

R01 Import de données

A partir du package de base, R peut lire les données stockées dans des fichiers texte, à partir notamment des fonctions `read.table`, `read.csv`, `read.delim`, `read.csv2`, `read.delim2`, et `scan`.

R lit également les fichiers d'autres formats, parmi lesquels Excel et SAS que nous présentons dans ce document. La lecture de ces formats nécessite l'installation préalable de certains packages qui ne font pas partie du package de base : `gdata` et `gremisc` pour la lecture de classeurs Excel (fonction `read.xls`), `foreign` pour la lecture de tables SAS (fonction `read.ssd`).

1. Import de fichiers texte

On se placera dans le cas où les données se présentent sous la forme tabulaire avec des lignes correspondants aux observations et des colonnes correspondant aux variables.

Exemple 1 :

Pays	gini	farm	rent	gnpr	labo	inst	ecks	deat	demo
Argentine	86.3	98.2	32.9	374	25	13.6	57	217	2
Australie	92.9	99.6	.	1215	14	11.3	0	0	1
Autriche	74	97.4	10.7	532	32	12.8	4	0	2
Belgique	58.7	85.8	62.3	1015	10	15.5	8	1	1
Bolivie	93.8	97.7	20	66	72	15.3	53	663	3
Brésil	83.7	98.5	9.1	262	61	15.5	49	1	3
Canada	49.7	82.9	7.2	1667	12	11.3	22	0	1
Chili	93.8	99.7	13.4	180	30	14.2	21	2	2

1.1. La fonction `read.table`

La manière la plus simple d'importer un fichier texte tel que celui de l'exemple 1 est d'utiliser la fonction `read.table`.

La fonction `read.table` comporte de nombreuses options dont voici les valeurs par défaut (c'est-à-dire celles utilisées par le logiciel si elles ne sont pas spécifiées par l'utilisateur).

```
read.table(  
  file,  
  header=FALSE,  
  sep="",  
  quote="\\"",  
  dec=".",  
  row.names,  
  col.names,  
  as.is=FALSE,  
  na.strings="NA",  
  colClasses=NA,  
  nrow=-1,  
  skip=0,  
  check.names=TRUE,  
  fill=!blank.lines.skip,  
  strip.white=FALSE,  
  blank.lines.skip=TRUE,  
  comment.char="#")
```

`file`

L'option `file` donne le nom du fichier (entre "", ou entre ' ' ou une variable de mode caractère), éventuellement avec son chemin d'accès (le symbole \ est interdit et doit être remplacé par /, même sous Windows), ou un accès distant à un fichier de type URL (<http://...>).

header

Si `header=TRUE`, le fichier contient les noms des variables sur la première ligne. Sinon, `header=FALSE` et on n'a pas besoin de le préciser puisqu'il s'agit de la valeur par défaut.

Ainsi, si on importe le fichier de l'exemple 1, on utilise l'option `header=true`.

sep

Dans un fichier texte, il peut y avoir différents séparateurs. On précise le type de séparateur avec l'option `sep=`. On a le choix entre :

- le séparateur par défaut `sep=""` qui utilise n'importe quel espace blanc (espaces, tabulations, ou retour chariot),
- `sep=" "`
- `sep="\t"` pour les tabulations.

Dans le cas où l'on dispose d'un fichier séparé par des tabulations qui contient des valeurs manquantes, il est indispensable de préciser l'option `sep="\t"`.

Exemple 2 : importation du fichier auto2004_original

Les premières lignes du fichier se présentent sous la forme suivante :

Modèle	Cylindree	Puissance	Vitesse	Poids	Largeur	Longueur	
Citroen C2 1.1 Base		1124	61	158	932	1659	3666
Smart Fortwo Coupe		698	52	135	730	1515	2500
Mini 1.6 170	1598	170	218	1215	1690	3625	
Nissan Micra 1.2 65		1240	65	154	965	1660	3715
Renault Clio 3.0 V6		2946	255	245	1400	1810	3812
Audi A3 1.9 TDI 1896		105	187	1295	1765	4203	

La 1^{ère} ligne correspond au nom des variables. On utilise par conséquent l'option `header = TRUE`.

