Science des données I : module 5

Importation de données

Philippe Grosjean & Guyliann Engels

Université de Mons, Belgique
Laboratoire d'Écologie numérique des Milieux aquatiques

http://biodatascience-course.sciviews.org
sdd@sciviews.org

UMONS

Données locales

UMONS

Petits jeux de données : création dans le R Markdown

Les petits jeux de données peuvent être créés directement dans le R Markdown où ils sont analysés, par exemple, à l'aide de la fonction `tribble()`.

```
small_dataset <- tribble(
  ~treatment, ~dose, ~response,
   ”control”,   0.5,     18.35,
   ”control”,   1.0,     26.43, # Ceci est un commentaire
   ”control”,   2.0,     51.08,
    ”test”  ,   0.5,     10.29,
    ”test”  ,   1.0,     19.92,
    ”test”  ,   2.0,     41.06)
```

Dans ce cas, aucun problème d'**accessibilité**, ni de choix concernant l'**endroit où stocker les données**.

UMONS

Jeux de données de taille moyenne : dossier `data`

Pour des jeux de données de taille moyenne **qui sont spécifiques au projet**, ils peuvent être stockés dans les projet.

- Utiliser le **sous-répertoire `data`** pour une organisation compréhensible,
- Choisir le **format** le plus adéquat, privilégier CSV, TSV, XLS ou XLSX.
- Utiliser `read()`avec un **chemin relatif** vers les données.
- Pas de problèmes d'**accessibilité**. Données physiquement dans le projet
- Dépôt Git/Github : taille petite à moyenne et peu de mises-à-jour => OK.

Exemple

```
projet-exemple
 |- analysis
 |    rapport-exemple.Rmd
 |- data
      local-data-exemple.csv
```

On aura l'instruction suivante dans `rapport-exemple.Rmd` pour charger les données :

```
df <- read("../data/local-data-exemple.csv")
```

Gros jeux de données locales et/ou données externes au projet

Pour des gros jeux de données locales ou des données partagées entre plusieurs projets, **il vaut mieux placer les données sur le Net**, en utilisant éventuellement des dépôts spécialisés tels figshare ou Zenodo, ou une **base de données**.

- Pas de stockage de gros fichiers (binaires) dans un dépôt Git/Github !
- On allège notre disque dur en évitant les **multiples copies locales** des données
- Mais aussi, on évite de se référer à des données stockées sur le disque local **en dehors du projet**

Solution alternative

Dédier **un dossier réseau sur un serveur** pour le stockage des données partagées. **Dans ce cas, les projets qui dépendent de ces données ne sont portables et reproductibles que pour autant qu'on garde l'accès au dossier partagé !** Solution acceptable pour des analyses qui ne sortiront jamais de l'entreprise.

UMONS

Données distantes

UMONS

Importation manuelle des données sur son disque

L'approche *naturelle* consiste à vouloir **télécharger** les données et enregistrer une **copie locale** dans le projet pour ensuite y accéder comme d'habiture (fonction `read()`).

C'est une très mauvaise idée : le lien vers les données d'origine est *perdu* dès lors que l'enregistrement local est réalisé *manuellement !*

Exemple : feuille Google Sheet partagée

https://docs.google.com/spreadsheets/d/1iEuGrMk4IcCkq7gMNzy04DkSaPeWH35Psb0E56KEQMw

Dans Google Sheet, on serait tenté de faire :

- **Add to my drive**, et puis copier le fichier à la souris dans le projet
- **File -> Download as -> Microsoft Excel (.xlsx)**, et puis déplacer le fichier à la souris dans le projet

L'origine des données (l'URL vers la feuille Google Sheet) est perdu dans le processus => non reproductible !

UMONS

Importation depuis le R Markdown : bonne solution

- La fonction `read()` permet **aussi** d'importer directement des données depuis un lien URL pour certains formats tels que CSV.
- La difficulté est généralement de déterminer quelle est l'URL à employer pour **accéder directement aux données** depuis la source dans un **format** adéquat.
- Problème d'**accessibilité** au cas où Internet n'est pas disponible, ou bien si les données disparaissent du Net. **Solution :** enregistrer *aussi* une copie locale (ajouter `.rds` dans `.gitignore` pour sortir ce fichier de la gestion de version).

Example

Code d'importation et sauvegarde dans un chunck avec `eval=FALSE` :

```
google_sheet_id <- "1iEuGrMk4IcCkq7gMNzy04DkSaPeWH35Psb0E56KEQMw"
url <- "https://docs.google.com/spreadsheets/d/{id}/export?format=csv"
coral <- read$csv(glue::glue(url, id = google_sheet_id)) # Téléchargement
write$rds(coral, file = "../data/coral_growth.rds") # Copie locale
```

Ensuite, lire depuis la version locale dans un autre chunk avec `eval=TRUE` :

```
coral <- read("../data/coral_growth.rds")
```

R Markdown séparé pour l'importation des données

Lorsque **plusieurs sources** doivent être consolidées ou si le **remaniement** préliminaire des données est lourd, le code dans le premier chunk risque d'être très long.

- Il vaut mieux sortir ce code d'importation du fichier R Markdown d'analyse.
- On peut utiliser un **R Markdown séparé** pour l'importation

Astuce

Nommer les R Markdowns avec des numéros pour qu'ils apparaissent dans l'ordre dans lequel ils doivent être exécutés :

```
projet-exemple
 |    .gitignore
 |- analysis
 |    corail01_importation.Rmd
 |    corail02_analyse.Rmd
 |- data
      coral_growth.rds
```

Script R dédié à l'importation des données

- On peut aussi utiliser un **script R** pour l'importation

Astuce

```
projet-exemple
 |    .gitignore
 |- analysis
 |    coral_analyse.Rmd
 |- data
 |    coral_growth.rds
 |- R
      coral_growth_importation.R
```

Dans le R Markdown d'analyse, on peut ajouter un chunk en `eval=FALSE` qui **source** le code du script R d'importation :

```
source("../R/coral_growth_importation.R")
```

Métadonnées

UMONS

Ajout des labels et unités dans des données importées

- On peut utiliser la fonction `labelise()` pour ajouter des labels et unités pour chaque variable du jeu de données importé.
- Utiliser un format de stockage intermédiaire qui **préserve** les métadonnées, comme RDS.

Exemple

```
google_sheet_id <- "1iEuGrMk4IcCkq7gMNzy04DkSaPeWH35Psb0E56KEQMw"
url <- "https://docs.google.com/spreadsheets/d/{id}/export?format=csv"
coral <- read$csv(glue::glue(url, id = google_sheet_id)) # Téléchargement
# Ajout des labels et unités
coral <- labelise(coral, self = FALSE,
  label = list(
    id = "Identifiant",
    species = "Espèce",
    gain_std = "Gain de masse standardisé"),
  units = c(gain_std = "mg/J")
)
write$rds(coral, file = "../data/coral_growth.rds") # Copie locale
```