```
#---------------------------------------------------------------------
# Chapitre 3 Présentation de l'interface et premières manipulations
#---------------------------------------------------------------------

# 3.2 Session et répertoire de travail

setwd()
help()
dir()

# 3.3 Exercice
setwd("")
?setwd
help(setwd)
dir()

setwd("D:/WKSP/5_GROUPES/180227_G2R/FORMATION_R/git_g2r/parcours-r/m1_socle_book")

# 3.4 Prise en main de la console

# 3.5 Création d'une variable
ma_variable <- 2
ma_variable <- "Toulouse"
ma_variable <- c ("Toulouse", "Nantes", "Strasbourg")
ma_variable <- 1:10

ls()
paste()
rm()


# 3.6 Exercice
a<-5
b<-4
ls()

a+b
c<-a+b*3

paste(a,b) #!!erreur dans exo

A<-15
B<-12
rm()
?ls
?rm
rm (list = ls ())

# 3.7 Utilité des scripts
print()

# 3.8 Exercice : utiliser un script
# Supprimer toutes les variables existantes
rm (list = ls())

# Création de mes variables taille et poids ; attention au point décimal
poids <- 91
taille <- 1.87

# Calcul de l'IMC : poids sur taille au carré
imc <- poids/(taille^2)

# Affichage du résultat
print (imc)

#---------------------------------------
# Chapitre 4 Mon premier jeu de données
#---------------------------------------

install.packages ("tidyverse")
library ("tidyverse")
base <- read.csv ("Data/Base_synth_territoires.csv",
                  header = T, sep=";", dec=",")
nrow (base)
ncol (base)
dim (base)

# 4.5 Gérer le type des variables
base <- mutate  (base, LIBGEO = as.character (LIBGEO))

# 4.6 Les valeurs particulières
V1 <- c (1 , 14 , NA , 32.7)
mean (V1)              # renvoie NA. Not good !
mean (V1, na.rm = T)   # renvoie 15.9   - OK !
```

```
# 4.7 Exercice : Importer les données et premier coup d'oeil
#!! reprendre base : déjà chargé
df <- read.csv (file = "Data/Base_synth_territoires.csv",
                sep = ";", dec = ",",
                colClasses = c (NA, "NULL", "character", "NULL","NULL","NULL",
                                NA, NA, NA, NA, rep ("NULL", 28)))

str (df)
head (df)
tail (df)
names (df)
typeof (df) # le type d'objet du point de vue "interne" à R
class (df$CODGEO)
typeof (df$CODGEO)
length (df)
length (df$REG)

#-------------------------------------
# 5 Première manipulation des données
#-------------------------------------

# 5.1 Afficher les valeurs et manipuler les variables
library (tidyverse)
base <- read.csv ("Data/Base_synth_territoires.csv",
                  header = T, sep=";", dec=",")
str (pull (base, DEP))

TableEnSortie <- mutate (TableEnEntree,
                         NouvelleVariable = DefinitionDeLaVariable)

base <- mutate (base, log_SUPERF = log (SUPERF))

base <- mutate (base, log_SUPERF = 100 * log_SUPERF)

# 5.2 Sélectionner des variables
TableEnSortie <- select (TableEnEntree, Variable1, Variable2, ..., VariableN)
TableEnSortie <- select (TableEnEntree, -Variable1, -Variable2, ..., -VariableN)
base_select <- select (base, CODGEO, LIBGEO, P14_POP)
base_select <- select (base, -CODGEO)

# 5.3 Sélectionner des observations
TableEnSortie <- filter (TableEnSortie, Condition1, ..., ConditionN)
base_filter <- filter (base, DEP == "01" & P14_POP > 10000)

