Master 1 GSI - Mentions ACCIE et RIM - ULCO, La Citadelle, 2012/2013

Mesures et Analyses Statistiques de Données - Probabilités

TP 1 - Initiation au logiciel R

1 Introduction - Qu'est-ce que le logiciel R ?

R est un logiciel permettant de faire des analyses statistiques et de produire des graphiques. 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 homepage du projet :

http ://www.r-project.org/

R est un clône gratuit du logiciel S-Plus commercialisé par MathSoft et développé par Statistical Sciences autour du langage S.

• R fonctionne avec plusieurs fenêtres sous Windows. En particulier nous distinguons la fenêtre R console, c'est-à-dire la 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 fichiers, ...), toutes appelées par des commandes spécifiques via la console.

• Le menu File ou Fichier contient les outils nécesaires 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,...

• Le menu Edit ou Edition contient les commandes habituelles 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 en ligne et aux manuels de références du logiciel R.

• Ce qui est entré par l'utilisateur figure en rouge et la réponse de R est en bleu.

• Les nombre entre crochets au début de chaque ligne donnent l'indice du premier nombre de la ligne.

2 Objectifs de ce TP et commencements

Ce TP 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 fichiers de données,...).

(a) **Démarrer** R : On lance le logiciel R en cliquent 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 R console. C'est à la suite de ce symbole > que vous pourrez taper les commandes R . Une fois la commande tapée, vous devez 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 sont possibles : `y` (pour yes), `n` (pour no) ou `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 on ne peut 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 `R.data`, qui contient des informations sur les variables utilisées,

(ii) le fichier `R.history`, qui contient l'ensemble des commandes utlisées.

(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
```

Faire défiler le texte avec la touche « Entrée » ou « Flèche vers le bas ». Une fois arrivé à END, taper `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 ».

3 Rentrer des données sous `R`

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

3.1 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, taper son nom, par exemple pour `n` :

```
>n
[1] 15
```

Remarque 3.1
• `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 de l'objet.
• En fait, il faut remarquer que le signe `=` convient également pour faire des affectations. Essayez :

```
>a=3
>a
[1] 3
```

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

3.2 Suite

Exemple 3.1 Si vous souhaitez créer la suite d'entiers de 1 à 12, on peut procéder de la sorte :

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

On peut également utiliser la fonction `seq`, qui crée une suite (séquence) de nombres et possède 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
```

On peut également écrire plus simplement :

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

Exemple 3.2 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
```

3.3 Combinaison ou vecteur

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

Exemple 3.3 `>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 ?

Exemple 3.4 `>serie2<-c("bleu","vert","marron")`

`serie2` est un vecteur de chaînes de caractères.

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

Remarque 3.2 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 guillements, sinon `R` ne reconnaît pas les composantes du vecteur :

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

Exemple 3.5 `>serie3<-c(T,T,F,F,T)`

`serie3` est un vecteur logique.

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

Exemple 3.6 Lors d'une étude statistique, il peut arriver que certaines données ne soient pas disponibles : la donnée est dite manquante. Pour saisir une donnée manquante, il est conseillé d'utiliser le symbole `NA` (Not Available), quelle que soit la nature de l'objet : numérique, caractère ou logique :

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

La troisième valeur est manquante.

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

3.4 Mode et longueur

Les objets sont caractérisés par deux attributs : le mode et la longueur.

- Le mode est le type des éléments d'un objet. Comme nous venons de le voir, un objet peut être une valeur numérique, une chaîne de caractères ou une valeur logique.
- La longueur est le nombre d'éléments de l'objet.

Par exemple, si vous saisissez une série d'observations obtenues sur un échantillon sous la forme d'un vecteur, la longueur de ce vecteur correspondra à la taille de l'échantillon. Pour connaître le mode et la longueur d'un objet, vous utiliserez les fonctions `mode` et `length` :

```
>mode(serie1)
[1] "numeric"
>mode(serie2)
[1] "character"
>mode(serie3)
[1] "logical"
>length(serie1)
[1] 4
>length(serie2)
[1] 3
>length(serie3)
[1] 5
```

3.5 Saisie au clavier d'un jeu de données

En utilisant la fonction `scan()`, la saisie d'une série de données peut paraître moins fastidieuse.

```
>jeu1<-scan()
```

`R` vous redonne la main et vous pouvez taper les valeurs du jeu de données :

```
1 :1.2
2 :36
3 :5.33
4 :-26.5
5 :le premier retour-chariot après une chaîne vide met fin à la saisie.
>jeu1
[1] 1.20 36.00 5.33 -26.50
```

3.6 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. Par exemple, pour afficher respectivement le troisième élément de `serie1` et ses troisième et quatrième éléments, on utilisera :

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

4 Manipuler des vecteurs

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

4.1 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
```

4.2 Extraire des données d'un vecteur

Il est possible d'extraire des données d'un vecteur.