Le séparateur est une tabulation. On utilise l'option `sep='\t'`.

```
A = read.table(
  'd:/R - données import/auto2004_original.txt',
  header = TRUE,
  sep = '\t')
```

On peut ensuite visualiser les données avec la commande `fix(A)`.

encoding

L'option `encoding` est utilisée pour lire des fichiers qui contiennent des caractères non ASCII.

Par exemple, si le fichier contient des caractères latin1 tels que ci-dessous :

Subset: Latin-1															
Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð
Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	à
á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð
ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	Ā

La lecture du fichier se fera selon la syntaxe suivante :

```
read.table(file("file.dat", encoding="latin1"))
```

skip

Indique le nombre de lignes à sauter avant de commencer la lecture des données.

Exemple 3 : importation du fichier auto2004_original_ligne3

Les 3 premières lignes du fichier décrivent les données. On importe les données qu'à partir de la 4^{ème} ligne.

Fichier auto

24 observations

1 variable caractère, 5 variables numériques

Modele	Cylindree	Puissance	Vitesse	Poids	Largeur	Longueur	
Citroen C2 1.1 Base		1124	61	158	932	1659	3666
Smart Fortwo Coupe		698	52	135	730	1515	2500
Mini 1.6 170	1598	170	218	1215	1690	3625	
Nissan Micra 1.2 65		1240	65	154	965	1660	3715
Renault Clio 3.0 v6		2946	255	245	1400	1810	3812
Audi A3 1.9 TDI 1896		105	187	1295	1765	4203	

```
B = read.table(  
  'd:/R - données import/auto2004_original_ligne3.txt',  
  header = TRUE,  
  sep = '\t',  
  skip=3)
```

quote

Par défaut, les chaînes de caractères sont inscrites entre "" ou "" et dans chacun des cas, tous les caractères entre deux quotes identiques font partie de la chaîne de caractères.

Si aucun séparateur n'est spécifié, si des quotes sont insérées dans une chaîne de caractères, elles ne seront pas considérées comme la fin de la chaîne de caractères si elles sont précédées par \.

Si un séparateur est précisé, les quotes insérées à l'intérieur d'une chaîne de caractères en seront pas prises en compte. Par exemple :

'Cette chaîne de caractère n'est pas coupée en deux', en voici une autre'
peut être lue grâce à l'instruction :

```
read.table("testfile", sep = ",")
```

dec

L'option dec indique le caractère utilisé pour les décimales.

fill

Lignes de longueur variable.

Lorsqu'un fichier comportent des lignes qui n'ont pas le même nombre de variables, on utilisera l'option fill=TRUE.

strip.white

Espaces blancs dans les chaînes de caractères

Si un séparateur est spécifié, les espaces blancs avant et après les chaînes de caractères sont considérées comme faisant partie de la chaîne de caractères. Pour éviter cela, on utilise l'argument strip.white=TRUE.

blank.lines.skip=FALSE

Par défaut, read.table ignore les lignes blanches. Pour que les lignes blanches soient prises en compte, il faut utiliser l'option blank.lines.skip=FALSE en conjonction avec l'option fill=TRUE.

`as.is`

Format des variables.

En l'absence de spécification, l'instruction `read.table` détermine automatiquement le type de la variable. Il essaie successivement les types logique, entier, numérique, et complexe. Si aucun de ses types ne convient, il convertit la variable en facteur.

L'argument `as.is` contrôle la conversion des variables caractères en facteur (si `FALSE`) ou les conserve en caractères (`TRUE`). `as.is` peut être un vecteur logique ou un vecteur numérique précisant les variables conservées en caractère.

`colClasses`

Vecteur de caractères donnant les classes à attribuer aux colonnes.

On notera que les options `colClasses` et `as.is` sont spécifiées par colonne, et non par variable. Elles incluent par conséquent la colonne de l'identifiant (s'il existe).