# 5.4 Les tests logiques dans R

# 5.5 Renommer des colonnes

# 5.6 Exercice : Créer, filtrer, sélectionner
df <- mutate (base, densite = P14_POP / SUPERF,
              tx_natal = 1000 * NAISD15 / P14_POP,
              tx_mort = DECESD15 / P14_POP)

selection <- select (df, CODGEO, ZAU, REG, DEP,
                     densite, tx_natal, tx_mort)

S0 <- filter (selection, DEP == "62")

S1 <- filter (selection, DEP != "62") # tout sauf le 62 :(
S2 <- filter (selection, DEP %in% c ("59","62")) # L'ancien NPdC :)
S3 <- filter (selection, !(DEP %in% c ("59","62"))) # Le "sud" de la France
S4 <- filter (selection, densite > 100) # l'urbain
S5 <- filter (selection, DEP=="62" & densite > 100) # le PdC urbain
S6 <- filter (selection, DEP=="62" | densite > 100) # le PdC et l'urbain

selection_62 <- select (df, CODGEO, ZAU, REG, DEP, densite, tx_natal) %>%
  filter(DEP=="62")

selection_62 <- df [df$DEP == "62",
                    c ("CODGEO", "ZAU", "REG", "DEP", "densite",
                       "tx_natal")]

# ------------------------------------------
# Chapitre 6 Premiers traitements statistiques
#------------------------------------------

# 6.1 La fonction summary
library (tidyverse)
base <- read.csv ("Data/Base_synth_territoires.csv",
                  header = T, sep=";", dec=",")

base_extrait <- select (base, 1, 3, 5, 7:12)

summary (base_extrait)
summary (pull (base_extrait, NAIS0914))
```

```
summary (pull (base_extrait, ZAU))

# 6.2 Calculer des statistiques spécifiques
sum (pull (base_extrait, P14_POP), na.rm = T)
mean (pull (base_extrait, P14_POP), na.rm = T)
median (pull (base_extrait, P14_POP), na.rm = T)
quantile (pull (base_extrait, P14_POP), probs = c (.25, .5, .75), na.rm = T)

# 6.3 Agréger des données selon un facteur
TableauGroupes <- group_by (TableEnEntree, Variable1, ..., VariableN)
summarise (TableauGroupes, NomVariableAgregee = Fonction (NomVariableEtude))

base_reg_ann <- group_by (base_extrait, ZAU, REG) %>%
  summarise (population_med = median (P14_POP, na.rm = T))

# 6.4 Une autre manière de sélectionner une variable : $
Dataframe$Variable
pull (base_extrait, ZAU) # méthode "tidyverse"
base_extrait$ZAU # méthode "base"

# 6.5 Tableau de contingence
t <- table (base_extrait$ZAU, base_extrait$REG)
print (t)

# 6.6 Tableau de proportions
round (100 * prop.table (t), digits = 1)
print (chisq.test (t))

# 6.7 Exercice : calcul de statistiques
df <- base %>%
  select (1:24) %>%
  mutate (densite = P14_POP / SUPERF,
          tx_natal = 1000 * NAISD15 / P14_POP,
          tx_mort = DECESD15 / P14_POP,
          ZAU2 = as.factor (substr (ZAU, 1, 3))) # Parce que la variable originale est trop longue et caractères
bizarres

summary (df)

mean (df$densite)
sd (df$densite)
median (df$densite)
var (df$densite)

mean (df$densite, na.rm = T)
sd (df$densite, na.rm = T)
median (df$densite, na.rm = T)
var (df$densite, na.rm = T)

df <- mutate (df, std_dens = (densite - mean (densite, na.rm = T)) / sd (densite, na.rm = T))
mean (df$std_dens, na.rm = T)

sd (df$std_dens, na.rm = T)

quantile (df$densite, na.rm = T)

t <- table(df$ZAU2)

100 * prop.table(t) %>% round(digits = 4)

t <- table (df$REG, df$ZAU2)

100 * prop.table (t) %>% round (digits = 4)

#--------------------------------
