Importer des données: la fonction read.table()

Fabrice Dessaint
Inra, UMR1347 Agroécologie, Dijon

Décembre 2015

Quelques fonctions *R* permettant l'importation de données contenues dans des fichiers avec des structures simples.

La fonction `read.table()`

Cette fonction est très générale. Sa syntaxe est donnée ci-dessous :

```
read.table(file, header = FALSE, sep = "", quote = "\"'", dec = ".", numerals = c("allow.loss",
    "warn.loss", "no.loss"), row.names, col.names, as.is = !stringsAsFactors, 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 = "#", allowEscapes = FALSE,
    flush = FALSE, stringsAsFactors = default.stringsAsFactors(), fileEncoding = "",
    encoding = "unknown", text, skipNul = FALSE)
```

Elle possède de nombreux arguments qui permettent la lecture de fichiers de formes et de structures différentes.

Pour illustrer l'usage des principaux arguments de cette fonction, on va utiliser le fichier `Mesure.txt`. Ce fichier contient les valeurs de 4 variables (dont le nom n'est pas fourni) pour 7 observations.

Dans ce fichier, les différentes valeurs sont séparées par le caractère « : ». Le fichier contient des valeurs manquantes que l'on a signalées par la valeur numérique « -99 ». On trouve aussi des commentaires (information ne devant pas être lue par *R*). Ils sont précédés par le caractère « ! ». Enfin, on a indiqué le nom des individus mesurés à la fin de chaque ligne.

La lecture de ce fichier va se faire avec l'instruction suivante :

```
Mesure <- read.table(file="FichiersEx/Mesure.txt", header=FALSE,
                     sep=":", dec=",", comment.char="!",
                     na.string="-99")
```

```
! mesuré le 15/03/2010
12,3:5,2:7,1:pom2 ! à vérifier
11,3:5,2:7:pom3
10,2:5,5:7:poi1
! mesuré le 16/03/2010
11,9:6:7,2:poi2
10,8:6,9:-99:pom1
14,2:8,0:7,9:ban1
```

Le fichier `Mesure.txt`

L'argument `file`

C'est le seul argument obligatoire de la fonction ; les autres arguments possèdent des valeurs par défaut. Il renseigne sur le nom et la localisation du fichier que l'on souhaite lire dans *R*.

Le fichier peut être localisé sur une machine distance (internet) ; dans ce cas, on fournit une URL à la fonction

Le nom doit être indiqué sous la forme d'une chaîne de caractères entourée de guillemets simples (`'Mesure.txt'`) ou doubles (`"Mesure.txt"`).

La localisation sur la machine locale (le chemin) peut être absolue ou relative et dans ce cas, c'est le répertoire de travail qui sert de point de référence. L'utilisation de la contre oblique (ou antislash « \ ») dans l'écriture du chemin

est interdite et on doit la remplacer par la barre oblique (ou slash « / », même sous le système d'exploitation `Windows`).

```
Mesure <- read.table(file="FichiersEx/Mesure.txt", ...
# ou
fichier <- "FichiersEx/Mesure.txt"
Mesure <- read.table(file=fichier,
```

L'objet `Mesure` est un tableau ou `data.frame`.

L'argument `header=FALSE`

Cet argument permet de préciser que la première ligne qui sera utilisée [1], contient (`header=TRUE`) ou non (`header=FALSE`) le nom des variables qui vont constituer les colonnes du tableau qui sera construit par *R*.

Par défaut, l'argument a la valeur `FALSE` et *R* considère que les données commencent dès la première ligne utile. Le nom des variables est construit de façon automatique selon le motif `V1`, `V2`, ...

1. Ce n'est pas forcément la première ligne du fichier, voir l'argument `skip`

```
Mesure <- read.table(file="FichiersEx/Mesures.txt", header=FALSE, ...
names(Mesure)
```

```
[1] "V1" "V2" "V3" "V4"
```

Si l'argument prend la valeur `TRUE`, les informations de la première ligne lue sont utilisées pour nommer les variables. *R* procède, par défaut, à une vérification de la conformité des noms (voir l'argument `check.names`).

L'argument `sep=""`

Cet argument précise le caractère utilisé pour séparer les valeurs prises par les différentes variables. Plusieurs caractères peuvent être utilisés pour séparer les colonnes du fichier.

Le séparateur par défaut est le blanc qui correspond à n'importe quel espace blanc : un ou plusieurs espaces, la tabulation, la caractère de fin de ligne (LF) ou celui de retour charriot (CR).

