

Importer des données dans R

Fousseynou Bah

17-Dec-2018

- 1 Introduction
- 2 Fichiers plats: cas du format *CSV*
- 3 Excel: *xls*, *xlsx*
- 4 Formats issues d'autres logiciels statistiques: Stata et SPSS
- 5 Base de données relationnelles
- 6 Depuis Internet

Introduction

Objectif de ce cours

Dans le flux de travail (*workflow*) du *data scientist*, l'importation constitue très généralement le point de départ. Les données ne sont toujours disponibles sous le format qui se prête à l'analyse souhaitée. Elles peuvent exister dans un classeur Excel sous format *xls*, *xlsx* ou *csv*. Elles peuvent aussi se trouver dans une base de données relationnelles, où diverses tables sont connectées entre elles. Elles peuvent même être disponibles sur Internet (page Wikipédia, Twitter, Facebook, etc.) Dans tous les cas, il revient au *data scientist* de connaître la technique appropriée pour les importer dans son environnement de travail, les organiser et les analyser selon l'objectif qu'il s'assigne.

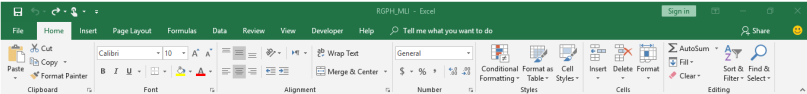
Dans ce cours, nous allons voir quelques techniques d'importation de données dans R.

Que nous faut-il?

- R (évidemment) et RStudio (de préférence) installés sur le poste de travail
- les fichiers fournis dans le cadre du module
- une connexion Internet pour illustrer les exemples d'importation depuis la toile

Aperçu

Nous allons illustrer ce chapitre avec une compilation de données tirées des Recensements Généraux de la Population et de l'Habitat au Mali, menés respectivement en 1976, 1987, 1998 et 2009. En voici un aperçu depuis Excel:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	annee	id	groupe	Homme-Urbain	Homme-Rural	Femme-Urbain	Femme-Rural	Homme	Femme	Urbain	Rural	Total	source	office		
1	1976	1	0-4	100 416	486 599	99 453	489 941	587 015	589 394	199 889	976 540	1 176 409	RGPH	DNSI		
2	1976	2	5-9	80 816	411 456	82 665	400 186	492 272	482 851	163 481	811 642	975 123	RGPH	DNSI		
3	1976	3	10-14	60 647	282 160	65 075	256 884	342 807	321 959	125 722	539 044	664 766	RGPH	DNSI		
4	1976	4	15-19	58 662	249 945	60 765	272 743	308 607	333 508	119 427	522 688	642 115	RGPH	DNSI		
5	1976	5	20-24	46 239	172 152	47 487	218 355	218 391	265 842	93 726	390 507	484 233	RGPH	DNSI		
6	1976	6	25-29	36 390	183 705	43 143	223 875	200 095	267 018	79 533	387 580	467 113	RGPH	DNSI		
7	1976	7	30-34	30 895	154 834	33 504	190 446	185 729	223 950	64 399	345 280	409 679	RGPH	DNSI		
8	1976	8	35-39	27 926	133 457	28 505	137 444	161 383	165 949	56 431	270 901	327 332	RGPH	DNSI		
9	1976	9	40-44	22 479	116 947	22 007	125 822	139 426	147 829	44 486	242 769	287 255	RGPH	DNSI		
10	1976	10	45-49	17 864	93 466	15 445	83 008	111 330	98 453	33 309	176 474	209 783	RGPH	DNSI		
11	1976	11	50-54	14 692	89 927	13 238	90 369	104 619	103 607	27 930	180 296	208 226	RGPH	DNSI		
12	1976	12	55-59	10 659	66 919	8 928	53 989	77 578	62 917	19 587	120 908	140 495	RGPH	DNSI		
13	1976	13	60-64	8 630	67 990	9 537	72 129	76 620	81 466	17 967	140 119	158 086	RGPH	DNSI		
14	1976	14	65-69	5 043	35 236	5 032	31 800	40 279	36 832	10 075	67 036	77 111	RGPH	DNSI		
15	1976	15	70-74	3 460	28 430	4 610	33 137	31 890	37 747	8 070	61 567	69 637	RGPH	DNSI		
16	1976	16	75-79	2 084	15 065	2 630	14 080	17 149	16 730	4 734	29 145	33 879	RGPH	DNSI		
17	1976	17	80+	2 188	25 958	3 677	29 082	28 146	32 759	5 865	55 040	60 905	RGPH	DNSI		
18	1976	18	NO	125	272	93	281	397	374	218	533	771	RGPH	DNSI		
19	1976	19	Total	529 235	2 594 518	545 614	2 723 571	3 125 733	3 269 185	1 074 829	5 318 089	6 392 918	RGPH	DNSI		
20	1987	1	0-4	149 762	570 042	147 536	565 971	719 804	713 507	297 298	1 136 013	1 433 311	RGPH	DNSI		
21	1987	2	5-9	128 986	504 220	126 068	485 494	633 206	611 562	255 054	989 714	1 244 768	RGPH	DNSI		
22	1987	3	10-14	101 282	350 884	101 294	313 008	452 166	414 302	202 576	663 892	866 468	RGPH	DNSI		
23	1987	4	15-19	88 302	259 898	98 163	281 359	348 200	379 522	186 465	541 257	727 722	RGPH	DNSI		
24	1987	5	20-24	72 800	187 315	77 355	238 998	260 215	315 753	150 245	425 758	575 968	RGPH	DNSI		
25	1987	6	25-29	61 829	169 337	70 244	236 307	231 166	306 551	132 073	405 644	537 717	RGPH	DNSI		

