```
setwd("D:/WKSP/5_GROUPES/180227_G2R/FORMATION_R/git_g2r/parcours-r/m2_preparation_donnees/V2.2/Préparer ses données avec R et le Tidyverse/")


# ------------------------
# Chapitre 2 Le Tidyverse
# ------------------------

# 2.2 Activer les packages

library (dplyr)
library (tidyr)
library (forcats)
library (lubridate)
library (stringr)
library (RcppRoll)
library (DT)
library (readxl)
library (dbplyr)
library (RPostgreSQL)
library (rsdmx)
library (sf)

# ----------------------------
# Chapitre 3 Lire des données
# ----------------------------

# 3.1 readxl : lire des données Excel

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

datatable (sitadel)

# 3.2 read_delim : lire des fichiers plats

# 3.3 Télécharger des données disponibles sur le web

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

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)

# 3.5 Lire des données sous PostgreSQL

# 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 simple feature
station = st_read_db (con, query = "SELECT * FROM pesticides.station")

# 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 = "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
analyse_db <- tbl (con, in_schema ("pesticides", "analyse"))

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

analyse_db

analyse_db <- collect (analyse_db)

# 3.6 Lire des données du webservice Insee

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

url <-
"https://bdm.insee.fr/series/sdmx/data/SERIES_BDM/001564471+001564503+001564799+001564823+001582441+001582578+001582597+001582745+001656155+001656161+001655989+001655995"

datainsee <- as.data.frame (readSDMX (url))

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

#---------------------------------
# Chapitre 4 Manipuler des données
#---------------------------------

# 4.3 Chargement des données

load (file = "data/FormationPreparationDesDonnées.RData")

# 4.4 Les verbes clefs de dplyr pour manipuler une table

# 4.4.1 Sélectionner des variables : select()

prelevementb <- select (prelevement, date_prelevement, code_prelevement,
                        code_reseau, code_station)

datatable (prelevementb)
```

```
prelevementb <- select (prelevement, -commentaire)

names (prelevementb)

prelevementb <- select (prelevement, starts_with ("code_"))

prelevementb <- select (prelevement, starts_with ("code_"), one_of ("date_prelevement"))

# 4.4.2 Trier une table : arrange()

prelevementb <- arrange (prelevementb, date_prelevement)

# 4.4.3 Renommer une variable : rename()

prelevementb <- rename (prelevementb, date_p = date_prelevement)

prelevementb <- select (prelevement, date_p = date_prelevement, code_prelevement,
                        code_reseau, code_station)

# 4.4.4 Filtrer une table : filter()

ars <- filter (prelevement, code_reseau == "ARS")
ars <- filter (prelevement, code_reseau == "ARS", code_intervenant == "44")
ars <- filter (prelevement, code_reseau == "ARS" | code_reseau == "FREDON")
ars <- filter (prelevement, code_reseau %in% c ("ARS", "FREDON"))
non_ars <- filter (prelevement, !(code_reseau == "ARS"))

# 4.4.5 Modifier/rajouter une variable : mutate()

prelevementb <- mutate (prelevementb,
                        code_prelevement_caract = as.character (code_prelevement),
                        code_reseau_fact = as.factor (code_reseau))

prelevementb <- mutate (prelevementb,
                        code_prelevement = as.character (code_prelevement),
                        code_reseau = as.factor (code_reseau))