Dans le fichier `Mesure.txt`, les valeurs sont séparées par le caractère « : ».

```
Mesure <- read.table(file="FichiersEx/Mesures.txt", header=FALSE,
                     sep=":", ...
```

Attention : Lorsque l'on a des données manquantes, représentées par des blancs, il est fortement recommandé d'utiliser un séparateur explicite comme le point-virgule « ; » par exemple.

L'argument `dec="."`

Permet de spécifier le caractère utilisé pour séparer la partie décimale d'un nombre de sa partie entière. Pour des données issues de tableurs français, le caractère généralement utilisé est la virgule (`,`). C'est le cas dans l'exemple utilisé.

```
Mesure <- read.table(file="FichiersEx/Mesures.txt", header=FALSE,
                     sep=":", dec=",", ...
```

On ne peut spécifier qu'un seul caractère. Il faut donc éviter le mélange d'une notation avec la virgule et le point décimal.

`comment.char="#"`

On indique à *R*, via cet argument, le caractère utilisé pour commencer une ligne de commentaire ou d'information ne devant pas être lue par *R*. Dès que ce caractère apparait sur une ligne, la lecture s'arrête pour cette ligne.

Les lignes ne contenant que des commentaires sont considérées comme des lignes vides qui sont par défaut supprimées du tableau (voir l'argument `blank.lines.skip`).

Le fichier `Mesure.txt` contient 9 lignes mais le tableau `Mesure` ne contient que `7` observations (présence de 2 lignes de commentaire). Dans ce fichier, le début d'un commentaire est indiqué par la caractère « ! ».

```
Mesure <- read.table(file="FichiersEx/Mesure.txt", header=FALSE,
                     sep=":", dec=",", comment.char="!", ...
```

L'argument `na.strings="NA"`

C'est le dernier argument utilisé pour lire le fichier `Mesure.txt`. Il indique la façon dont sont codées les données manquantes. On peut spécifier plusieurs valeurs, sous la forme d'un vecteur de chaînes de caractères.

```
Mesure <- read.table(file="FichiersEx/Mesure.txt", header=FALSE,
                     sep=":", dec=",", comment.char="!",
                     na.strings="-99")
```

Pour les variables de type logique et numérique, l'absence de valeurs est interprétée comme une valeur manquante.

LE FICHIER A ÉTÉ LU et le tableau `Mesure` contient 4 variables et 7 observations :

```
Mesure
    V1  V2  V3    V4
1 12.3 5.2 7.1 pom2
2 11.3 5.2 7.0  pom3
3 10.2 5.5 7.0  poi1
```

```
4 11.9 6.0 7.2  poi2
5 10.8 6.9  NA  pom1
6 14.2 8.0 7.9  ban1
7   NA  NA 8.2  ora1
```

Les variables ont été nommées par *R* et les observations sont identifiées par un numéro. Or on sait que le nom des observations est contenu dans la dernière variable. On peut donc l'utiliser puis la supprimer du tableau.

```
row.names(Mesure) <- Mesure$V4
Mesure <- Mesure[,-4]
```

On peut aussi le faire directement lors de la lecture du fichier avec l'argument `row.names`.

De la même façon, on peut souhaiter modifier le nom des variables pour qu'il soit plus informatif.

```
names(Mesure) <- c("Hauteur","Epaisseur","Largeur","Id")
```

Cette opération peut aussi être effectuée lors de la lecture du fichier avec l'argument `col.names`.

L'argument `col.names`

Permet de renommer les variables/colonnes du tableau. Le nom des variables est contenu dans un vecteur ou donné directement à l'argument. Le vecteur doit avoir une longueur égale au nombre de colonnes du fichier (y compris une éventuelle colonne d'identification des lignes).

```
nom.var <- c("Hauteur","Epaisseur","Largeur","Id")
Mesure <- read.table(file="FichiersEx/Mesure.txt", header=FALSE,
                     sep=":", dec=",", comment.char="!",
                     na.string="-99", col.names=nom.var)
names(Mesure)
[1] "Hauteur"   "Epaisseur" "Largeur"   "Id"
```

L'argument `row.names`

Cet argument permet de nommer les lignes du tableau. Le nom des lignes peut être un vecteur construit préalablement ou le numéro ou le nom d'une des colonnes du tableau en cours de création. Les noms des lignes doivent être uniques.

```
Mesure <- read.table(file="FichiersEx/Mesure.txt", header=FALSE,
                     sep=":", dec=",", comment.char="!",
                     na.string="-99", col.names=nom.var,
                     row.names=4)
row.names(Mesure)
[1] "pom2 " "pom3"  "poi1"  "poi2"  "pom1"  "ban1"  "ora1"
```

Si cet argument est manquant, les lignes sont simplement numérotées.

