Fiche TD avec le logiciel  : tdr12

Première session de travail

D. Chessel, A.B. Dufour & J.R. Lobry

Table des matières

1 Importer des données

Il est conseillé au débutant de noter quelques mots-clefs pour mémoriser les fonctions de première nécessité. Commencer par **getwd()**. Lancer le programme comme indiqué dans la fiche *tdr11* et vérifier que le dossier de travail est bien le votre :

```
getwd()
```

Puisque R est un logiciel de statistique autant commencer par importer ses données. Un fichier de données peut être créé à partir de plusieurs logiciels : tableur, traitement de texte, bloc-notes à condition qu'il soit sauvegardé dans le format `.txt`. Tout se passe dans le dossier de travail. Dans ce dossier de travail,

1. Créer un fichier Excel en tapant sur trois colonnes (`sex`, `poi`, `tai`) les données ci-dessous. Sauvegarder en `t3var.xls` régulièrement pour ne pas perdre votre travail.

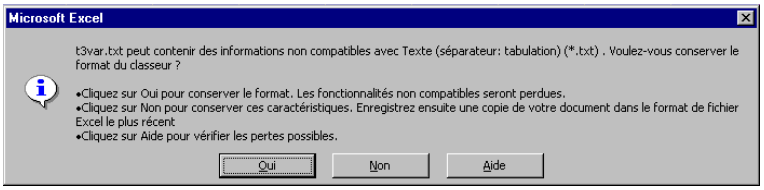
	A	B	C
1	sexe	poi	tai
2	h	60	170
3	f	57	169
4	f	51	172
5	f	55	174
6	f	50	168
7	f	50	161
8	f	48	162
9	h	72	189
10	f	52	160
11	h	64	175
12	f	53	165
13	h	72	164
14	h	61	175
15	h	78	184
16	h	68	178
17	f	51	158
18	f	53	164
19	h	79	179
20	h	74	182
21	h	62	174
22	f	49	158

	A	B	C
23	f	50	163
24	h	74	172
25	h	80	185
26	f	53	170
27	h	73	178
28	h	70	180
29	h	72	189
30	f	70	172
31	f	62	174
32	h	77	200
33	h	70	178
34	h	76	178
35	f	51	168
36	f	52	170
37	f	57	160
38	f	53	163
39	f	55	168
40	f	66	172
41	h	65	175
42	h	75	180
43	f	50	162
44	f	53	177

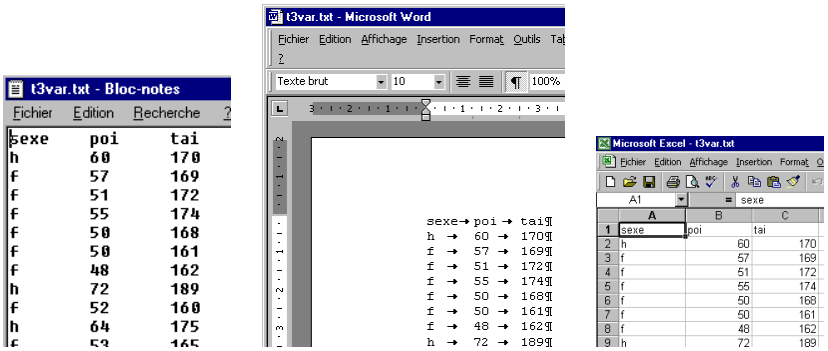
	A	B	C
45	h	55	169
46	h	55	173
47	h	72	182
48	h	75	183
49	h	73	184
50	h	71	181
51	h	66	180
52	h	71	178
53	h	79	178
54	h	62	168
55	f	47	161
56	h	73	171
57	h	72	180
58	h	60	174
59	h	67	175
60	h	85	182
61	h	73	181
62	h	82	188
63	h	86	182
64	h	85	189
65	h	65	178
66	f	47	150
67	h	74	186

2. Quand c'est fini, enregistrer à nouveau, puis utiliser la commande *enregistrer sous ...* pour sauvegarder cette information au format `Texte(séparateur : tabulation)` :

Excel vous demandera de confirmer ce que vous faites (c'est tellement extraordinaire de ne pas suivre les sentiers battus). Confirmer en répondant Oui :



3. Ouvrir le fichier créé, qui doit être dans votre dossier de travail, par un double clic pour le voir dans l'éditeur de texte, puis par *Word* pour afficher les caractères cachés, puis l'ouvrir avec *Excel* pour bien comprendre que la même information est lue et utilisée par différents programmes.