1. Utiliser un vecteur pour préciser le numéro d'ordre des composantes à extraire. Ainsi, pour extraire les 2ème et 5ème composantes du vecteur `x` :

   ```
   >x[c(2,5)]
   [1] 3.5 12.0
   ```

2. L'utilisation du signe « tiret »permet de supprimer des composantes, par exemple pour supprimer les 2ème et 3ème composantes du vecteur `x` :

   ```
   >x[-c(2,3)]
   [1] 2.3 14.0 12.0
   ```

3. Utiliser un vecteur formé de valeurs logiques. Par 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 à extraire de `y` les valeurs de `y` pour lesquelles `x` est supérieur à 4 en utilisant la ligne de commandes suivante :

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

4.3 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] 17.125 20.875 28.250 51.000 46.375
```

4.4 Remplacer des données dans un vecteur

Il est possible de remplacer certaines composantes d'un vecteur par de nouvelles valeurs.
Considérons une suite de valeurs numériques :

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

Exemple 4.1 Si vous voulez remplacer le 3ème valeur de `x` par 35, vous utiliserez la ligne de commandes 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
```

Exemple 4.2 Si vous voulez remplacer la valeur 1 par la valeur 25, vous utiliserez alors la ligne de commandes 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
```

Exemple 4.3 Si vous voulez remplacer toutes les valeurs supérieures ou égales à 5 par 20, vous utiliserez la ligne de commandes 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
```

4.5 Répéter les données d'un vecteur

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

Exemple 4.4 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)
```

Exemple 4.5 Vous pouvez également demander qu'un vecteur contienne 50 fois la valeur 1 :

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

Exemple 4.6 ou 4 fois la chaîne de caractères "chien" :

```
>rep("chien",4)
```

4.6 Nommer les composantes d'un vecteur

Il est possible de donner un nom à chaque composante d'un vecteur.

Exemple 4.7 Le vecteur `notes.Jean` contient les notes obtenues par Jean en anglais, informatique et biologie.

- Première façon : vous pouvez utiliser la commande :

  ```
  >notes.Jean<-c(anglais=12,informatique=19.5,biologie=14)
  ```

  Affichez ensuite le vecteur `notes.Jean`, on obtient le résultat suivant :

  ```
   anglais  informatique  biologie
   12.0     19.5          14.0
  ```

- Seconde façon : on peut nommer les composantes d'un vecteur en définissant un vecteur formé de chaînes de caractères puis en utilisant le fonction `names` :

  ```
  >matiere<-c("anglais","informatique","biologie")
  >matiere
  >note<-c(12,19.5,14)
  >names(note)<-matiere
  >note
   anglais  informatique  biologie
   12.0     19.5          14.0
  ```

Remarque 4.1 Pour supprimer les noms, on tapera la ligne de commandes :

```
>names(note)<-NULL
```

4.7 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 4.7 :

```
>sort(note)
[1] 12.0 14.0 19.5
```

ou dans l'ordre décroissant :

```
>rev(sort(note))
[1] 19.5 14.0 12.0
```

5 Lire des données dans un fichier

Quand les données sont plus volumineuses, il n'est pas très conseillé d'utiliser `R` comme outil de saisie. Dans ce cas, vous pouvez utiliser un éditeur de texte ou un tableur quelconque pour saisir vos données (Excel par exemple) et le transférer ensuite sous `R`.
Il est nécessaire d'indiquer au logiciel `R` l'endroit où sont stockés les fichiers de données. Ceci peut être fait soit à chaque chargement de fichier soit pour la durée de chaque utilisation du logiciel.
Pour connaître le répertoire de travail actuellement utilisé par `R`, qui est par défaut le répertoire où le logiciel est installé, il suffit de taper l'instruction suivante :

```
>getwd()
```

Pour changer le répertoire de travail par défaut, pour la durée de la session `R`, pour par exemple le répertoire "C :/Data", il suffit de taper :

```
>setwd("C :\\Data")
```

ou de manière équivalente

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

Pour des raisons liées à la syntaxe de `R`, plus précisément la syntaxe des systèmes Unix, la barre oblique inversée « \ » a été remplacée soit par une barre oblique « / » soit par deux barres obliques inversées "\\".

Dans le dossier relatif au TP 1, vous trouverez des fichiers au format texte (.txt) `table1.txt, table2.txt, table3.txt, table4.txt`, des fichiers au format csv (.csv) `table5.csv` et `table6.csv` et enfin des fichiers au format Excel (.xls) `table7.xls` et `table8.xls`. Faites devenir ce répertoire le répertoire de travail par défaut de `R`.

Les données suivantes ont été saisies dans le fichier `table1.txt` :
53.5 160
74.4 172
52.6 151
88.6 163
49.2 169

Ces données sont accessibles grâce à la ligne de commandes :