# 4.4.6 Extraire un vecteur : pull()

stations_de_la_table_prelevement <- pull (prelevement, code_station)
stations_de_la_table_prelevement <- unique (stations_de_la_table_prelevement)

# 4.5 La boîte à outils pour créer et modifier des variables avec R

# 4.5.1 Manipuler des variables numériques

x <- sample (1:10)
lagx <- lag (x)
leadx <- lead (x)
lag2x <- lag (x, n = 2)
lead2x <- lead (x, n = 2)
cbind (x = x, lagx = lagx, lag2x = lag2x, leadx = leadx, lead2x = lead2x)

x <- sample (1:10)
cumsumx <- cumsum (x)
rollsumx <- roll_sum (x, n=2)
rollsumx

rollsumrx <- roll_sumr (x, n=2)
rollsumrx

length(rollsumrx) == length(x)

cbind (x = x, cumsumx = cumsum (x), rollsumx = rollsumx, rollsumrx = roll_sumr (x, n=2))

analyseb <- mutate (analyse,
                    resultat_ok = ifelse (code_remarque %in% c (1,2,7,10),
                                          yes = TRUE, no = FALSE))

analyseb <- mutate (analyse, classe_resultat_analyse = case_when (
  resultat_analyse == 0  ~ "1",
  resultat_analyse <= 0.001  ~ "2",
  resultat_analyse <= 0.01  ~ "3",
  resultat_analyse <= 0.1  ~ "4",
  resultat_analyse > 0.1  ~ "5",
  TRUE ~ ""
))

# 4.5.2 Exercice : Les données mensuelles sitadel

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

# corrigé
rm (list = ls ())

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

sitadel52 <- filter (sitadel, REG == "52")

sitadel52 <- mutate (sitadel52,

                     i_AUT = ip_AUT + ig_AUT, # somme des logements individuels autorisés
                     i_AUT_cum12 = roll_sumr (i_AUT, 12), # cumul sur 12 mois
                     i_AUT_cum12_lag12 = lag (i_AUT_cum12, 12), # décalage de 12 mois
                     i_AUT_cum12_delta = i_AUT_cum12 - i_AUT_cum12_lag12,
                     i_AUT_cum_evo = round (100 * i_AUT_cum12_delta /  i_AUT_cum12_lag12, 1),# taux d'évolution

                     log_AUT_cum12 = roll_sumr (log_AUT, 12), # somme des logements autorisés toutes catégories

                     i_AUT_cum_part = round (100 * i_AUT_cum12 / log_AUT_cum12, 1) # part de l'individuel dans le logement total
)

# 4.5.3 Manipuler des dates

dmy ("jeudi 21 novembre 2017")
dmy ("21112017")
ymd ("20171121")
dmy_hms ("mardi 21 novembre 2017 9:30:00")
dmy ("25 décembre 2018") - dmy ("16 avril 2018")
wday (dmy ("19012038"), label = TRUE)

# corrigé
rm (list = ls ())
load (file = "data/FormationPreparationDesDonnées.RData")

exercice <- mutate (exercice,
                    date_prelevement = ymd (date_prelevement),
                    date_creation = ymd (date_creation),
                    date_formatee = format (date_prelevement, "%d/%m/%Y")) # plus joli, mais en texte
```

```r
# 4.5.4 Manipuler des chaînes de caractères

# 4.5.4.1 Manipulations sur les caractères

str_length ("abc")

x <- c ("abcdefg", "hijklmnop")
str_sub (string = x, start = 3, end = 4)
str_sub (string = x, start = 3, end = -2)
str_sub (x, start = 3, end = 4) <- "CC"

# 4.5.4.2 Gestion des espaces

code_insee <- 1001
str_pad (code_insee, 5, pad = "0")

proust <- "   Les paradoxes d'aujourd'hui sont les préjugés de demain.  "
str_trim (proust)
str_trim (proust, side = "left")
txt <- c ("voiture","train", "voilier", "bus", "avion", "tram", "trotinette")
str_detect (string = txt, pattern = "^tr") # les éléments qui commencent pas les lettre "tr"
txt [str_detect (string = txt, pattern = "^tr")]
str_detect (string = txt, pattern = "e$") # les éléments qui terminent par la lettre e
txt [str_detect (string = txt, pattern = "e$")]

# 4.5.4.3 Opérations liées à la langue

proust <- "Les paradoxes d'aujourd'hui sont LES préjugés de Demain."