4. Retourner dans R pour le lire à nouveau.

Observer d'bord ce qui se passe quand on lit les 5 premières lignes du tableau.

```
read.table("t3var.txt")[1:5, ]
   V1  V2  V3
1 sexe poi tai
2    h  60 170
3    f  57 169
4    f  51 172
5    f  55 174
```

Le premier individu contient les en-têtes ! Rajouter une valeur du paramètre `header`. Quand le résultat semble correct, placer le résultat de cette lecture dans un objet. On dit qu'on fait une affectation. Les 3 formes d'affectation donnent le même résultat. A vous de savoir laquelle vous préférez. Noter que la flèche en haut, au clavier, rappelle la dernière ligne frappée. Il vaut mieux s'en souvenir !

```
t3var <- read.table("t3var.txt", h=T)
t3var = read.table("t3var.txt", h=T)
read.table("t3var.txt", h=T) -> t3var
```

Dans le cas précédent, le fichier à charger dans R était sauvegardé dans un répertoire local. Il est également possible de charger des données sauvegardées sur un répertoire distant et notamment sur un site internet. Un grand nombre de jeux de données sont disponibles sur le site

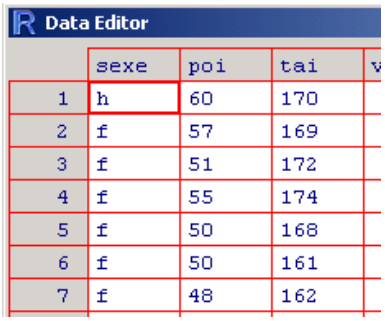
`http://pbil.univ-lyon1.fr/R/donnees/`

Visiter ce dossier, repérer le fichier `t3var`, l'importer à la main dans son dossier de travail, l'afficher dans le navigateur, l'ouvrir avec un éditeur, le télécharger directement dans son dossier. On peut même le lire directement par :

```
read.table("http://pbil.univ-lyon1.fr/R/donnees/t3var.txt", h = T)
t3var <- read.table("http://pbil.univ-lyon1.fr/R/donnees/t3var.txt",
    h = T)
```

Utiliser au choix votre fichier ou le fichier importé ou les deux avec des noms différents pour comparer. R possède un éditeur d'objets qu'on peut invoquer par :

```
edit(t3var)
```



Modifier les données et fermer la fenêtre de l'éditeur de données. Qu'observez vous ?

La fonction `edit` permet de modifier les données mais, par défaut, elle ne sauvegarde pas ces modifications. Il faut donc stocker les résultats de la fonction `edit` dans un objet :

```
t3varmodif <- edit(t3var)
```

Mais il existe également une alternative pour conserver les modifications réalisées sans créer un nouvel objet :

```
fix(t3var)
```

2 Consulter la documentation

```
?read.table
help("read.table")
```

Consulter la fiche de documentation est une opération fondamentale !

```
read.table               package:base               R Documentation

Data Input
Description:
     Reads a file in table format and creates a data frame from it,
     with cases corresponding to lines and variables to fields in the
     file.
Usage:
     read.table(file, header = FALSE, sep = "", quote = "\"'", dec = ".",
                row.names, col.names, as.is = FALSE, na.strings = "NA",
                colClasses = NA, nrows = -1,
                skip = 0, check.names = TRUE, fill = !blank.lines.skip,
                strip.white = FALSE, blank.lines.skip = TRUE,
                comment.char = "#")
```

Dans chaque fiche, on trouvera systématiquement (des programmes contrôlent ce *systématiquement* !) l'objet de la fonction, une description, l'ordre d'appel, la définition des paramètres et des exemples éventuels.

Exercice : la fonction `readLines()` permet également de lire des fichiers, par exemple :

```
readLines("http://pbil.univ-lyon1.fr/R/donnees/toto")
[1] "c'est toto"
```

Consultez la documentation de cette fonction pour comprendre en quoi elle diffère de la fonction `read.table()`.

