Chapitre 3 Lire des données

3.1 readxl : lire des données Excel

La fonction *read_excel()* permet d'importer les données d'un fichier Excel. On peut spécifier :

- la feuille, les colonnes, les lignes ou la zone à importer
- les lignes à supprimer avant importation
- si on souhaite importer la première ligne comme des noms de variables ou non
- le format des variables importées
- la valeur qui sera interprétée comme étant la valeur manquante

```
sitadel <- read_excel ("data/ROES_201702.xls", sheet = "AUT_REG",
                       col_types = c ("text","text","numeric","numeric","numeric","numeric"
datatable (sitadel)
```

Show 10 entries Search:

	date	REG	log_AUT	ip_AUT	ig_AUT	colres_AUT
1	200001	01	440	194	12	234
2	200002	01	564	228	18	318
3	200003	01	348	220	19	109
4	200004	01	315	220	42	53
5	200005	01	390	250	66	74
6	200006	01	749	269	214	266
7	200007	01	420	185	76	159
8	200008	01	578	243	19	316
9	200009	01	496	299	42	155
10	200010	01	569	238	32	299

Showing 1 to 10 of 5,356 entries

Previous 1 2 3 4 5 … 536 Next

3.2 read_delim : lire des fichiers plats

La fonction *read_delim()* permet d'importer les données d'un fichier csv. Elle fonctionne de la même façon que *read_excel()*. On peut spécifier :

- le délimiteur de colonne
- les lignes à supprimer avant importation
- si on souhaite importer la première ligne comme des noms de variables ou non
- le *locale* du fichier
- la valeur qui sera interprétée comme étant la valeur manquante

read_csv(), *read_csv2()* et *read_tsv()* sont des implémentations prérenseignées de read_delim pour lire des fichiers plats avec séparateurs **,** **;** et **tabulaire**.

3.3 Télécharger des données disponibles sur le web

Parfois, les données que nous exploitons sont disponibles sur le web. Il est possible, directement depuis R, de télécharger ces données et, si nécessaire, de les décompresser (dans le répertoire de travail). Exemple sur les données SEQUOIA de l'ACOSS :

```
url <- "http://www.acoss.fr/files/Donnees_statistiques/SEQUOIA_TRIM_REGION.zip"
download.file (url, destfile = "SEQUOIA_TRIM_REGION.zip", method = "auto")
unzip (zipfile = "SEQUOIA_TRIM_REGION.zip")
SEQUOIA <- read_excel ("SEQUOIA_TRIM_REGION_BRUT.xls", sheet = "PAYS_DE_LA_LOIRE")
datatable (SEQUOIA)
```

3.4 Lire des fichiers avec une dimension spatiale

Le package sf (pour simple feature) permet d'importer dans R un fichier ayant une dimension spatiale. Après importation, le fichier est un dataframe avec une variable d'un type nouveau : la géométrie. Deux exemples ici pour lire des données au format shape et geojson.

```
Carte_EPCI_France <- st_read (dsn = "refgeo2017", layer = "Contour_epci_2017_region")
plot (Carte_EPCI_France)

communes2017 <- st_read (dsn = "refgeo2017/communes2017.geojson")
plot (communes2017)
```

Le package sf contient l'ensemble des fonctions permettant des manipulations sur fichiers géomatiques. On ne traitera pas ici de toutes ces fonctions en détail, mais la documentation se trouve ici.

A noter que sf étant complètement compatible avec les packages du tidyverse, la géométrie se conçoit comme une une donnée comme une autre, sur laquelle par exemple on peut réaliser des aggrégations.

3.5 Lire des données sous PostgreSQL

Deux approches possibles pour lire les données du patrimoine de la Dreal :

- *Importer* toutes ces données dans l'environnement R
- se *connecter* à ces données et utiliser un interpréteur permettant de traduire du code R comme une requête SQL.

3.5.1 Lire des données sous PostgreSQL : première approche

```
#Définition du driver
drv <- dbDriver ("PostgreSQL")

#Définition de la base de données
con <- dbConnect (drv, dbname = "dbname", host = "ip", port = numero_du_port,
                  user = "user_name", password = "pwd")

#Spécification de l'encodage, obligatoire avec Windows
postgresqlpqExec (con, "SET client_encoding = 'windows-1252'")

#Téléchargement de la table analyse du schéma pesticide
parametre <- dbGetQuery (con, "SELECT * FROM pesticides.parametre")

#Téléchargement de données avec dimension spatiale via la fonction st_read_db du package
station = st_read_db (con, query = "SELECT * FROM pesticides.station")
```

On voit que pour importer notre table analyse, on a simplement lancé une requête SQL. On peut bien sûr avec la même fonction lancer n'importe quelle requête sur la base et reccueillir le résultat.

3.5.2 Lire des données sous PostgreSQL : seconde approche

```
#définition du driver
drv <- dbDriver ("PostgreSQL")

#définition de la base de données
con <- dbConnect (drv, dbname = "dbname", host = "ip", port = numero_du_port, user = "us

#spécification de l'encodage, obligatoire avec windows
postgresqlpqExec (con, "SET client_encoding = 'windows-1252'")

#téléchargement de la table analyse du schéma pesticide
analyse_db <- tbl (con, in_schema ("pesticides", "analyse"))
```

Ici la table *analyse* n'est pas chargée dans l'environnement R, R s'est juste *connecté* à notre base de données.

On peut réaliser des opérations sur la table analyse avec du code R très simplement, par exemple ici pour filtrer sur les analyses relatives au Glyphosate :

```
analyse_db <-  filter (analyse_db, code_parametre == 1506)
```

Attention, ce code ne touche pas la base de donnée, il n'est pas exécuté. Pour l'exécuter, il faut par exemple afficher la table.

```
analyse_db
```

Même une fois le code exécuté, cette base n'est pas encore un dataframe. Pour importer la table, on utile la fonction **collect()**

```
analyse_db <- collect (analyse_db)
```

Cette approche est à conseiller sur d'importantes bases de données, et sans dimension spatiale, car dbplyr ne sait pas encore lire ce type de variable (ce qui ne saurait tarder).

3.6 Lire des données du webservice Insee

L'insee met à disposition un webservice d'accès à des données de référence sous un format appelé sdmx. Un package r, rsdmx permet de se connecter directement à ces données. Deux approches sont possibles. La première permet d'accéder à une série particulère.

```
url <- "https://bdm.insee.fr/series/sdmx/data/SERIES_BDM/001564471"
datainsee <- as.data.frame (readSDMX (url))
```

Cette approche peut être utilisée pour télécharger plusieurs séries en même temps. Ici par exemple nous téléchargeons l'ensemble des données sur les créations et défaillances d'entreprises pour les secteurs de la construction et de l'immobilier sur les Pays de la Loire.

```
url <- "https://bdm.insee.fr/series/sdmx/data/SERIES_BDM/001564471+001564503+001564799+(
datainsee <- as.data.frame (readSDMX (url))
```

L’autre approche permet de télécharger un ensemble de données d’une thématique appelé dataflow. Ici, par exemple, on va télécharger l’ensemble des données relatives à la construction neuve :

```
url <- "https://bdm.insee.fr/series/sdmx/data/CONSTRUCTION-LOGEMENTS"
datainsee <- as.data.frame (readSDMX (url))
```