Seules trois variables ont été conservées dans le tableau : la quatrième ayant servi à nommer les lignes a été supprimée.

```
names(Mesure)
[1] "Hauteur"   "Epaisseur" "Largeur"
```

L'argument `skip=0`

Cet argument permet de commencer la lecture des données après avoir « ignoré » un certain nombre de lignes du fichier. Supposons que sur le fichier `Mesure.txt`, on ne soit intéressé que par les observations faites après le 16/03/2010 (soit les lignes de 6 à la fin du fichier).

```
Mesure2 <- read.table(file="FichiersEx/Mesure.txt", header=FALSE,
                      sep=":", dec=",", comment.char="!",
                      na.string="-99", col.names=nom.var,
                      row.names=4, skip=5)
Mesure2
     Hauteur Epaisseur Largeur
poi2    11.9       6.0     7.2
pom1    10.8       6.9      NA
ban1    14.2       8.0     7.9
ora1      NA        NA     8.2
```

Selon, le système d'exploitation, la fin de ligne est signalée par une fin de ligne (LF, line feed), un retour charriot (CR, carriage return) ou les 2.

Attention Le nombre de lignes à ignorer inclus les lignes de commentaire et les lignes vides.

L'argument `nrows=`

À l'inverse, on peut ne s'intéresser qu'aux premières lignes d'un fichier. Pour cela, on va utiliser l'argument `nrows=`. Cet argument précise le nombre d'observations (les lignes de commentaire sont ignorées) qui doivent être lues par *R*.

Si `nrow` a une valeur négative, l'ensemble du fichier est lu. La valeur par défaut est de `-1`

```
Mesure3 <- read.table(file="FichiersEx/Mesure.txt", header=FALSE,
                      sep=":", dec=",", comment.char="!",
                      na.string="-99", col.names=nom.var,
                      row.names=4, nrows=3)
Mesure3
      Hauteur Epaisseur Largeur
pom2     12.3       5.2     7.1
pom3     11.3       5.2     7.0
poi1     10.2       5.5     7.0
```

Attention C'est le nombre d'observations et non le nombre de lignes qui doit être précisé. Dans notre exemple, la première ligne est un commentaire.

La fonction `read.table()` suite

Plusieurs autres arguments peuvent se révéler utiles pour certaines structures de fichiers. Pour illustrer l'utilisation de ces arguments, on va utiliser le fichier, `Nom.txt`.

Dans ce fichier, les valeurs sont séparées par le caractère « ; ». La première ligne contient le nom des variables. Il n'y a pas de valeurs manquantes mais plusieurs lignes vides et la ligne 7 est incomplète.

La lecture de ce fichier va se faire avec l'instruction suivante :

```
Id; Genre; Especе; Nombre rares; 100Communes
1203; G1;      Sp1; 1; Trouville
2389; G1 ; Sp1; 0; La Tranche sur Mer
5674; G3; Sp1    ; 0; Nice
4356; G2; Sp1; 0; L'Isle

     6756; G4; Sp1; 1
```

Le fichier `Nom.txt`

```
Nom <- read.table(file="FichiersEx/Nom.txt", header=TRUE, sep=";",
                  check.names=TRUE, stringsAsFactors=FALSE,
                  quote="", fill=TRUE, strip.white=TRUE,
                  blank.lines.skip=TRUE)
```

L'argument `check.names=TRUE`

Cet argument permet à *R* de vérifier que les noms des variables sont conformes à ce qu'il attend. En effet, sous *R*, ne sont autorisés pour les noms de variables que les lettres minuscules (a–z) ou majuscules (A–Z), les chiffres (0–9) et les signes « . » et « _ ».

La fonction `make.name()` permet de faire cette vérification.

De plus, le nom ne doit pas commencer par un chiffre et doit être différent de `TRUE, FALSE, NA, NULL`.

```
Nom <- read.table(file="FichiersEx/Nom.txt", header=TRUE, sep=";",
                  check.names=TRUE,


names(Nom)
[1] "Id"           "Genre"        "Especе"       "Nombre.rares" "X100Communes"
```

Dans cet exemple, *R* a remplacé les noms « `Nombre rares` » et « `100Communes` » par `Nombre.rares` et `X100Communes`.

L'argument `stringsAsFactors`

Plusieurs des valeurs contenues dans le fichier sont des chaînes de caractères. Par défaut, *R* les lit avec une classe spéciale, la classe `factor`.