Avant de commencer

Notre fichier se nomme: “*RGPH_MLI*”. Nous allons l’ouvrir à partir de différents formats, qui seront traduits par diverses extensions (.csv, .txt, .xls, .xlsx, etc).

Avant d’ouvrir ces variantes, assurons-nous d’abord que le fichier est bien présent dans notre espace de travail (*working directory*):

```
dir(pattern = "data/RGPH_MLI.csv")
```

```
## character(0)
```

Le résultat indique que le fichier est présent dans l’espace de travail. Maintenant, ouvrons-le!

Fichiers plats: cas du format CSV

Aperçu

Comme son nom l'indique, le format *CSV* (*Comma Separated Values*) est un format où les valeurs de données rectangulaires sont séparées par une virgule (","),. Cette règle de base est commode pour les anglophones pour lesquels, les décimales viennent après un point (".") et non après une virgule. Par contre, pour les francophones, le format de sauvegarde de données réquiert une règle différente. Pour ne pas confondre le séparateur de valeurs et la virgule de la décimale, l'on utilise le point-virgule (";"). Nous verrons les fonctions qui conviennent pour chacun de ces deux formats.

Importation avec *R-base*: read.csv

La version d'installation de R, couramment appelée R-base (ou base-R), vient avec un ensemble de fonctions qui sont disponibles par défaut. Parmi celles-ci: "read.csv". L'importation d'un fichier *csv* avec *R-base* se fait de la façon suivante:

```
RGPH_MLI <- read.csv(
  file = "data/RGPH_MLI.csv", # spécifier le nom du fichier
  header = TRUE, # déclarer la première ligne comme nom de colonne/variable
  sep = ",", # déclarer la virgule comme séparateur
  stringsAsFactors = TRUE # voir cours sur les types de données dans R
)
```

Les éléments à l'intérieur de la fonction (*file*, *header*, *sep*, *stringsAsFactors*, *etc.*) sont les arguments de la fonction. Celles-ci ont des valeurs défaut, qui peuvent toutefois être altérer par l'utilisateur. Pour voir ces valeurs par défaut, consulter la documentation intégrer à R en entrant:

```
## ?read.csv
```