`col.names`

Un vecteur contenant les noms des variables (par défaut : `V1, V2, V3, ...`)

Exemple 4 : importation du fichier `auto2004_sans_nom`

Le nom des variables ne figure pas sur la 1^{ère} ligne du fichier. On va donc préciser le nom des variables par l'instruction `col.names`.

Citroen C2 1.1 Base	1124	61	158	932	1659	3666
Smart Fortwo Coupe	698	52	135	730	1515	2500
Mini 1.6 170	1598	170	218	1215	1690	3625
Nissan Micra 1.2 65	1240	65	154	965	1660	3715
Renault Clio 3.0 v6	2946	255	245	1400	1810	3812
Audi A3 1.9 TDI 1896	105	187	1295	1765	4203	

```
C = read.table(
  'd:/R - données import/auto2004_sans_nom.txt',
  sep='\t',
  col.names=c('Modele', 'Cylindree', 'Puissance', 'Vitesse', 'Poids',
  'Largeur', 'Longueur'))
```

Si on ne précise pas `col.names`, les noms des variables sont `V1, V2...V7`.

`row.names`

Un vecteur contenant les noms des lignes qui peut être un vecteur de mode caractère, ou le numéro (ou le nom) d'une variable du fichier (par défaut : `1, 2, 3, ...`)

Exemple 5 : importation du fichier `auto2004_sans_nomcol_nomrow`

Le fichier comprend les 3 premières lignes du fichier d'origine sur les voitures, sans le nom des variables, ni le nom des voitures.

1124	61	158	932	1659	3666
698	52	135	730	1515	2500
1598	170	218	1215	1690	3625

```
D = read.table(
  'd:/R - données import/auto2004_sans_nomcol_nomrow.txt',
  sep='\t',
  col.names=c('Cylindree', 'Puissance', 'Vitesse', 'Poids', 'Largeur',
  'Longueur'),
  row.names=c('Citroen C2 1.1 Base', 'Smart Fortwo Coupe', 'Mini
  1.6 170'))
```

L'inconvénient de cette procédure est qu'on n'a pas le nom « Modele » pour la colonne `row.names`. Il faudra renommer cette variable.

`check.names`

Si TRUE, vérifie que les noms des variables sont valides pour R.

`nrows`

Le nombre maximum de lignes à lire (les valeurs négatives sont ignorées).

Exemple 6 : importation des 10 premières lignes du fichier `auto2004_original_ligne3`

```
E = read.table(  
  'd:/R - données import/auto2004_original_ligne3.txt',  
  header = TRUE,  
  sep = '\t',  
  skip=3,  
  nrows=10)
```

`comment.char`

Par défaut, `read.table` utilise # pour définir les commentaires dans un fichier de données. Dès qu'il rencontre ce caractère, le reste de la ligne est ignorée. Les lignes qui ne contiennent que des espaces et des commentaires sont considérées comme des lignes vides.

Si l'on sait qu'il n'y aura pas de commentaires dans le fichier, il est préférable de préciser `comment.char=""`.

`na.strings`

Valeurs manquantes.

Par défaut, R considère que dans le fichier à importer, les valeurs manquantes sont représentées par la chaîne de caractères `na`. Mais on peut préciser les valeurs manquantes par l'argument `na.chaine_de_caractères`, `chaine_de_caractère` étant un vecteur comprenant un ou plusieurs caractères correspondant aux valeurs manquantes. La valeur des données manquantes sera convertie en `na`.

Exemple 7 : importation du fichier `auto2004_don_manquante`

Les valeurs manquantes sont repérées par des blancs.