3 Le passage des paramètres

Noter le signe = utilisé dans R pour passer les valeurs des paramètres et noter les points importants :

- ⋆ On peut passer les valeurs des paramètres dans l'ordre sans donner leur nom ;
- ⋆ On peut les passer dans le désordre en donnant leur nom ;
- ⋆ Toute abréviation non ambiguë est acceptée ;
- ⋆ La plupart des paramètres ont des valeurs par défaut.

Essayer par exemple :

```
read.table(header = T, file = "t3var.txt")[1:2, ]
read.table(hea = T, file = "t3var.txt")[1:2, ]
read.table(h = T, file = "t3var.txt")[1:2, ]
read.table(h = T, fi = "t3var.txt")[1:2, ]
```

Ne retenir que le nom de la fonction `read.table` et consulter la documentation autant que nécessaire. Quitter en sauvegardant l'espace de travail.

4 Utiliser l'historique

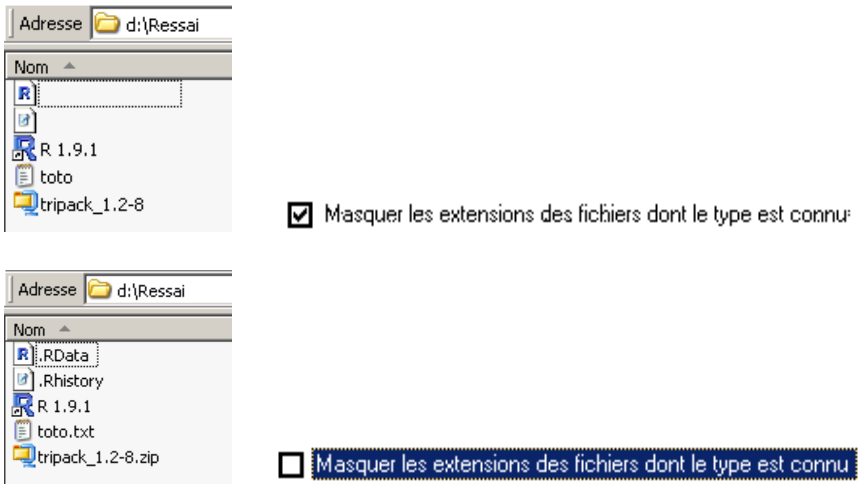
Vérifier l'existence du fichier .Rhistory. Faire en sorte qu'il s'ouvre systématiquement avec l'éditeur et l'ouvrir. Il contient l'historique des commandes utilisées et peut servir d'une session à l'autre. Relancer et vérifier que "Flèche en haut" renvoient les ordres passés.

Remarque à l'intention des débutants sous Windows.

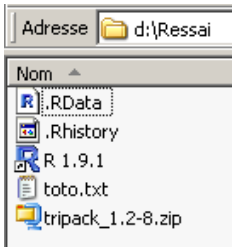
Sous Windows quelques options par défaut peuvent créer de grosses difficultés aux utilisateurs non avertis. Pour un dossier quelconque, dans le menu des dossiers, rechercher :

- ⋆ le menu *Outils* ;
- ⋆ l'option *Options des dossiers* ;
- ⋆ l'onglet *Affichage*.
- ⋆ la case à cocher *Masquer les extensions des fichiers*

Analyser la situation dans laquelle vous êtes.



Si le type `.Rdata` et `.Rhistory` est inconnu les fichiers n'apparaissent pas du tout. Utiliser donc la seconde option **systématiquement**. Si le type `.Rhistory` est inconnu, ce qu'on voit par l'icône :



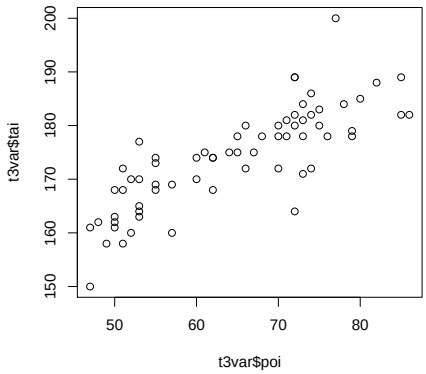
cliquer sur `.Rhistory` et sélectionner le programme (un éditeur de fichier texte) qui l'ouvrira par défaut, en observant la case :

5 Conserver les résultats

Ouvrir un fichier *Word*, définir un style listing avec la police `Courier New` pour enregistrer les listings.