Importation avec *R-base*: read.csv2

Avec read.csv déjà, le séparateur peut être spécifié pour permettre la prise en charge des fichiers qui ont point-virgule comme séparateur (“;”). Toutefois, il existe une fonction bâtie au dessus de celle-ci, qui prend en charge les spécificités de ces fichiers. Il s’agit de read.csv2.

```
RGPH_MLI2 <- read.csv2(
  file = "data/RGPH_MLI2.csv", # spécifier le nom du fichier
  header = TRUE, # déclarer la première ligne comme nom de colonne/variable
  sep = ";", # déclarer la virgule comme séparateur
  stringsAsFactors = TRUE # voir cours sur les types de données dans R
)
```

Ici aussi, les valeurs par défaut des arguments sont maintenus. Pour en savoir plus sur read.csv2, explorer la documentation R en tapant:

```
## ?read.csv2
```

Importation avec *readr*: aperçu

readr est un package créé par Hadley Wickham. Ses fonctions sont similaires à *read.csv* et *read.csv2*. *readr* présente l'avantage de faire partie du *tidyverse*. Il travaille harmonieusement avec les autres packages de cet écosystème. Sa syntaxe est très simple: pour importer un fichier csv, on utilise *read_csv* au lieu de *read.csv*. Cette logique est valable pour d'autres formats, dont il suffit seulement d'ajouter l'extension après le tiret d'en bas ("_"). Ainsi, on a :

- *read_csv2*: pour les fichiers CSV ayant le point-virgule (";") comme séparateur
- *read_tsv*: pour les fichiers avec les valeurs séparées par des tabulations
- *read_fwf*: pour les fichiers avec les valeurs séparées par des espaces fixes

Importation avec *readr*: *read_csv* et *read_csv2*

Ouvrons le fichier csv avec *read_csv*

```
library(readr) # chargement du package "readr"
RGPH_MLI <- read_csv(
  file = "data/RGPH_MLI.csv", # spécifier le nom du fichier
  col_names = TRUE, # déclarer la première ligne comme nom de colonne/variable
  na = "", # indiquer la valeur à attribuer aux cellules vides
  skip = 0 # nombre de lignes à ne pas importer, partant du fichier
)
```

et *read_csv2*

```
library(readr) # chargement du package "readr"
RGPH_MLI2 <- read_csv2(
  file = "data/RGPH_MLI2.csv", # spécifier le nom du fichier
  col_names = TRUE, # déclarer la première ligne comme nom de colonne/variable
  na = "", # indiquer la valeur à attribuer aux cellules vides
  skip = 0 # nombre de lignes à ne pas importer, partant du fichier
)
```

Importation avec *readr*: généralisation

read_csv est un cas spécifique d'une fonction généraliste qui couvrent un large spectre de formats : *read_delim*.

```
library(readr) # chargement du package "readr"
RGPH_MLI <- read_delim(
  file = "data/RGPH_MLI.csv", # spécifier le nom du fichier
  delim = ",", # indiquer le séparateur
)
RGPH_MLI2 <- read_delim(
  file = "data/RGPH_MLI2.csv", # spécifier le nom du fichier
  delim = ";" # indiquer le séparateur
)
```

Avec *read_delim*, divers format de fichiers peuvent être importés, à partir du moment où le séparateur est bien spécifié: virgule (","), point-virgule (";"), tabulation ("^"), barre verticale ("|"), espace (" "), etc. Les autres arguments peuvent être ajustés pour prendre en compte les spécificités des données. Toutefois, il faut noter que *read_delim* est assez intuitive pour détecter le type des données (entiers, réels, caractères, etc.). Elle inspecte jusqu'à 1000 lignes (et peut aller jusqu'à la nième ligne) pour déterminer le type de données d'une colonne.

Excel: *xls*, *xlsx*

Aperçu

Faisant partie de la suite MS Office, Excel est l'un des tableurs les plus populaires. Il est difficile de l'exclure du *workflow* du *data scientist* car c'est l'outil de prédilection d'organisation et de sauvegarde des données dans beaucoup de domaines et secteurs (entreprises, administrations). Les *data scientist* opérant exclusivement sur des bases de données constituent une niche. Pour la majorité, amenée à interagir avec des spécialistes d'autres domaines, il est important de pouvoir disposer de techniques pour importer les données collectées et organisées par leurs soins dans Excel.