```
>read.table("table1.txt")
```

`R` affiche alors le tableau de données en numérotant les lignes et les colonnes, les lignes correspondant aux individus et les colonnes aux variables. `R` affiche un message d'avertissement concernant le nom des variables.
Vous pouvez également conserver la table comme un objet pour pouvoir la réutiliser directement :

```
>tab<-read.table("table1.txt")
```

et demander l'affichage de cet objet :

```
>tab
```

ou seulement d'une colonne de cet objet :

```
>tab$V1
```

ou seulement de l'élément de la première ligne et de la première colonne :

```
>tab[1,c(1)]
```

ou

```
>tab[1,1]
```

ou les éléments des deux premières lignes et de la première colonne :

```
>tab[1 :2,1]
```

ou les éléments des deux premières lignes et des deux premières colonnes :

```
>tab[1 :2,1 :2]
```

Pour travailler ensuite sur les variables de la table, vous pouvez leur attribuer un nom (plus simple que la syntaxe utilisée) :

```
>V1<-tab$V1
>V2<-tab$V2
```

Si vous avez spécifié le nom des variables de la table, dans la première ligne de votre fichier de données (comme dans le fichier `table2.txt`), vous devez l'indiquer par l'option `header=TRUE` ou `header=T` :

```
> read.table("table2.txt",header=T)
```

Par défaut `R` lit la première ligne comme une ligne de données et nomme les colonnes sous la forme `V1,V2,...` (comme pour `table1.txt`).
Par défaut un point (.) est utilisé pour les décimales. Mais si les décimales sont notée par une virgule dans votre fichier de données (comme dans `table3.txt`), il faut le spécifier par :

```
>read.table("table3.txt",dec=",")
```

Par défaut un espace est utilisé pour séparer les valeurs appartenant à différentes colonnes. Mais si les colonnes sont séparées par un point virgule dans votre fichier de données (comme dans `table4.txt`), il faut le spécifier par :

```
>read.table("table4.txt",sep=",")
```

Pour ouvrir un fichier de données sans avoir à indiquer son emplacement en utilisant une boîte de dialogue conviviale :

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

Enfin la plupart des formats de fichiers sont connus par `R`.
Travaillons par exemple avec les fichiers csv. Il en existe deux types :

- anglo-saxon avec un « . » comme séparateur décimal et une « , » comme séparateur de colonne et,
- français avec un « , » comme séparateur décimal et une « , » 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())
```

On traitera des fichiers Excel un peu plus tard.

6 Fichiers scripts

Il est souvent plus pratique de composer le code `R` dans une fenêtre spécifique du logiciel : la fenêtre de script. Les entrées `Nouveau script` et `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`.

Exercice 1

1. Créer le vecteur $x = (101; 102; \ldots; 112)$.
2. Créer un vecteur de longueur 12 formé de 4 fois la suite de nombres $(4; 6; 3)$.
3. Créer un vecteur composé de huit 4, de sept 6 et de cinq 3.

Exercice 2

1. Saisir la variable `poids` contenant les 15 valeurs suivantes :

   28 ; 27.5 ; 27 ; 28 ; 30.5 ; 30 ; 31 ; 29.5 ; 30 ; 31 ; 31 ; 31.5 ; 32 ; 30 ; 30.5

2. Saisir la variable `poids1` contenant les 5 valeurs suivantes :

   40 ; 39 ; 41 ; 37.5 ; 43

3. Sans refaire de saisie, créer la variable `nouveau.poids` contenant 20 valeurs (les 5 valeurs de `poids1` répétées 2 fois et les 10 dernières valeurs de `poids`).
4. Enregistrer dans votre répertoire de travail la variable `nouveau.poids` dans une feuille nommée « Nouveau Poids » du classeur Excel « Poids.xls ».

Exercice 3

1. Créer le vecteur `nom` contenant les noms de 10 personnes.
2. Créer le vecteur `age` contenant l'âge des 10 personnes précédentes (entre 20 et 60 ans). Les noms des personnes seront utilisées comme légende pour le vecteur `age`.
3. Créer le vecteur `poids` contenant le poids des 10 personnes (entre 50 et 100 kg) en utilisant à nouveau le nom des personnes comme légende pour ce vecteur.
4. Même chose pour le vecteur `taille` contenant la taille des 10 personnes.
5. Créer le vecteur `poids.lourds` contenant le poids des personnes de plus de 80 kg.
6. Créer le vecteur `taille.poids.lourds` contenant la taille des personnes de plus de 80 kg.
7. Créer le vecteur `taille.vieux.poids.lourds` contenant la taille des personnes de plus de 80 kg et âgées de plus de 30 ans. Pour répondre à cette question, vous pourrez utiliser le connecteur logique ET dont la syntaxe est donnée dans l'aide sur opérateurs logiques accessible en tapant l'instruction `?Logic` .

Références

[1] Frédéric BERTRAND. *Initiation au logiciel R - Master Statistique 2ème année.*
http ://www-irma.u-strasbg.fr/∼fbertran/enseignement/Statistique_Master2SA_2009.html