Dans un certain nombre de cas, on peut souhaiter que ce ne soit pas le cas. En particulier lorsque l'on a de très nombreuses variables et observations. Il suffit alors de mettre l'argument à `FALSE`.

Voir aussi l'argument `as.is`

```
Nom <- read.table(file="FichiersEx/Nom.txt", header=TRUE, sep=";",
                  check.names=TRUE, stringsAsFactors=FALSE,
```

```
str(Nom, max.level=1)
'data.frame': 5 obs. of  5 variables:
 $ Id          : int  1203 2389 5674 4356 6756
 $ Genre       : chr  "G1" "G1" "G3" "G2" ...
 $ Espece      : chr  "Sp1" "Sp1" "Sp1" "Sp1" ...
 $ Nombre.rares: int  1 0 0 0 NA
 $ X100Communes: chr  "Trouville" "La Tranche sur Mer" "Nice" "L'Isle" ...
```

L'argument `quote=`

Un problème qui peut survenir avec les chaînes de caractères concerne la présence de l'apostrophe dans la valeur. Par exemple, dans notre fichier, on a, en ligne 5, la chaine de caractères « L'Isle ». La présence de l'apostrophe dans la valeur perturbe la lecture du fichier :

Les valeurs par défaut sont le guillemet droit simple ou double (`' "`)

```
Nom <- read.table(file="FichiersEx/Nom.txt", header=TRUE, sep=";",
                  check.names=TRUE, stringsAsFactors=FALSE)
dim(Nom)
[1] 0 5
```

On obtient un tableau ne comportant aucune ligne. Pour éviter ce problème, il suffit de spécifier

```
Nom <- read.table(file="FichiersEx/Nom.txt", header=TRUE, sep=";", nrow=4,
                  check.names=TRUE, stringsAsFactors=FALSE,
                  quote="")
Nom
    Id Genre    Espece Nombre.rares        X100Communes
1 1203    G1       Sp1            1           Trouville
2 2389   G1        Sp1            0  La Tranche sur Mer
3 5674    G3   Sp1                0                Nice
4 4356    G2       Sp1            0              L'Isle
```

L'argument `fill`

Dans ce fichier, il manque 2 valeurs pour l'observation n° 5 (ligne 6 du fichier).

```
Nom <- read.table(file="FichiersEx/Nom.txt", header=TRUE, sep=";", nrow=6,
                  check.names=TRUE, stringsAsFactors=FALSE,
                  quote="")
```

```
Error in scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings, :
la ligne 5 n'avait pas 5 éléments
```

L'argument `fill`, permet de compléter l'observation en ajoutant un blanc ou une valeur « NA ».

```
Nom <- read.table(file="FichiersEx/Nom.txt", header=TRUE, sep=";", nrow=6,
                  check.names=TRUE, stringsAsFactors=FALSE,
                  quote="", fill=TRUE)
Nom
    Id Genre    Especе Nombre.rares        X100Communes
1 1203    G1       Sp1            1           Trouville
2 2389   G1        Sp1            0  La Tranche sur Mer
3 5674    G3   Sp1                0                Nice
4 4356    G2       Sp1            0              L'Isle
5 6756    G4       Sp1           NA
```

L'argument `strip.white=FALSE`

Un autre souci avec ce fichier[2] concerne les valeurs des variables `Genre` et `Especе`. On peut noter la présence de blancs dans la chaîne de caractères.

2. Décidemment, il est mal fichu!!

```
Nom[, 2:3]
  Genre    Especе
1    G1       Sp1
2   G1        Sp1
3    G3   Sp1
4    G2       Sp1
5    G4       Sp1
```

On peut supprimer ces blancs lors de la lecture avec l'argument `strip.white` en le mettant à `TRUE`

```
Nom <- read.table(file="FichiersEx/Nom.txt", header=TRUE, sep=";", nrow=6,
                  check.names=TRUE, stringsAsFactors=FALSE,
                  quote="", fill=TRUE, strip.white=TRUE)
Nom[,2:3]
  Genre Especе
1    G1    Sp1
2    G1    Sp1
3    G3    Sp1
4    G2    Sp1
5    G4    Sp1
```