Modele	Cylindree	Puissance	Vitesse	Poids	Largeur	Longueur	
Citroen C2 1.1 Base		1124	158	932	1659		
Smart Fortwo Coupe		698	52	730	1515	2500	
Mini 1.6 170	1598	170	218	1215	1690	3625	
Nissan Micra 1.2 65		1240	65	154	965	1660	3715
Renault Clio 3.0 v6		2946	255	245		1810	3812
Audi A3 1.9 TDI 1896		105	187	1295	1765	4203	

```
F = read.table(  
  'd:/R - données import/auto2004_don_manquante.txt',  
  sep='\t',  
  header=TRUE)
```

Dans cet exemple, l'option `na.strings` est inutile car R détecte automatiquement les valeurs manquantes si deux tabulations se suivent.

Exemple 8 : importation du fichier auto2004_don_manquan_99999

Les valeurs manquantes sont maintenant codées 99999.

Modele	Cylindree	Puissance	Vitesse	Poids	Largeur	Longueur	
Citroen C2 1.1 Base		1124	99999	158	932	1659	99999
Smart Fortwo Coupe		698	52	99999	730	1515	2500
Mini 1.6 170	1598	170	218	1215	1690	3625	
Nissan Micra 1.2 65		1240	65	154	965	1660	3715
Renault Clio 3.0 V6		2946	255	245	99999	1810	3812
Audi A3 1.9 TDI 1896		105	187	1295	1765	4203	

```
G = read.table(  
  'd:/R_données_import/auto2004_don_manquan_99999.txt',  
  sep='\t',  
  header=TRUE,  
  na.strings=99999)
```

1.2. Quatre variantes de la fonction `read.table` : `read.csv`, `read.delim`, `read.csv2` et `read.delim2`

Dans plusieurs documents sur R, il est indiqué que ces fonctions s'utilisent dans les cas suivants :

- `read.csv` et `read.delim` : dans le cas où l'on importe des fichiers de type csv ou des fichiers à tabulation avec des `.` pour les décimales ;
- `read.csv2` et `read.delim2` : dans le cas où l'on importe des fichiers de type csv ou des fichiers à tabulation avec des `,` pour les décimales.

Les fonctions s'utilisent avec les mêmes options que `read.table` :

```
read.csv(file, sep = ",", dec=".",...)  
read.csv2(file, sep = ";", dec=".",...)  
read.delim(file, sep = "\t", dec=".",...)  
read.delim2(file, sep = "\t", dec=".",...)
```

Sur les deux exemples suivants, respectivement un fichier csv avec des `.` pour les décimales, et un fichier csv avec des `,` pour les décimales, on a pu utiliser indifféremment les 5 fonctions `read.table`, `read.csv`, `read.delim`, `read.csv2` et `read.delim2`. Il semble donc que dans la plupart des cas, on puisse utiliser ces 5 fonctions indifféremment.

Exemple 9 : importation du fichier pays.csv

```
Argentine,86.3,98.2,32.9,374,25,13.6,57,217,2  
Australie,92.9,99.6,,1215,14,11.3,0,0,1  
Autriche,74.0,97.4,10.7,532,32,12.8,4,0,2  
Belgique,58.7,85.8,62.3,1015,10,15.5,8,1,1  
Bolivie,93.8,97.7,20.0,66,72,15.3,53,663,3
```

On peut utiliser indifféremment les 5 instructions suivantes :

```
H=read.table('d:/R_données_import/pays.csv',sep = ",", dec=".",  
col.names=c('Pays','gini','farm','rent','gnpr','labo','inst','ecks',  
'deat','demo'))
```

```
H=read.csv('d:/R_données_import/pays.csv',sep = ",", dec=".",  
col.names=c('Pays','gini','farm','rent','gnpr','labo','inst','ecks',  
'deat','demo'))
```

```
H=read.delim('d:/R_données_import/pays.csv',sep = "\t", dec=".",  
col.names=c('Pays','gini','farm','rent','gnpr','labo','inst','ecks',  
'deat','demo'))
```

```
H=read.csv2('d:/R_données_import/pays.csv', sep = ",", dec=".",  
col.names=c('Pays','gini','farm','rent','gnpr','labo','inst','ecks',  
'deat','demo'))
```

```
H=read.delim2('d:/R_données_import/pays.csv', sep = ",", dec=".",  
col.names=c('Pays','gini','farm','rent','gnpr','labo','inst','ecks',  
'deat','demo'))
```

