Introduction à la programmation R - avec corrigé en fin de document

Marie Laure Delignette-Muller et Karine Chalvet-Monfray

2024-02-22

Contents

Préambule

Nous sommes conscientes que vous abordez ces TD de biostatistique du deuxième semestre avec des niveaux très hétérogènes en programmation et c'est pourquoi nous les encadrons les premiers à deux enseignantes. Certains d'entre vous ont eu de la programmation dans leur cursus, certains ont même déjà utilisé le langage R, et d'autres n'ont jamais fait de programmation. Afin que vous puissiez tous avancer efficacement durant les séances de biostatistique qui vont suivre, nous vous demandons une forte coopération, car même à deux nous risquons d'avoir du mal à aider tout le monde de façon efficace. Nous demandons donc en particulier à ceux qui sont le plus à l'aise en programmation de se répartir dans la salle de façon à pouvoir donner un coup de main aux autres, avec bienveillance (sans moqueries). Il ne s'agit pas de faire à leur place, mais de répondre à leurs questions et de leur donner un coup de pouce pour démarrer. En parallèle nous demandons à ceux qui sont moins à l'aise au départ d'être très actifs / volontaires en TD, en faisant bien les choses eux-mêmes (vous allez tous y arriver, quel que soit votre point de départ, et vous allez peut-être même trouver cela rigolo au final), sans rester bloqués, mais en posant des questions à vos voisins et aux enseignantes quand vous êtes bloqués (sans avoir pas peur de poser des questions "idiotes" - toutes sont légitimes si vous vous les poser, et oser les poser peut servir aussi aux autres). Et nous demandons à tous de terminer ce TD en travail personnel (exceptées les parties notées OPTIONNEL) si vous ne l'avez pas terminé en séance, en vous servant du corrigé qui sera fourni sur vetAgro Tice si besoin.

Concepts de base

A FAIRE en début, en cours et en fin de session Rstudio

- En début de session on **ouvre un rapport d'analyse** (fichier .rmd) et si c'est un nouveau fichier on **le sauve d'emblée avec l'extension .Rmd ou .rmd** dans le même dossier que celui où sont les fichiers de données. *Pour ce premier TD vous allez ouvrir le fichier .rmd correspondant à cet énoncé (TD_progR_S4.rmd que vous aurez mis dans un dossier avec les fichiers de données, après décompression de ceux-ci), faire tourner ligne à ligne les chunks qui y sont fournis, et compléter les chunks des exercices.* Un chunk dans un fichier .rmd ressemble à cela:

```
```{r}
# commentaire
ligne de code R
ligne de code R
ligne de code R

# Commentaire
ligne de code R
ligne de code R
```
```

- On **change le répertoire de travail** (Onglet Session, Set Working Directory) et on le définit au répertoire où se trouve le fichier .rmd et les données (option **To Source File location**).
- On **sauve le fichier très régulièrement** !!! D'ailleurs lorsque l'on "tricote" (bouton knit) le .rmd pour produire le rapport d'analyse, il sauve le .rmd automatiquement.
- En fin de session **Rstudio vous demande si vous voulez sauver l'espace de travail** (workspace). Il est fortement conseillé de **toujours refuser** pour éviter des problèmes (sauf cas très exceptionnels qui ne devraient pas vous concerner de suite).

A SAVOIR avant de commencer :

- R est **sensible à la casse** (majuscule / minuscule)
- Dans un code R , tout ce qui se trouve à **droite du signe #** est considéré comme un **commentaire**, donc non interprété comme du code (mais bien utile pour comprendre son code).

- Le **séparateur de décimale** dans R est **le point .** et non la virgule.
- Le **symbole <-** est utilisé pour **affecter une valeur à un objet**. Mieux vaut ne pas le remplacer par le symbole = même si souvent ça fonctionne (mais pas toujours !). Le symbole = est à réserver pour la définition des valeurs des arguments dans l'appel à une fonction et le symbole == pour tester une égalité dans une condition logique.
- Nommez les objets R avec des noms parlants (pas trop courts donc) mais pas trop longs non plus.
- Evitez de donner des noms qui sont susceptibles de correspondre à des noms de fonctions R (ex. : `median`)
- Ne commencez pas le nom d'un objet par un chiffre.
- **Evitez les accents, les espaces et les caractères spéciaux** dans les **noms des objets R** ainsi que dans les **fichiers de données** que vous allez importer dans R (donc y penser bien en amont dès la saisie des données dans un tableur par exemple). Les seuls caractères spéciaux conseillés dans les noms d'objets, tant qu'ils ne sont pas en début de nom, sont _ et . (ex. : `animal_prefere`).
- Dans les données, **codez les variables quantitatives par des nombres** (donc pas J1, J3, J10 par ex.) et les **variables qualitatives par des chaînes de caractères** (donc pas 1 pour les mâles et 0 pour les femelles par ex.). Cela vous évitera une étape de recodage dans R.
- Les **données manquantes** doivent être **codées** `NA` (pas de cellule vide dans un tableau de données que l'on souhaite importer dans R).