# Chapitre 7 Premiers graphiques
#--------------------------------

# 7.1 Package ggplot2
install.packages ("ggplot2")
library ("ggplot2")

ggplot (TableEnEntree, aes (VariablesATracer)) + geom_FonctionAChoisir ()

# 7.2 Histogramme
rm (list = ls ())

base <- read.csv ("Data/Base_synth_territoires.csv",
                  header = T, sep=";", dec=",") %>%
  select (1:24) %>%
  mutate (log_SUPERF = log (SUPERF),
          REG = as.factor (REG),
          densite = P14_POP / SUPERF,
          tx_natal = 1000 * NAISD15 / P14_POP,
          tx_mort = DECESD15 / P14_POP)
```

```
ggplot (base, aes (x = log_SUPERF)) + geom_histogram ()

# 7.3 Nuages de points
ggplot (base, aes (x=P14_EMPLT, y=MED13)) + geom_point (colour = "blue")
ggplot (base, aes (x=log (P14_EMPLT), y=log (MED13))) + geom_point (colour = "blue")

# 7.4 Matrice de nuages
# install.packages ("GGally")
library ("GGally") # pour des graphiques plus jolis
num <- select (base, P14_LOGVAC:PIMP13) %>% sample_n (10000) %>% log ()
ggpairs (num) ## fonction ggpairs() de GGally

# 7.5 Bonus : faire un graphique "dynamique"
require (plotly)
g <- ggplot (data = base, aes (x = REG, fill = REG)) +
  geom_bar ()

ggplotly (g)

# 7.6 Exercices : créer des graphiques
ggplot (data = base, aes (x = P14_POP)) +
  geom_histogram()

ggplot (data = base, aes (x = log (P14_POP))) +
  geom_histogram()

ggplot (data = base, aes (x = REG)) +
  geom_bar()

ggplot (data = base, aes( x = REG, fill = REG)) +
  geom_bar()

ggplot (data = base, aes (x = log (P14_POP), y = log (P14_LOGVAC))) +
  geom_point()

ggplot (data = base, aes( x = densite, y = tx_mort, color = REG)) +
  geom_point()

plot (iris)

#-----------------------------------
# Chapitre 8 Sauvegarder son travail
#-----------------------------------

# 8.1 Exportation des résultats

res <- summary (base)
write.table (base, file = 'resultat_R.csv', sep = ';', row.names = F)
# row.names=F pour éviter un décalage entre première ligne et les suivantes

write.table (base, file = 'base_R.csv', sep = ';', row.names = F)

png ('mongraphe.png') # Alloue et ouvre le fichier où inscrire le graphe
ggplot (base, aes (x = P14_EMPLT, y = MED13)) +
  geom_point (colour = "blue")
dev.off ()

# 8.2 Environnement et RData

save (list = ls(), file = "env_entier.RData") # sauvegarde de tout l'environnement sur l'ordinateur
rm (list = ls()) # suppression de notre environnement dans R
load ("env_entier.RData") # chargement de l'environnement stocké sur l'ordinateur

save (base, V1, file = "petit_env.RData") # sauvegarde des éléments base et V1
rm (list = ls()) # suppression de notre environnement
load ("petit_env.RData")

#-------------------------------------------------------------------------------------
# Chapitre 9 Aller plus loin avec les objets, crochets et la programmation fonctionnelle
#-------------------------------------------------------------------------------------

# 9.1 Les objets dans R
# 9.1.1 Créer des vecteurs
v1 <- seq(1 : 10)
v2 <- c ("lundi", "mardi", "mercredi", "jeudi",
         "vendredi", "samedi", "dimanche")

# 9.1.2 Créer une matrice
m <- matrix (v1, nrow = 10, ncol = 7)
l <- list (v1, v2, m)

# 9.1.3 Visualiser ces objets et leurs attributs
v1
print (v2)
```