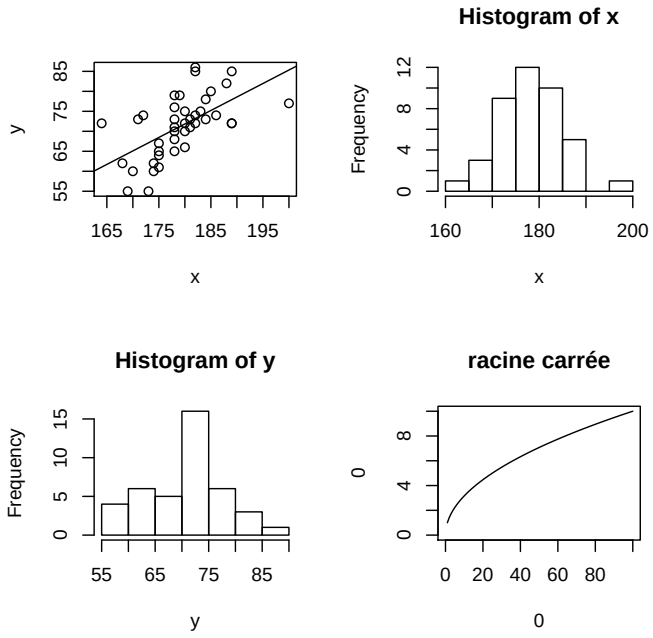
```
summary(t3var)
sexe        poi             tai
f:25   Min.   :47.00   Min.   :150.0
h:41   1st Qu.:53.00   1st Qu.:168.0
       Median :65.50   Median :174.5
       Mean   :64.52   Mean   :174.1
       3rd Qu.:73.00   3rd Qu.:180.0
       Max.   :86.00   Max.   :200.0
plot(t3var$poi, t3var$tai)
```

Copier les listings (sélection à la souris) et les figures (cliquer dessus avec le bouton droit) et coller dans un document *Word*. Vous pouvez faire un rapport.



6 Écrire une fonction

Nous voulons écrire une fonction qui doit faire ce dessin :



Créer un fichier texte dans le dossier de travail et lui donner le nom `fessai.R`. Ouvrez le dans avec le menu *Fichier* et l'option *Ouvrir un script...*. Une fenêtre apparaît et on peut intégrer du texte, par exemple :

```
fessai <- function() {
    par(mfrow = c(2, 2))
    x <- t3var$tai[t3var$sex == "h"]
    y <- t3var$poi[t3var$sex == "h"]
```

```
    plot(x, y)
    abline(lm(y ~ x))
    hist(x)
    hist(y)
    plot(0, 0, type = "n", xlim = c(0, 100), ylim = c(0, 10))
    lines(1:100, sqrt(1:100))
    title("racine carrée")
}
```

L'indentation n'est pas indispensable mais plus agréable. Vous pouvez sauvegarder ce fichier, l'ouvrir à nouveau, l'enregistrer ailleurs, comme dans un éditeur simple. Noter que la console et la fenêtre d'édition sont actives à votre choix mais partage le menu *Fichier* qui change. On peut s'en servir de plusieurs manières.

1. Avec la fenêtre de la console active, au menu *Fichier* l'option *Sourcer du code R...* indique à qu'il doit en exécuter le contenu d'un fichier. On lui propose `fessai.R`. Comme ce fichier contient la définition d'une fonction, cette fonction, si tout se passe bien, deviendra un objet de l'espace de travail.
2. On obtient le même résultat, que l'éditeur soit ouvert ou fermé, par la commande :

   ```
   source("fessai.R")
   ```

   Observer que la console a bien exécuté l'ordre :

   ```
   ls()
   [1] "RweaveInPng" "fessai"      "go"          "lll"         "qqq"
   [6] "t3var"
   ```

   La commande `ls()` fait la liste des objets de l'espace de travail. On y trouve maintenant le tableau t3var et la fonction fessai. On peut exécuter la fonction :