L'argument `blank.lines.skip=TRUE`

Enfin le dernier argument vu avec ce fichier, concerne la présence de lignes vides. On peut les supprimer avec l'argument `blank.lines.skip`. Par défaut, cet argument est à `TRUE` et les lignes vides sont supprimées.

```
Nom <- read.table(file="FichiersEx/Nom.txt", header=TRUE, sep=";",
                  check.names=TRUE, stringsAsFactors=FALSE,
                  quote="", fill=TRUE, strip.white=TRUE,
                  blank.lines.skip=TRUE)
dim(Nom)
[1] 5 5
```

mais si on le met à `FALSE`

```
Nom <- read.table(file="FichiersEx/Nom.txt", header=TRUE, sep=";",
                  check.names=TRUE, stringsAsFactors=FALSE,
                  quote="", fill=TRUE, strip.white=TRUE,
                  blank.lines.skip=FALSE)
dim(Nom)
[1] 7 5
```

on conserve les lignes vides, ce qui se traduit dans le tableau par des observations ne présentant que des valeurs manquantes.

```
Nom
    Id Genre Espece Nombre.rares       X100Communes
1 1203    G1    Sp1            1          Trouville
2 2389    G1    Sp1            0 La Tranche sur Mer
3 5674    G3    Sp1            0               Nice
4 4356    G2    Sp1            0             L'Isle
5 6756    G4    Sp1           NA
6   NA                        NA
7   NA                        NA
```

La fonction `read.table()` fin

Pour finir quelques arguments qui peuvent être très utiles mais pour des situations très particulières.

L'argument `fileEncoding`

Indique le jeu de caractères utilisé dans un fichier. Il permet de recoder les chaînes de caractères. Le fichier `TexteWin.txt` a été créé sous Windows. Il est lu sur une machine avec OSX qui utilise le jeu de caractères

```
localeToCharset()
[1] "UTF-8"
```

La lecture du fichier avec l'instruction suivante

```
# Lecture du fichier
tmp <- read.table(file="FichiersEx/TexteWin.txt", header=TRUE, stringsAsFactors=FALSE)
```

conduit à des erreurs sur le codage des caractères :

```
tmp
          Article Nombre
1 T\xe9l\xe9phone     10
2       Gla\xe7on     20
3        For\xeat      5
Encoding(tmp$Article)
[1] "unknown" "unknown" "unknown"
```

En précisant le jeu de caractères du fichier

```
(tmp <- read.table(file="FichiersEx/TexteWin.txt", header=TRUE, stringsAsFactors=FALSE,
                   fileEncoding="latin1") )
    Article Nombre
1 Téléphone     10
2    Glaçon     20
3     Forêt      5
Encoding(tmp$Article)
[1] "unknown" "unknown" "unknown"
```

L'affichage est correct mais le codage de la variable `Article` est inconnu.

L'argument `encoding`

Marque les chaînes de caractères avec le jeu de caractères indiqué.

```
(tmp <- read.table(file="FichiersEx/TexteWin.txt", header=TRUE, stringsAsFactors=FALSE,
                   encoding="latin1") )
    Article Nombre
1 Téléphone     10
2    Glaçon     20
3     Forêt      5
Encoding(tmp$Article)
[1] "latin1" "latin1" "latin1"
```

Les chaînes de caractères ne sont pas recodées mais elles sont marquées et donc reconnues par *R*

L'argument `as.is`

Par défaut, la fonction `read.table()` essaie de déterminer le type des différentes variables. Elle essaie les types logique, entier, numérique et complexe. Si aucun de ces types ne convient, c'est le type `factor` qui est utilisé.

C'est donc le cas pour les variable de type chaînes de caractères. Dans certains cas et/ou pour certaines variables, ce comportement n'est pas souhaitable. On peut le modifier en spécifiant dans un vecteur logique (succession de valeurs `TRUE`, `FALSE`), les variables ne devant pas faire l'objet de cette conversion. La longueur du vecteur doit être égale au nombre de variables (y compris celles non incluses dans le tableau et celles permettant de nommer les lignes).

On peut aussi utiliser un vecteur listant le numéro ou le nom des variables qui ne doivent pas être converties.

Fonctions dérivées

Ces fonctions appellent toutes `read.table()` mais la valeur par défaut de certains des arguments est différente. Les « ... » dénotent les arguments supplémentaires qui sont transmis tels quels à la fonction `read.table()`.

La fonction `read.csv()`

```
read.csv(file, header = TRUE, sep = ",", quote = "\"", dec = ".", fill = TRUE, comment.char = "",
    ...)
```

La fonction `read.csv2()`

```
read.csv2(file, header = TRUE, sep = ";", quote = "\"", dec = ",", fill = TRUE, comment.char = "",
    ...)
```

La fonction `read.delim()`

```
read.delim(file, header = TRUE, sep = "\t", quote = "\"", dec = ".", fill = TRUE,
    comment.char = "", ...)
```

La fonction `read.delim2()`

```
read.delim2(file, header = TRUE, sep = "\t", quote = "\"", dec = ",", fill = TRUE,
    comment.char = "", ...)
```