str_to_upper (proust)
str_to_lower (proust)
str_to_title (proust)

x <- c ("y", "i", "k")
str_order (x)
str_sort (x)

proust2 <- "Les paradoxes d'aujourd'hui sont les préjugés de demain ; et ça c'est embêtant"
iconv (proust2, to = "ASCII//TRANSLIT")

# 4.5.5 Manipuler des variables factorielles (=qualitatives)

library (ggplot2)
library (forcats)
num <- c (1, 8, 4, 3, 6, 7, 5, 2, 11, 3)
cat <- c (letters [1:10])
data <- data.frame (num, cat)

ggplot (data, aes (x = cat, num)) +
  geom_bar (stat = "identity") +
  xlab (label = "Facteur") + ylab (label ="Valeur")

ggplot (data, aes (x = fct_reorder (cat, -num), num)) +
  geom_bar (stat = "identity") +
  xlab (label = "Facteur ordonné") + ylab (label ="Valeur")

# 4.6 Aggréger des données : summarise()
summarise (exercice,
           mesure_moyenne = mean (resultat_analyse, na.rm = T))

summarise (exercice,
           mesure_moyenne = mean (resultat_analyse, na.rm = T),
           mesure_total = sum (resultat_analyse, na.rm = T)
)

# 4.7 Aggréger des données par dimension : group_by()

exercice <- mutate (exercice,
                    annee = year (date_prelevement))

paran <- group_by (exercice, annee)

summarise (paran,
           mesure_moyenne = mean (resultat_analyse, na.rm = T),
           mesure_total = sum (resultat_analyse, na.rm = T)
)

# 4.8 Le pipe
summarise (
  group_by (
    mutate (
      exercice,
      annee = year (date_prelevement)
    ),
    annee
  ),
  mesure_moyenne = mean (resultat_analyse, na.rm = T),
  mesure_total = sum (resultat_analyse, na.rm = T)
)

exercice %>%
  mutate (annee = year (date_prelevement)) %>%
  group_by (annee) %>%
  summarise (mesure_moyenne = mean (resultat_analyse, na.rm = T),
             mesure_total = sum (resultat_analyse, na.rm = T))

# 4.10

rm (list = ls())

sitadel <- read_excel ("data/ROES_201702.xls", sheet = "AUT_REG",
                       col_types = c ("text","text","numeric","numeric","numeric","numeric")) %>%
  group_by (REG) %>%
  mutate (i_AUT = ip_AUT + ig_AUT,
          i_AUT_cum12 = roll_sumr (i_AUT, 12),
          i_AUT_cum12_lag12 = lag (i_AUT_cum12, 12),
          i_AUT_cum12_delta = i_AUT_cum12 - i_AUT_cum12_lag12,
          i_AUT_cum_evo = round (100 * i_AUT_cum12_delta /  i_AUT_cum12_lag12, 1),

          log_AUT_cum12 = roll_sumr (log_AUT, 12),
          i_AUT_cum_part = round (100 * i_AUT_cum12 / log_AUT_cum12, 1)
  )

sitadel <- read_excel ("data/ROES_201702.xls", sheet = "AUT_REG",
                       col_types = c ("text","text","numeric","numeric","numeric","numeric")) %>%
  mutate (annee = str_sub (date, 1, 4),
          i_AUT = ip_AUT + ig_AUT) %>%
  group_by (REG, annee) %>%
  summarise(
    log_AUT_cum = sum (log_AUT),
    i_AUT_cum = sum (i_AUT)) %>%
  ungroup () %>%
  group_by (REG) %>%
```

```
  mutate (i_AUT_cum_lag = lag (i_AUT_cum, 1), # décalage de 1 année
          i_AUT_cum_delta = i_AUT_cum - i_AUT_cum_lag,
          i_AUT_cum_evo = round (100 * i_AUT_cum_delta /  i_AUT_cum_lag, 1),# taux d'évolution

          i_AUT_cum_part = round (100 * i_AUT_cum / log_AUT_cum, 1) # part de l'individuel dans le logement total
  )

# 4.11

rm (list = ls ())
load (file = "data/FormationPreparationDesDonnées.RData")

taux_de_quantification <- exercice %>%
  mutate (year = year (date_prelevement),
          num = 1 * (code_remarque == 1),
          denom = 1 * (code_remarque %in% c (1,2,7,10))) %>%
  group_by (year, code_parametre) %>%
  summarise (taux_de_quantification = 100 * sum (num) / sum (denom))

datatable (taux_de_quantification)

pire_echantillon_par_station_en_2016 <- exercice %>%
