Import et export de données

Les différents types et formats de données pris en charge par R

Hugues pecout

03/02/2023

Table des matières

De nombreuses de fonctions (primitives ou non) permettent d'importer et d'exporter des données de différents formats.

0.1 Tableau de données

0.1.1 Fichier texte simple

Un fichier texte simple (ou fichier texte brut) est un fichier dont le contenu représente uniquement une suite de caractères. Il peut s'ouvrir avec n'importe quel éditeur de texte et utilise nécessairement une forme particulière de codage des caractères.

Plusieurs fonctions primitives permettent d'importer et d'exporter des fichiers texte simples, comme les fichiers **csv**, **txt** , **tsv**…

0.1.1.1 Import

- `read.delim()` : fichiers délimités par un symbole quelconque et "." en séparateur décimal
- `read.delim2()` : fichiers délimités par un symbole quelconque et "," en séparateur décimal
- `read.table()` : pour des fichiers texte délimités par des espaces

- `read.csv()` : pour des fichiers texte délimités par des virgules (format csv)
- `read.csv2()` : pour des fichiers texte délimités par des points-virgules (format csv français)

Pour que l'import de données s'effectue correctement, il est parfois nécessaire de renseigner plusieurs arguments, comme par exemple :

header = valeur logique qui indique si la première ligne contient les noms des variables. **sep** = indique le caractère utilisé comme séparateur de champ (ex : ";") **encoding** = Chaîne de caractère qui précise l'encodage utilisé pour le fichier (ex : "UTF-8").

> **!** Important
>
> N'oubliez pas d'assigner le résultat dans un objet pour garder en mémoire vos données importées.

```
# Exemple d'utilisation de read.table()
mon_tableau  <- read.table(file = "../data/DEV_AFRICA_2018/afrika_don_meta.csv",
                           header = TRUE,
                           sep= ";",
                           encoding = "UTF-8")


# Le tableau importé est stocké dans un objet data.frame
class(mon_tableau)
```

```
[1] "data.frame"
```

0.1.1.2 Export

Des fonctions primitives permettent également d'exporter votre tableau de données vers différents format texte.

- `write.table()` : pour tous les types de formats texte simple (séparateur à renseigner)
- `write.csv()` : pour exporter en csv (séparateur virgule)
- `write.csv2()` : pour exporter en csv (séparateur points-virgules)

```r
# Exemple write.table()
write.table(x = mon_tableau,
            file = "../data/tableau.txt",
            sep = "\t", col.names = TRUE,
            fileEncoding = "UTF-8")

# Exemple write.csv()
write.csv(x = mon_tableau, file = "../data/tableau.csv")
```

0.1.2 Fichier Excel

Il est parfois nécessaire d'importer des tableaux de données stockées dans un format propriétaire, comme par exemple Excel (xls, xlsx) ou SAS. Plusieurs packages vous permettent d'importer ce genre de format, et même d'exporter vos données dans ce type de format.

0.1.2.1 Import

Vous pouvez par exemple importer un fichier Excel avec le package `readxl`.

```r
install.packages("readxl")
```

> **!** Important
>
> Le packages **readxl** fait partie de l'écosystème **tidyverse** (cf. module x). Pour cette raison, le tableau importé est mis en mémoire dans un objet `tibble` et non **dataframe**. Il s'agit de deux objets très semblables mais pas identiques. Pour convertir un **tibble** en **dataframe**, vous pouvez utiliser la fonction **as.data.frame()**.

```r
library(readxl)
mon_tableau <- read_excel(path = "../data/DEV_AFRICA_2018/afrika_don.xls",
                          sheet = "afrika_meta",
                          skip = 0,
                          col_names = TRUE)

# Le tableau importé est stocké dans un objet data.frame
class(mon_tableau)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

0.1.2.2 Export

Pour exporter un dataframe (ou tibble) dans un fichier au format Excel, vous pouvez utiliser le package `openxlsx` et sa fonction `write.xlsx()`.

```
install.packages("openxlsx")
```

> **!** Important
>
> Le packages `openxlsx` permet uniquement de lire et écrire les fichiers Excel comportant l'extension **.xlsx**.

```
library(openxlsx)
write.xlsx(x = mon_tableau, file = "../data/DEV_AFRICA_2018/afrika_don.xlsx")
```

0.1.3 SAS, SPSS & Stata

Le package `haven` permet de gérer des fichiers propriétaires de différents formats comme SAS, SPSS, Stata, dbf...