Exemple 10 : importation du fichier pays1.csv

```
Argentine;86,3;98,2;32,9;374;25;13,6;57;217;2  
Australie;92,9;99,6;;1215;14;11,3;0;0;1  
Autriche;74,0;97,4;10,7;532;32;12,8;4;0;2  
Belgique;58,7;85,8;62,3;1015;10;15,5;8;1;1  
Bolivie;93,8;97,7;20,0;66;72;15,3;53;663;3  
Brésil;83,7;98,5;9,1;262;61;15,5;49;1;3
```

Dans ce cas également, on peut utiliser les 5 fonctions, avec les options `sep` et `dec` suivantes :

```
I=read.table('d:/R_données_import/pays1.csv', sep = ";", dec=".",  
col.names=c('Pays','gini','farm','rent','gnpr','labo','inst','ecks',  
'deat','demo'))
```

1.3. Fichiers texte à colonnes fixes : `read.fwd`

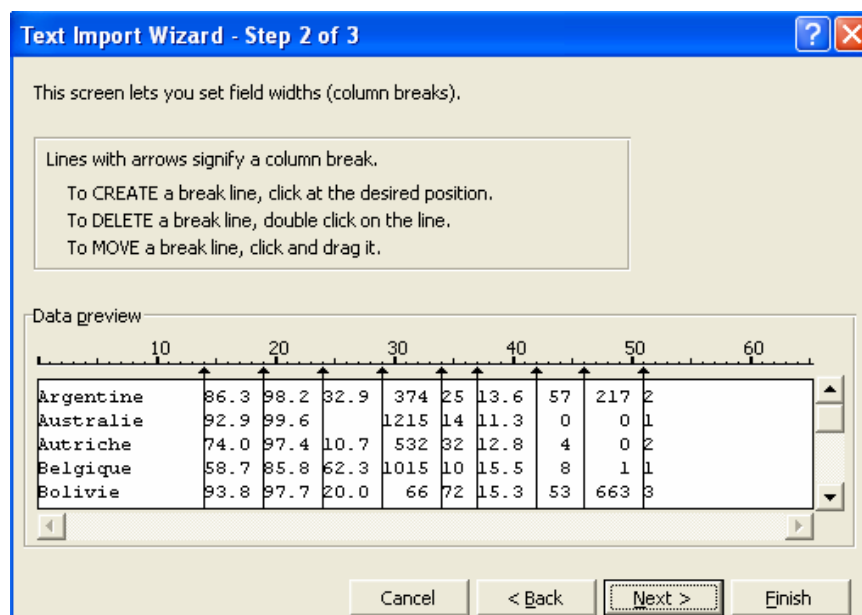
La fonction `read.fwd` s'utilise pour importer des fichiers à colonnes fixes, comme par exemple le fichier `pays2.txt` suivant :

Argentine	86.3	98.2	32.9	374	25	13.6	57	217	2
Australie	92.9	99.6		1215	14	11.3	0	0	1
Autriche	74.0	97.4	10.7	532	32	12.8	4	0	2
Belgique	58.7	85.8	62.3	1015	10	15.5	8	1	1
Bolivie	93.8	97.7	20.0	66	72	15.3	53	663	3
Brésil	83.7	98.5	9.1	262	61	15.5	49	1	3

Les arguments sont les mêmes que pour la fonction `read.table`, on a seulement un argument supplémentaire `widths` qui est un vecteur indiquant la largeur en nombre de colonnes de chaque variable du fichier importé.