  filter (code_remarque == 1, year (date_prelevement) == 2016) %>%
  group_by (libelle_station, code_prelevement) %>%
  summarise (concentration_cumulee = sum (resultat_analyse)) %>%
  group_by (libelle_station) %>%
  filter (concentration_cumulee == max (concentration_cumulee)) %>%
  ungroup ()

datatable (pire_echantillon_par_station_en_2016)

# 4.12 Les armes non conventionnelles de la préparation des donnéees

sitadel <- read_excel ("data/ROES_201702.xls", "AUT_REG") %>%
  group_by (REG) %>%
  mutate_if (is.numeric, funs (cumul12 = roll_sumr (., n = 12))) %>%
  mutate_at (vars (ends_with ("cumul12")), funs (evo = 100 * . / lag (., 12) - 100)) %>%
  mutate_at (vars (ends_with ("cumul12")), funs (part = 100 *./ log_AUT_cumul12))

#--------------------------------------
# Chapitre 5 Manipuler plusieurs tables
#--------------------------------------

rm (list = ls ())
load (file = "data/FormationPreparationDesDonnées.RData")

recalcul_exercice <- analyse %>%
  inner_join (prelevement) %>%
  inner_join (station) %>%
  mutate (date_creation = as.character (date_creation),
          annee = year (date_prelevement))

nb_analyses_presentes_dans_referentiel <- analyse %>%
  inner_join (parametre) %>%
  summarise (n = count (.)) %>%
  pull (n)

nb_analyses_presentes_dans_referentiel2 <- analyse %>%
  inner_join (parametre) %>%
  nrow ()

codes_modecules_absents_du_referentiel <- analyse %>%
  anti_join (parametre) %>%
  group_by (code_parametre) %>%
  tally () # comptage

analyses_avec_code_prelevement_non_retrouve_dans_table_prelevement <- analyse %>%
  anti_join (prelevement) # méthode 1

analyse_avec_code_prelevement_non_retrouve_dans_table_prelevement2 <- analyse %>%
  filter(!(code_prelevement %in% unique(prelevement$code_prelevement)))

sitadel_large <- sitadel_long %>%
  spread (key = ANNEE, value = log_AUT_EVO, sep = "_") # méthode 2

#---------------------------------
# Chapitre 6 Structurer ses tables
#---------------------------------

# 6.2 Les deux fonctions clefs de tidyr

sitadel_long <- read_excel ("data/ROES_201702.xls", "AUT_REG") %>%
  mutate (ANNEE = str_sub (date, 1, 4)) %>%
  group_by (REG, ANNEE) %>%
  summarise_if (is.numeric, funs (sum (., na.rm = T))) %>%
  mutate_if (is.numeric, funs (EVO = 100 * . / lag (.) - 100)) %>%
  select (REG, ANNEE, log_AUT_EVO) %>%
  ungroup ()

sitadel_large <- sitadel_long %>%
  spread (key = ANNEE, value = log_AUT_EVO, sep = "_")

sitadel_long2 <- sitadel_large %>%
  gather (key = annee, value = log_aut_evo, -REG)

#----------------------------------------
# Chapitre 7 Exercice : Les données majic
#----------------------------------------

rm (list = ls ())
library(ggplot2)
load("data/majic.RData")

# 1.1 pour chaque millésime de majic, on remet les données sur la nouvelle carte des territoires et on crée une variable artif

majic_2009 <- bind_rows (majic_2009_com44, majic_2009_com49, majic_2009_com53, majic_2009_com72, majic_2009_com85) %>% #!! on ne voit pas bind_rows dans le cours
  left_join (com2017, by = c ("idcom" = "depcom")) %>%
  select (-idcom, -idcomtxt) %>%
  group_by (epci_2017, depcom2017) %>%
  summarise_all (funs (sum)) %>% #!! à voir
  ungroup %>%
  mutate (artif_2009=dcnt07+dcnt09+dcnt10+dcnt11+dcnt12+dcnt13) %>%
  select(-starts_with("dcnt"))


majic_2014 <- bind_rows (majic_2014_com44, majic_2014_com49, majic_2014_com53, majic_2014_com72, majic_2014_com85) %>%
  left_join (com2017, by = c ("idcom" = "depcom")) %>%
  select (-idcom, -idcomtxt) %>%
  group_by (epci_2017, depcom2017) %>%
  summarise_all (funs (sum)) %>%
  ungroup %>%
  mutate (artif_2014=dcnt07+dcnt09+dcnt10+dcnt11+dcnt12+dcnt13) %>%
```

```