```
install.packages("haven")
```

> **!** Important
>
> Tout comme `readxl`, ce package fait partie de l'écosystème `tidyverse` (cf. module x). Le tableau importé est mis en mémoire dans un objet `tibble` et non `dataframe`. Vous pouvez utiliser la fonction `as.data.frame()` pour le convertir.

0.1.3.1 Fichier SAS

```
library(haven)

# Import
mon_tableau  <- read_sas(data_file = "../data/data_sas.sas7bdat")
```

```
# Export
write_sas(data = mon_tableau, path = "../data/mon_tableau_sas.sas7bdat")
```

0.1.3.2 Fichier SPSS

```
library(haven)

# Import
mon_tableau  <- read_sav(file =  "../data/data_spss.sav")

# Export
write_sav(data = mon_tableau, path =  "../data/mon_tableau_spss.sav")
```

0.1.3.3 Fichier Stata

```
library(haven)

# Import
mon_tableau  <- read_stata(file =  "../data/data_stata.dta")

# Export
write_dta(data = mon_tableau, path =  "../data/mon_tableau_stata.dta")
```

0.2 Couche géographique

0.2.1 Vecteur

Le package sf permet de lire les différents formats de couche géographique vectorielle (ESRI Shapefile, GeoJSON, Keyhole Markup Language, GeoPackage…). Pour cela, ce package interface avec la librairie système GDAL. Les SIG fonctionnent de la même façon !

```
install.packages("sf")
```

> **Note**
>
> L'installation du package `sf` demande un prérequis. La librairie système `GDAL` doit être installée sur votre machine. Il est parfois nécessaire de la faire soi-même sur certains système d'exploitation comme GNU/Linux.

0.2.1.1 Import

```
library(sf)
map_africa <- st_read("../data/GADM_AFRICA_2020/afrika_map.shp",  quiet = TRUE)

class(map_africa)
```

```
[1] "sf"         "data.frame"
```

> **Important**
>
> Une couche géographique importée via la fonction `st_read()` du package `sf` est mise en mémoire dans un objet sf (simple feature). Il s'agit en quelque sorte d'un dataframe ou chaque individu est associé à une géométrie.

0.2.1.2 Export

La fonction `st_write()` permet d'enregistrer un objet sf sur sa machine, dans le format que l'on souhaite.

```
library(sf)

# Enregistrement en format ESRI Shapefile
st_write(obj = map_africa,
         dsn = "../data/map_africa.shp",
         layer_options = "ENCODING=UTF-8")


# Enregistrement en format GeoPackage
st_write(obj = map_africa,
         dsn = "../data/map_africa.gpkg",
         layer = "pays")
```

0.2.2 Raster

Le package `terra` permet aussi de lire et d'écrire des données géographiques vectorielles (comme `sf`) mais sa valeur ajoutée se situe au niveau de la manipulation de données raster.

```
install.packages("terra")
```

0.2.2.1 Import

Pour importer des données Raster, vous pouvez utiliser la fonction `rast()`.

> **!** Important
>
> Un Raster importé via la fonction `rast()` du package `terra` est mis en mémoire dans un objet SpatRaster.

```
library(terra)
Elevation_Benin <- rast("../data/elevation.tif")

Elevation_Benin
```

```
class       : SpatRaster
dimensions  : 742, 369, 1  (nrow, ncol, nlyr)
resolution  : 0.008339098, 0.008333169  (x, y)
extent      : 0.774574, 3.851701, 6.23514, 12.41835  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : elevation.tif
name        : elevation
```

0.2.2.2 Export

La fonction `writeRaster()` permet d'enregistrer un objet SpatRaster sur sa machine, dans le format que l'on souhaite.

```
library(terra)
writeRaster(x = Elevation_Benin, filename = "../data/Benin_Elevation.tif")
```

0.3 Image

0.3.1 Import

Pour importer des images, le package `pnget` `jpeg`, pré-installés avec le language R vous permet d'importer des images.

0.3.1.1 PNG

```
library(png)
mon_image_png <- readPNG("../img/map.png")
```

0.3.1.2 JPEG

```
library(jpeg)
mon_image_jpg <- readJPEG("../img/wip.jpg")
```

0.3.2 Export

Il est également possible d'exporter les sorties graphiques en format image avec les fonctions primitives suivantes :

- `bmp()`
- `jpeg()`
- `png()`
- `tiff()`

Ces fonctions doivent être utilisées avec la fonction `dev.off()`. Exemple :

```
# Ouverture de la création de l'image
png(filename = "../img/mon_image.jpg")