   ```
   fessai()
   ```

   Introduire des fautes de frappe. Observer les erreurs. Corriger le source, sauvegarder le texte, recharger le source et exécuter la fonction. Reprendre le cycle jusqu'à satisfaction. Les fichiers .R que vous pouvez toujours ouvrir avec un éditeur ordinaire serviront à conserver vos sources. Vous pouvez laisser l'éditeur ouvert et recharger le source : c'est la manière la plus efficace de faire des essais.
3. Les fenêtres de l'éditeur fonctionne également dans l'autre sens. Créer un nouveau script au menu *Fichier* par l'option *Nouveau script*. Une fenêtre s'ouvre : taper `2+2`. Avec la touche Ctrl+R ou l'option *Exécuter la ligne ou sélection* du menu contextuel (clic droit de la souris) l'ordre est recopié dans la console et exécuté. Vous pouvez ainsi exécuter tout ou partie des ordres du script : dans cette option, les ordres sont recopiés à l'écran et seront dans l'historique. Dans l'autre sens, ce n'est pas le cas. Vous vous ferez une opinion.

7 Exporter des données

Pour sortir un objet dans un fichier texte du dossier de travail :

```
dput(t3var, "toto.txt")
```

Ouvrir le fichier créé et examiner son contenu. Pour recharger ce fichier en mémoire :

```
provi <- dget("toto.txt")
```

Vérifier la présence du nouvel objet `provi` (`ls()`), éditer son contenu, le détruire (`rm()`) et vérifier sa disparition :

```
ls()
provi
rm(provi)
ls()
```

Refaire cet exercice en utilisant les ordres d'écriture et de lecture des tableaux. Ouvrir le fichier créé et examiner son contenu. Le recharger et vérifier le contenu de la mémoire. Vous savez maintenant diriger votre session de travail.

```
ls()
write.table(t3var, "toto.txt", row = F, sep = "\t")
provi <- read.table("toto.txt")
ls()
provi
rm(provi)
ls()
```

8 Commencer un catalogue personnel

Il n'y a plus qu'une difficulté : connaître et comprendre quelques unes des 2500 fonctions de base (sans compter 450 librairies additionnelles). Au début, on fera la liste des fonctions passées en revue – qu'il semble logique de mémoriser – dans un petit catalogue personnel. Il convient de ne noter que des mots-clefs, la documentation de chaque fonction étant toujours là pour les détails. Noter la fonction `apropos()` qui peut vous aider à retrouver un nom de fonctions. On retiendra `file` pour tout ce qui touche aux fichiers :

```
apropos("file")
 [1] ".CurFileName"       "bzfile"             "close.srcfile"
 [4] "download.file"      "env.profile"        "file"
 [7] "file.access"        "file.append"        "file.choose"
[10] "file.copy"          "file.create"        "file.edit"
[13] "file.exists"        "file.info"          "file.path"
[16] "file.remove"        "file.rename"        "file.show"
[19] "file.symlink"       "file_test"          "gzfile"
[22] "list.files"         "memory.profile"     "open.srcfile"
[25] "open.srcfilecopy"   "parseNamespaceFile" "print.srcfile"
[28] "profile"            "readCitationFile"   "srcfile"
[31] "srcfilecopy"        "system.file"        "tempfile"
[34] "zip.file.extract"
```

On retiendra `read` et write pour les entrées-sorties.

```
apropos("read")
 [1] ".readRDS"          "read.DIF"          "read.csv"          "read.csv2"
 [5] "read.dcf"          "read.delim"        "read.delim2"       "read.fortran"
 [9] "read.ftable"       "read.fwf"          "read.socket"       "read.table"
[13] "read.table.url"    "readBin"           "readChar"          "readCitationFile"
[17] "readLines"         "readline"
apropos("write")
 [1] ".RweaveInPngWritedoc" "RtangleWritedoc"      "RweaveLatexWritedoc"
 [4] "write"                "write.csv"            "write.csv2"
 [7] "write.dcf"            "write.ftable"         "write.socket"
[10] "write.table"          "write.table0"         "writeBin"
[13] "writeChar"            "writeLines"
```

Vous entrez alors dans la caverne d'Ali Baba.