  select(-starts_with("dcnt"))


# 1.2 on passe également les données de population sur la nouvelle carte des territoires

p_2009 <- population_2009 %>%
  left_join (com2017, by = c ("idcom" = "depcom")) %>%
  select (-idcom) %>%
  group_by (epci_2017, depcom2017) %>%
  summarise(population_2009=sum(Population)) %>%
  ungroup

p_2014 <-population_2014 %>%
  left_join (com2017, by = c ("idcom" = "depcom")) %>%
  select (-idcom) %>%
  group_by (epci_2017, depcom2017) %>%
  summarise(population_2014=sum(Population)) %>%
  ungroup

# 2.1 indicateur à la commune
# on joint les 4 tables précédentes par commune et on calcul les indicateurs
etalement_urbain_commune <- majic_2009 %>%
  left_join(majic_2014) %>%
  left_join (p_2009) %>%
  left_join (p_2014) %>%
  mutate (evoarti = 100 * artif_2014 / artif_2009 - 100,
          evopop = 100 * population_2014 / population_2009 - 100,
          indicateur_etalement_simple=evoarti/evopop,
          indicateur_etalement_avance = case_when (
            evoarti < 0 & evopop >= 0 ~ "1",
            evoarti >= 0 & evopop >= 0 & (evoarti / evopop <= 1 | evopop==0) ~ "2a",
            evoarti < 0 & evopop < 0 & evoarti / evopop > 1 ~ "2b",
            evopop < 0 & evoarti / evopop >= 0 & evoarti / evopop <= 1 ~ "2c",
            evopop > 0 & evoarti > 0 & evoarti <= 4.9 & evoarti / evopop > 1 ~ "3",
            evopop > 0 & evoarti> 4.9 & evoarti / evopop > 1 & evoarti / evopop <= 2 ~ "4",
            evopop > 0 & evoarti> 4.9 & evoarti / evopop > 2 ~ "5",
            evopop < 0 & evoarti / evopop < 0 ~ "6"
          )
  )

# 2.2 indicateur à l'epci
# on joint les 4 tables précédentes par commune, on aggrege les compteurs par epci et on calcul les indicateurs

etalement_urbain_epci <- majic_2009 %>%
  left_join(majic_2014) %>%
  left_join (p_2009) %>%
  left_join (p_2014) %>%
  select(-depcom2017) %>%
  group_by(epci_2017) %>%
  summarise_all(funs(sum(.))) %>%
  mutate (evoarti = 100 * artif_2014 / artif_2009 - 100,
          evopop = 100 * population_2014 / population_2009 - 100,
          indicateur_etalement_simple=evoarti/evopop,
          indicateur_etalement_avance = case_when (
            evoarti < 0 & evopop >= 0 ~ "1",
            evoarti >= 0 & evopop >= 0 & (evoarti / evopop <= 1 | evopop==0) ~ "2a",
            evoarti < 0 & evopop < 0 & evoarti / evopop > 1 ~ "2b",
            evopop < 0 & evoarti / evopop >= 0 & evoarti / evopop <= 1 ~ "2c",
            evopop > 0 & evoarti > 0 & evoarti <= 4.9 & evoarti / evopop > 1 ~ "3",
            evopop > 0 & evoarti> 4.9 & evoarti / evopop > 1 & evoarti / evopop <= 2 ~ "4",
            evopop > 0 & evoarti> 4.9 & evoarti / evopop > 2 ~ "5",
            evopop < 0 & evoarti / evopop < 0 ~ "6"
          )
  )

# Deux graphiques de visualisation de notre indicateur
ggplot(data=etalement_urbain_epci) +
  geom_point(aes(x=evoarti,y=evopop,color=indicateur_etalement_avance))+
  theme_minimal() +
  labs(title="Indicateur d'étalement urbain sur les epci de la région Pays de la Loire",x="Evolution de l'artificialisation",y="Evolution de la
démographie",color="",caption="Source : Majic et Recensement de la population\nCarte des territoires 2017")

ggplot(data=etalement_urbain_commune) +
  geom_point(aes(x=evoarti,y=evopop,color=indicateur_etalement_avance),size=.5,alpha=.5)+
  theme_minimal()+
  labs(title="Indicateur d'étalement urbain sur les communes de la région Pays de la Loire",subtitle="Entre 2009 et 2014",x="Evolution de
l'artificialisation",y="Evolution de la démographie",color="",caption="Source : Majic et Recensement de la population\nCarte des territoires 2017")
```