Dans R, diverses solutions existent. Nous verrons deux packages: *xlsx* et *readxl*.

Importation avec xlsx: read.xlsx

Le package xlsx offre deux fonctions majeures pour lire des formats xls et xlsx. Il s'agit de *read.xlsx* et *read.xlsx2* (plus rapide sur les fichiers lourds).

```
library(xlsx) # chargement du package "xlsx"
# Exemple avec fichier Excel 97-2003
RGPH_MLI_xls <- read.xlsx(
  file = "data/RGPH_MLI.xls", # spécifier le nom du fichier (format "xls")
  sheetIndex = 1, # indiquer le numéro d'ordre de l'onglet à importer
  # sheetName = "RGPH_MLI" # indiquer le nom de l'onglet (alternative au numéro d'ordre)
  header = TRUE # déclarer la première ligne comme nom de colonne/variable
)
# Exemple avec fichier Excel 2007-plus
RGPH_MLI_xlsx <- read.xlsx(
  file = "data/RGPH_MLI.xlsx", # spécifier le nom du fichier (format "xlsx")
  sheetIndex = 1, # indiquer le numéro d'ordre de l'onglet à importer
  # sheetName = "RGPH_MLI" # indiquer le nom de l'onglet (alternative au numéro d'ordre)
  header = TRUE # déclarer la première ligne comme nom de colonne/variable
)
```

Importation avec *readxl*: *read_excel*

Le package *readxl* est bâti sur les mêmes principes que *readr* [voir]. Il permet d'importer des formats Excel avec la même logique syntaxique. Il contient des fonctions spécifiques, *read_xls* et *read_xlsx*, et une fonction généraliste, *read_excel*

```
library(readxl) # chargement du package "readxl"
# Exemple avec fichier Excel 97-2003
RGPH_MLI_xls <- read_excel(
  path = "data/RGPH_MLI.xls", # spécifier le nom du fichier (format "xls")
  sheet = "RGPH_MLI", # indiquer le nom de l'onglet ou le numéro d'ordre, les deux sont acceptés
  col_names = TRUE)
# Exemple avec fichier Excel 2007-plus
RGPH_MLI_xlsx <- read_excel(
  path = "data/RGPH_MLI.xlsx", # spécifier le nom du fichier (format "xls")
  sheet = "RGPH_MLI", # indiquer le nom de l'onglet ou le numéro d'ordre, les deux sont acceptés
  col_names = TRUE)
```

Formats issues d'autres logiciels statistiques: Stata et SPSS

Aperçu

Le data scientist peut aussi être amené à travailler sur des données sauvegardées à partir d'autres programmes de traitement de données tels que Stata, SPSS, SAS, etc. Dans R, les solutions sont nombreuses. Ici, nous allons voir deux packages: *foreign* et *haven*.

Importation avec *foreign*

Le package *foreign* permet d'ouvrir des fichiers *dta*, issus de Stata...

```
library(foreign) # chargement du package "foreign"
RGPH_MLI_stata <- read.dta(
  file = "data/RGPH_MLI.dta", # spécifier le nom du fichier
  convert.dates = TRUE, # conversion des dates du format Stata au format R (pour dire simple)
  convert.factors = TRUE, # conversion des étiquettes de variables catégorielles en facteurs
  convert.underscore = FALSE, # convertir "_" de Stata en "." dans R (surtout nom des variables)
  missing.type = FALSE, # donner ou pas une valeur particulière aux cellules vides?
  warn.missing.labels = TRUE # si oui, cet argument indique si cette valeur est présente ou pas
)
```

et des fichiers *sav*, issus de SPSS

```
library(foreign) # chargement du package "foreign"
RGPH_MLI_spss <- read.spss(
  file = "data/RGPH_MLI.sav", # spécifier le nom du fichier
  use.value.labels = TRUE, # conservation des étiquettes de variables catégorielles
  to.data.frame = TRUE # importation en data frame au lieu de matrice
)
```

Base de données relationnelles

Aperçu

R peut aussi importer des données depuis une base de données relationnelles. Celle-ci peut aussi bien être locale, installée sur le poste de travail, que distante, installée sur un serveur accessible via Internet. Nous allons illustrer ici avec une base locale.