Création d'un objet R simple

> **UTILISATION du signe d'affectation <-, des fonctions `class()`, `rm()`,`is.numeric()`, `is.character()`, `is.logical()` et `as.numeric()`, `as.character()`, `as.logical()` et des opérateurs mathématiques et logiques (ET : `&`, OU : `|`, EGAL: ==, DIFFERENT DE `!=`, NON : `!`, SUPERIEUR >, SUPERIEUR ou EGAL>=, etc.),**

Pour vous approprier les bases, exécutez chaque ligne de code une à une en prenant le temps d'examiner ce qui est produit et de comprendre le code.

Pour la plupart des exercices, nous vous indiquons le nom des fonctions à utiliser. Ayez le réflexe à chaque fois d'aller voir l'aide en ligne des fonctions que vous ne connaissez encore pas bien en tapant `?nom_de_la_fonction`.

```
# --- les nombres (`numeric`) ---
age <- 45 # affectation d'une valeur à un objet R
# vous pouvez mettre la valeur de votre choix pour cet âge
age # pour voir l'objet
```

```
## [1] 45
```

```
class(age) # pour connaître la classe de l'objet
```

```
## [1] "numeric"
```

```
rm(age) # pour effacer un objet de l'espace de travail (rm pour remove)
# regarder l'effet de cette commande dans onglet Environment en haut à droite de Rstudio

(age <- 45) # pour faire l'affectation et voir l'objet en même temps
```

```
## [1] 45
```

```r
(age_en_mois <- age * 12)
```

```
## [1] 540
```

```r
(age_au_carre <- age^2)
```

```
## [1] 2025
```

```r
# --- les chaînes de caractères (`character`) ---
genre <- "masculin"
pays <- "Autriche"
class(genre)
```

```
## [1] "character"
```

```r
# --- les booléens (`logical`) ---
(jeune <- age < 30)
```

```
## [1] FALSE
```

```r
class(jeune)
```

```
## [1] "logical"
```

```r
# utilisation des opérateurs logiques
(homme_jeune <- (age < 30) & (genre == "masculin"))
```

```
## [1] FALSE
```

```r
# autres façons d'écrire la même chose
(homme_jeune <- (age < 30) & (genre != "feminin"))
```

```
## [1] FALSE
```

```r
(homme_jeune <- !(age >= 30) & (genre == "masculin"))
```

```
## [1] FALSE
```

```r
(homme_jeune <- !((age >= 30) | (genre == "feminin")))
```

```
## [1] FALSE
```

```r
# Vérification de la classe et changement de classe d'un objet
# les fonctions commençant pas is. font un test logique
# les fonctions commençant pas as. font un changement de classe
is.numeric(age)
```

```
## [1] TRUE
```

```r
is.numeric(jeune)
```

```
## [1] FALSE
```

```r
(jeune_num <- as.numeric(jeune))
```

```
## [1] 0
```

```r
is.numeric(jeune_num)
```

```
## [1] TRUE
```

EXERCICES de création d'un objet R simple :

1. Créez un booléen indiquant si le répondant est autrichien d'au moins 18 ans et vérifier sa valeur bien sûr.
2. Créez un booléen indiquant si le répondant a moins de 16 ans ou plus de 70 ans.
3. Calculez l'âge du répondant en log en utilisant la fonction `log()`. Celle-ci donne-t-elle le logarithme népérien ou décimal de l'âge ? Trouvez à l'aide sur la fonction `log()` (en tapant `?log`) comment on obtient l'autre log.

Chunk à compléter

```r
## 1

## 2

## 3
```

Création d'un vecteur ou d'une matrice

UTILISATION du calcul vectorisé et des fonctions `c()`, `rep()`, `seq()`, `matrix()`, `rbind()`, `cbind()`. `rownames()`, `colnames()`, `is.vector()`, `is.matrix()`, `length()`, `dim()`, `nrow()`, `ncol()`, `table()`, `mean()` et `median()` avec l'argument `na.rm`.

Un vecteur est un objet R utilisé notamment pour stocker toutes les valeurs d'une variable en statistique. Il est à une dimension et ne contient que des éléments de même type. En statistique il permettra par exemple de stocker les valeurs d'une variable quantitative (vecteur de nombres), ou d'une variable qualitative (vecteur de chaînes de caractères). Une matrice, c'est la même chose en dimension 2, toujours ne contenant que des éléments de même type.

On peut créer un vecteur, par exemple,

- avec la fonction `c()` pour concaténer plusieurs valeurs,
- avec la fonction `rep()`pour répéter plusieurs fois une même valeur,
- avec la fonction `seq()` pour générer une séquence régulière de valeurs,
- ou par calcul à partir d'autres vecteurs créés auparavant. Le calcul est en effet vectorisé par défaut dans R, ce qui veut dire que si l'on écrit un calcul impliquant un ou plusieurs vecteurs, il fait automatiquement le calcul pour tous les éléments du (ou des) vecteur(s) impliqué(s), élément par élément (*i.e.* ligne à ligne).