Exemple 11 : importation du fichier `pays2.txt`

Si on importait le fichier `pays2.txt`, l'assistant d'importation ouvrirait la fenêtre suivante, indiquant la délimitation des champs en fonction des colonnes :



Avec l'option `width`, on va indiquer la largeur de chaque champ de la même façon :

- 14 premières colonnes pour 1ère variable,
- 5 colonnes pour chacune des 4 variables suivantes,
- 3 colonnes pour la 6^{ème} variable,
- 5 colonnes pour la 7^{ème} variable,
- 4 colonnes pour la 8^{ème} colonne,
- 5 colonnes pour la 9^{ème} colonne
- et 1 colonne pour la dernière variable.

```
J=read.fwf('d:/R_données_import/pays2.txt',  
width=c(14,5,5,5,3,5,4,5,1),  
col.names=c('Pays','gini','farm','rent','gnpr','labo','inst','ecks',  
'deat','demo'))
```

1.4. Fonction `scan`

La fonction `read.table` n'est pas la plus efficace lorsqu'on doit importer des fichiers volumineux. On préférera dans ce cas utiliser la fonction `scan`.

La fonction `scan` fait appel de nombreuses options déjà vues avec `read.table`. C'est normal puisque `read.table` utilise `scan`.

```
scan(  
  file = "",  
  what = double(0),  
  nmax = -1,  
  n = -1,  
  sep = "",  
  quote = if (sep=="\n") "" else "'\"'",  
  dec = ".",  
  skip = 0,  
  nlines = 0,  
  na.strings = "NA",  
  flush = FALSE,  
  fill = FALSE,  
  strip.white = FALSE,  
  quiet = FALSE,  
  blank.lines.skip = TRUE,  
  multi.line = TRUE,  
  comment.char = "")
```

`what`

C'est essentiellement par cet argument que les fonctions `read.table` et `scan` diffèrent.

Il indique le(s) mode(s) des données lues (numérique par défaut).

Exemple 12 : importation du fichier `iris.txt`

Le jeu de données (`iris.txt`) est composé de 6 variables numériques

```
K = scan("d:/R_données_import/iris.txt",  
what = list(0, 0, 0, 0, 0, 0))
```

Exemple 13 : importation du fichier `iris1.txt`

Le jeu de données (`iris1.txt`) est composé de 5 variables numériques et d'une variable nominale.

```
L = scan("d:/R_données_import/iris1.txt",  
what = list(0, 0, 0, 0, 0, " "))
```


`nmax`

Le nombre de données à lire, ou, si `what` est une liste, le nombre de lignes lues (par défaut, `scan` lit jusqu'à la fin du fichier)

`n`

Le nombre de données à lire (par défaut, pas de limite)

`sep`

Le séparateur de champ dans le fichier

`quote`

Les caractères utilisés pour citer les variables de mode character

`dec`

Le caractère utilisé pour les décimales

`skip`

Le nombre de lignes à sauter avant de commencer la lecture des données.

`nlines`

Le nombre de lignes à lire

`na.string`

Indique la valeur des données manquantes (sera converti en NA)

`flush`

Si `TRUE`, `scan` va à la ligne suivante une fois que le nombre de colonnes est atteint (permet d'ajouter des commentaires dans le fichier de données).

Exemple 14 : importation du fichier `iris1.txt`

Par exemple, on ne veut importer que les 5 premières variables numériques du fichier `iris1.txt`.

```
L = scan("d:/R_données_import/iris1.txt",  
what = list(0, 0, 0, 0, 0),  
flush=TRUE)
```

`fill`

Si `TRUE` et que les lignes n'ont pas tous le même nombre de variables, des "blancs" sont ajoutés.

`strip.white`

(Conditionnel à `sep`) si `TRUE`, efface les espaces (= blancs) avant et après les variables de mode character.

`quiet`

Si `FALSE`, `scan` affiche une ligne indiquant les champs lus

`blank.lines.skip`

Si `TRUE`, ignore les lignes "blanches"

`multi.line`

Si `what` est une liste, précisez si les variables du même individu sont sur une seule ligne dans le fichier (`FALSE`)

`comment.char`

Un caractère qui définit des commentaires dans le fichier de données, les lignes commençant par ce caractère sont ignorées

2. Import de fichiers Excel : la fonction `read.xls`

La fonction `read.xls` ne peut être utilisée qu'après avoir installé les packages `gdata` et `gregmisc` et les avoir activés par les instructions `library(gdata)` et `library(gregmisc)`. Il est nécessaire également de disposer de l'exécutable `perl.exe`, qui doit se trouver dans le répertoire `C:\perl\bin`.