Les packages varient d'un type de base à un autre:

- *RMySQL* pour MySQL,
- *RPostgreSQL* pour PostgreSQL
- *RSQLite*: pour SQLite
- etc.

Pour plus de détails, consulter cette page.

Importation avec *RODBC*

Commençons par ouvrir le chaîne de communication entre R et la base via ODBC (*Open Data Base Connectivity*).

```
library(RODBC) # chargement du package "RODBC"
dswr <- odbcConnect(
  dsn = "dswr", # indiquer le nom de la chaîne de connection
  uid = "", # indiquer l'identifiant (s'il y'en a)
  pwd = "" # indiquer le mot de passe (s'il y'en a)
)
```

Maintenant, importons la table qui nous intéresse, *RGPH_MLI*. Pour cela, deux méthodes sont possibles:

```
# Méthode 1: extraction de la table
RGPH_MLI_rodbs_tbl <- sqlFetch(
  dswr, # indiquer le nom de la chaîne de connection
  sqtable = "RGPH_MLI" # indiquer le nom de la table
)

# Méthode 2: requête SQL
RGPH_MLI_rodbs_sql <- sqlQuery(
  channel = dswr, # indiquer le nom de la chaîne de connection
  query = "SELECT * FROM RGPH_MLI;" # sélectionner toutes les colonnes et lignes de la table "RGPH_MLI"
)
```

Une fois les extractions de données finie, il faut penser à briser la chaîne de connection, fermer la porte.

Importation avec *odbc*

Une autre solution est de passer par le package *odbc*.

```
library(odbc) # chargement du package "odbc"
library(DBI) # chargement du package "DBI"
dswr <- dbConnect(
  odbc::odbc(), # indiquer le package utilisé par l'interface "DBI"
  "dswr" #, indiquer le nom de la chaîne de connection
  # dbname = "",
  # host = "",
  # user = "", # indiquer l'identifiant (s'il y'en a)
  # password = "" # indiquer le mot de passe (s'il y'en a)
)
```

Comme avant, on peut importer la table d'intérêt par deux méthodes

```
# Méthode 1: extraction de la table
RGPH_MLI_odbc <- dbReadTable(
  conn = dswr, # indiquer le nom de la chaîne de connection
  name = "RGPH_MLI" # indiquer le nom de la table à importer
)

# Méthode 2: requête SQL
RGPH_MLI_odbc_sql <- dbGetQuery(
  conn = dswr, # indiquer le nom de la chaîne de connection
  "SELECT * FROM RGPH_MLI;" # sélectionner toutes les colonnes et lignes de la table "RGPH_MLI"
)
```

Comme avant, en bon invité, on ferme la porte en sortant...

Depuis Internet

Aperçu

Les données peuvent aussi être tirées de la toile mondiale. Les outils disponibles dans R varient selon le type de données.

Chargement de fichier CSV

Pour un fichier CSV, le chargement dans l'environnement R se fait de la même façon que pour des fichiers locaux.

```
url <- "https://github.com/fousseynoubah/dswr/blob/master/data/rgph_groupage/RGPH_MLI.csv"  
RGPH_MLI_csv_online <- read.csv(url)
```

Chargement de fichier Excel

Pour le format Excel, la procédure est différente. Les données ont besoin d'être téléchargées localement avant d'être chargées dans l'environnement R.

```
url <- "https://github.com/fousseynoubah/dswr/blob/master/data/rgph_groupepage/RGPH_MLI.xlsx?raw=true"
download.file(url = url, # adresse du fichier
              destfile = "data/RGPH_MLI_github.xlsx", # nom du fichier une fois téléchargé
              mode = "wb" # mode de téléchargement
              )
knitr::asis_output("\\\\normalsize")
```