]<-25
```

puis vous demanderez à `R` d'afficher le résultat

```
> x
[1] 25  2  35  4  5  6  7  8  9  10
```

Troisième exemple :
Si vous voulez remplacer toutes les valeurs supérieures ou égales à 5 par 20, vous utiliserez alors la ligne de commande suivante :

```
> x[x>=5]<-20
```

puis vous demanderez à `R` d'afficher le résultat

```
> x
[1] 20  2  20 4 20 20 20 20 20 20
```

7.6 Répéter les données d'un vecteur

La fonction `rep` a deux arguments `x` et `times` et crée un vecteur où `x` est répété `times` fois.

Premier exemple :
Vous créez une variable `donnees` par :

```
> donnees<-c(1,2,3)
```

Si vous voulez qu'un nouveau vecteur contienne deux fois le vecteur `donnees`, alors vous écrirez :

```
> rep(x=donnees,times=2)
```

Deuxième exemple :
Vous pouvez également demander qu'un vecteur contienne 50 fois la valeur 1 :

```
> rep(1,50)
```

Troisième exemple :
ou 4 fois la chaîne de caractères « chien » :
```
> rep("chien",4)
```

7.7 Nommer les composantes d'un vecteur

Il est possible de donner un nom à chaque composante d'un vecteur.
Exemple
Le vecteur `notes.Jean` contient les notes obtenues par Jean en Anglais, en Informatique et en Biologie.

Première façon :
Vous pouvez utiliser la commande :
```
> notes.Jean<-c(Anglais=12,Informatique=19.5,Biologie=14)
```
Afficher le vecteur `notes.Jean` en tapant la commande :
```
> notes.Jean
```
Vous obtenez le résultat suivant :
```
  Anglais Informatique  Biologie
     12.0         19.5       14.0
```

Seconde façon :
Une autre façon de nommer les composantes d'un vecteur est de définir un vecteur formé de chaînes de caractères, puis utiliser la fonction `names` :
```
> matiere<-c("Anglais","Informatique","Biologie")
> matiere
> note <- c(12,19.5,14)
> note
[1] 12.0  19.5  14.0
> names(note)<-matiere
> note
```
```
  Anglais Informatique  Biologie
     12.0         19.5       14.0
```

Remarque : Pour supprimer les noms :
```
> names(note) <- NULL
```

7.8 Trier les composantes d'un vecteur

Vous pouvez trier les composantes d'un vecteur par ordre croissant en utilisant la fonction sort.
Retour à l'exemple précédent :
Pour trier les notes dans l'ordre croissant :
```
> sort(note)
```
ou dans l'ordre décroissant :
```
> rev(sort(note))
```