On présente sur l'exemple 15 les commandes à exécuter pour importer un fichier Excel.

Exemple 15 : importation du fichier `birth_rates.xls`

- Le fichier `birth_rates.xls` a été placé dans le répertoire `c:/Program Files / R/ R-2.2.1/library/gdata/xls`.

On exécute les commandes suivantes :

```
xlsfile <- file.path(.path.package('gdata'),'xls','Birth_rates.xls')
Birth_rates <-read.xls(xlsfile)
```

- On a créé un répertoire `fichiers_import` dans `c:/Program Files / R/ R-2.2.1/library/gdata` dans lequel on a placé le fichier `birth_rates.xls`.

```
xlsfile<file.path(.path.package('gdata'),'fichiers_import','Birth_rates.xls')
Birth_rates <-read.xls(xlsfile)
```

Si on ne dispose pas de l'exécutable `perl.exe`, il faut convertir préalablement le fichier excel en fichier texte et l'importer par les fonctions présentées au paragraphe 1.

3. Import de fichiers SAS : la fonction `read.ssd`

L'import de fichiers SAS nécessite l'installation préalable du package `foreign`¹ et son activation par l'instruction `library(foreign)`.

Exemple 16 : importation du fichier `paysniv3.sas7bdat`

- Le fichier `paysniv3.sas7bdat` a été placé dans le répertoire `C:/Program Files/SAS/SAS 9.1`

On exécute les instructions suivantes :

```
sashome <- "C:/Program Files/SAS/SAS 9.1"
G=read.ssd(file.path(sashome),"paysniv3",sascmd = file.path(sashome,
"sas.exe"))
```

Il ne faut pas mettre l'extension du fichier (`sas7bdat`).

¹ Le package `foreign` permet également d'importer des fichiers `EpilInfo`, `Minitab`, `S-PLUS`, `SPSS`, `Stata` et `Systat`.

- Le fichier paysniv3.sas7bdat a été placé dans le répertoire C:/Program Files/SAS/SAS 9.1/fic_R (on a créé ce répertoire).

```
H=read.ssd(file.path(sashome,"fic_R"),"paysniv3", sascmd = file.path(sashome, "sas.exe"))
```

- Le fichier paysniv3.sas7bdat a été placé dans le répertoire C:/Program Files/SAS/SAS 9.1/fichier_import/fichiers_vers_R (on a créé ce répertoire).

```
H=read.ssd(file.path(sashome,"fichier_import","fichiers_vers_R"), "paysniv3", sascmd = file.path(sashome, "sas.exe"))
```

Références bibliographiques

- documentation disponible sur le site de R et notamment :
 - R-data.pdf, disponible avec le package de base,
 - foreign.pdf, téléchargeable sur la page dédiée au package foreign,
 - gdata.pdf, téléchargeable sur la page relative au package gdata,
 - gremisc, téléchargeable sur la page relative au package gremisc.
- polycopiés du cours d'Arthur Tenenhaus dispensés au Cnam sur le logiciel R et disponible à l'adresse :
http://www.cnam.fr/math/LEnseignement.php3?id_article=154
- documentation du Centre for Mathematical Sciences de l'université de Lund, et notamment :
<http://www.maths.lth.se/help/R/.R/library/base/html/read.table.html>
<http://www.maths.lth.se/help/R/.R/library/utils/html/read.fwf.html>
<http://www.maths.lth.se/help/R/.R/library/base/html/scan.html>
<http://www.maths.lth.se/help/R/.R/library/gdata/html/read.xls.html>
<http://www.maths.lth.se/help/R/.R/library/foreign/html/read.ssd.html>