```
typeof (v2) # Permet de visualiser le type
dim (m)
str (l)    # Permet de visualiser les attributs

# 9.2 Sélectionner des lignes et des colonnes
base[1,3] # valeur de la première ligne et de la troisième colonne
base[2,] # toutes les variables pour la 2e observation
base[,4] # toutes les observations de la quatrième colonne
base[,"V6"] # toutes les observations de la variable V6

# 9.3 Créer une nouvelle fonction en R
calcul_IMC <- function (poids, taille)
{
  ## La taille est exprimée en mètres
  imc <- poids / taille ^ 2
  return (imc)
}
calcul_IMC (poids=80,taille=1.89)

calcul_IMC (poids=60,taille=1.55)

# 9.4 Les boucles conditionnelles

diag_IMC <- function(poids,taille)
{
  imc <- poids / taille ^ 2
  if (imc < 18.5) {diag <- "maigre"}
  else if (imc < 25) {diag <- "normal"}
  else {diag <- "surpoids"}
  return (diag)
}
diag_IMC (poids=60,taille=1.89)

diag_IMC (poids=80,taille=1.89)

diag_IMC (poids=80,taille=1.55)

# 9.5 Les boucles
for (pp in seq(from = 50, to = 100, by = 5))
{
  print(paste ("Taille = 1,70m, poids =", pp, "Diagnotic :",
               diag_IMC (poids = pp, taille = 1.70)))
}

# 9.6 Exercices
# 9.6.1 Vecteurs simples
rm (list = ls ())

x <- c (1, 160, 2, 9, 60)
x1 <- c("Je", "programme", "en", "R") # Guillemets pour indiquer que c'est une variable textuelle
y <- seq (from = 1, to = 10, by = 1)
z <- rep (x = 1, times = 100)
x <- rnorm (n = 30)

# création de vecteurs avec la fonction c() = combine
v1 <- c( 3, 4, 12, 15, 32, 6, 1, 2, 3, 9)

# avec la fonction seq() = sequence, g?n?ralisation de la syntaxe ci-dessus
v2 <- seq(from = 1 , to = 15 , by = 1.5)

# syntaxe équivalente mais préférable car plus lisible :

v2b <- seq (from=1, to=15, by=1.5)
v3 <- 1:10

# avec la fonction rep() = répétition
v4 <- rep (x = 4, times = 10)

# ces commandes peuvent être combinées. Pratique pour cr?er des variables "facteur"
v5 <- rep (x = c(3, 4, 1.2, 8, 9), times = 2)
v6 <- rep (x = 1:5, times = 2)

# vecteurs de type texte ou factor
vtaille <- rep (x = c ("S", "L"), times = 5)
vtaille <- factor (vtaille)

# concaténation de vecteurs
gtaille <- paste("X", vtaille, sep = "")
gtaille <- factor (gtaille)
toutes_taille <- c (as.character (vtaille), as.character (gtaille))
toutes_taille <- as.factor (toutes_taille)
levels (toutes_taille)

# 9.6.2 Dataframes et listes
```

```
dataf <- data.frame (vtaille, v1, v2, v3, v4, v5, v6)
liste <- list (vtaille, v1, v2, v3, v4, v5, v6)
names(liste) <- c ("vtaille", "v1", "v2", "v3", "v4", "v5", "v6")

dataf$vtaille

liste$vtaille

rm (dataf, vtaille, v1, v2, v2b, v3, v4, v5, v6)

# 9.6.3 Pour aller plus loin : matrices et arrays

mat <- matrix(rnorm(50), ncol = 5, nrow = 10)
arr <- array(rnorm(150),dim = c(10,5,3))
mat
arr

apply(mat, MARGIN = 1, FUN=mean)

apply(mat, MARGIN = 2, FUN=mean)

apply (arr, MARGIN = 3, FUN=mean)

apply (arr, MARGIN = c(2,3), FUN = mean)

mat [1,1]
mat [1,]
mat [,1]

arr [1,1,1]
arr [1,,]
arr [,,1]

str (liste [1])
str (liste [[1]])

# 9.6.4 Inspection d'un objet : la régression

data ("iris")
str (iris)

lm (data = iris, formula = Sepal.Length ~ Petal.Length)

reg <- lm(data = iris, formula = Sepal.Length ~ Petal.Length)
str (reg)

summary (reg)

plot (reg)
```