# Création de la représentation graphique souhaitée
plot(1:10)

# Fermeture (enregistrement) de l'image png
```

```
dev.off()
```

> **!** Important
>
> La fonction `dev.off()` permet de clôturer la représentation graphique et d'enregistrer l'image. Si aucun dev.off() n'est exécuté à la suite de ces fonctions d'export, votre fenêtre graphique restera figée, et inutilisable.

Il est également possible d'exporter vos représentations graphiques dans un format vectoriel, qui permet leur retouche avec des logiciel de DAO (Inkscape, Adobe Illustrator…). Les fonction pour réaliser cela sont :

- `pdf()`
- `svg()`

L'utilisation de ces fonctions primitives est similaire à l'export d'images matricielles. Exemple :

```
# Ouverture de la création de l'image
pdf(file = "../img/mon_image.pdf")

# Création de la représentation graphique souhaitée
plot(1:10)

# Fermeture (enregistrement) du pdf
dev.off()
```

0.4 Exercice

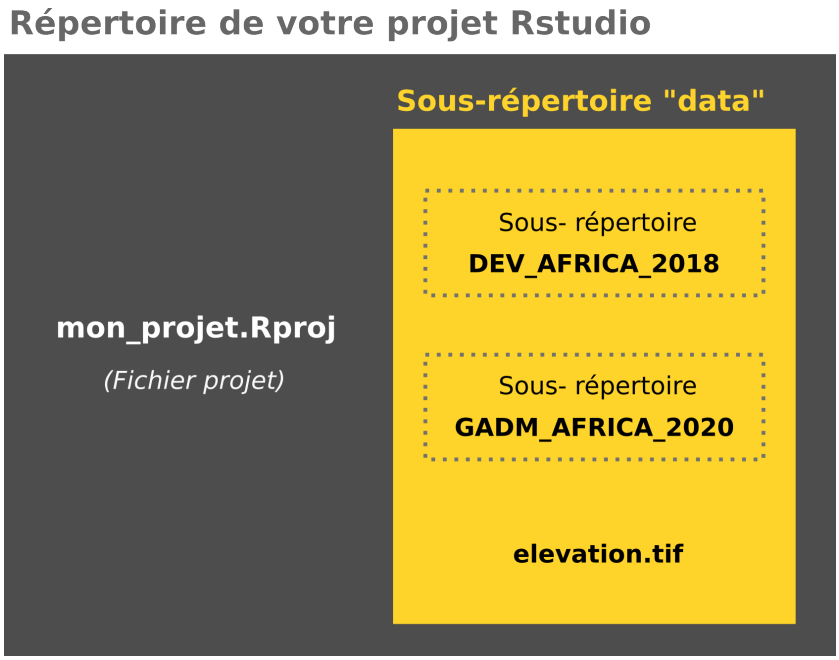
1. Créez un projet Rstudio

File/New Project/New Directory…

2. Téléchargez les données suivantes :

Intitulé	Téléchargement
Données pays africains (UN-CEPII)	Download
Fond de carte Afrique (GADM 2020)	Download
Raster délévation du Bénin (???)	Download

3. Placez les données (décompressées) dans le répertoire de votre projet, de la façon suivante :



4. Créez un script R à la racine de votre projet Rstudio

File/New File/R script

5. Importer les fichiers suivants en utilisant les fonctions adéquates :

- data/DEV_AFRIA_2018/**afrika_don.csv**
- data/DEV_AFRICA_2018/**afrika_don.xls** (1er onglet)
- data/GADM_AFRICA_2020/**afrika_map.shp**
- data/**elevation.tif**

```
# Pour importer un fichier csv (afrika_don.csv)
read.csv()
read.csv2()
```

```r
# Pour importer un fichier Excel (afrika_don.xls)
library(readxl)
read_excel()

# Pour importer un fichier ESRI Shapefile (afrika_map.shp)
library(sf)
st_read()
```

```r
# Import du fichier csv "afrika_don.csv"
data_from_csv <- read.csv2(file = "data/DEV_AFRICA_2018/afrika_don.csv")

# Import du fichier xls "afrika_don.xls"
library(readxl)
data_from_xls <- read_excel(path = "data/DEV_AFRICA_2018/afrika_don.xls")

# Importer du fichier shp "afrika_map.shp"
library(sf)
data_from_shp <- st_read("data/GADM_AFRICA_2020/afrika_map.shp",  quiet = TRUE)
```

Télécharger ce document format PDF