Pour bien comprendre l'usage de ces fonctionnalités, exécutez chaque ligne de code une à une en prenant le temps de comprendre le code, d'examiner ce qui est produit jusqu'à pouvoir l'anticiper.

```r
# Création de vecteurs avec c()
(Age <- c(71, 29, 45, NA, 81)) # pour rappel NA signifie donnée manquante
```

```
## [1] 71 29 45 NA 81
```

```r
is.vector(Age)
```

```
## [1] TRUE
```

```r
class(Age) # donne le type du vecteur
```

```
## [1] "numeric"
```

```r
length(Age) # donne la longueur du vecteur
```

```
## [1] 5
```

```r
Pays <- c("Autriche", "Danemark", "Danemark", "UK", "Autriche")
class(Pays)
```

```
## [1] "character"
```

```r
Genre <- c("feminin", "feminin", "masculin", "masculin", "feminin")

# Création d'un vecteur avec rep()
(Annee <- rep(2023, times = 5))
```

```
## [1] 2023 2023 2023 2023 2023
```

```r
# Création de séquences régulières
(Numero <- 1:5)
```

```
## [1] 1 2 3 4 5
```

```r
(Numero <- seq(from = 1, to = 5, by = 1))
```

```
## [1] 1 2 3 4 5
```

```r
(Numero10en10 <- seq(from = 10, to = 60, by = 10))
```

```
## [1] 10 20 30 40 50 60
```

```r
# Ajout de valeurs
(Agecomplete <- c(Age, 14, 17))
```

```
## [1] 71 29 45 NA 81 14 17
```

```r
class(Agecomplete)
```

```
## [1] "numeric"
```

```r
length(Agecomplete)
```

```
## [1] 7
```

```r
# Que se passe-t-il si on concatène des valeurs qui ne sont pas du même type ?
(Agecomplete2 <- c(Age, "mineur", "mineur"))
```

```
## [1] "71"     "29"     "45"     NA       "81"     "mineur" "mineur"
```

```r
class(Agecomplete2)
```

```
## [1] "character"
```

```r
# Calcul vectorisé
Age^2
```

```
## [1] 5041  841 2025   NA 6561
```

```r
Annee - Age
```

```
## [1] 1952 1994 1978   NA 1942
```

```r
(Majeur <- Agecomplete >= 18)
```

```
## [1]  TRUE  TRUE  TRUE    NA  TRUE FALSE FALSE
```

```r
class(Majeur)
```

```
## [1] "logical"
```

```r
# Application de quelques fonctions statistiques à un vecteur
mean(Age)
```

```
## [1] NA
```

```r
mean(Age, na.rm = TRUE)
```

```
## [1] 56.5
```

```r
median(Age, na.rm = TRUE)
```

```
## [1] 58
```

```r
table(Pays)
```

```
## Pays
## Autriche Danemark       UK
##        2        2        1
```

EXERCICES de création d'un vecteur (en vérifiant le résultat à chaque fois bien sûr) :

1. Créez un vecteur de type booléen (`logical`) indiquant si le répondant est autrichien ou du Royaume Uni et a au moins 50 ans.
2. Créez un vecteur de type numérique codant avec un 1 les femmes et un 0 les garçons en procédant en deux étapes : en utilisant le calcul vectorisé avec les opérateurs logiques pour obtenir une variable logique qui sera à `TRUE` pour les femmes, puis en utilisant la fonction `as.numeric()` sur cette variable logique (les `TRUE` seront transformés en 1 et les `FALSE` en 0).

Chunk à compléter

```r
## 1

## 2
```

Pour créer une matrice on peut par exemple utiliser la fonction `matrix()` ou les fonctions `rbind()` et `cbind()` qui permettent de juxtaposer des vecteurs de même type respectivement par ligne ou colonne. Voici quelques exemples à explorer par vous mêmes.

```r
# Création d'une matrice par remplissage par ligne
(Matrice1 <- matrix(c(1, 2, 3, 4, 11, 12, 13, 14), byrow = TRUE, ncol = 4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]   11   12   13   14
```

```r
class(Matrice1)
```

```
## [1] "matrix" "array"
```

```r
is.matrix(Matrice1)
```

```
## [1] TRUE
```

```r
is.vector(Matrice1)
```

```
## [1] FALSE
```

```r
dim(Matrice1)
```

```
## [1] 2 4
```

```r
nrow(Matrice1)
```

```
## [1] 2
```

```r
ncol(Matrice1)
```

```
## [1] 4
```

```r
# en changeant le nombre de colonnes
(Matrice2 <- matrix(c(1, 2, 3, 4, 11, 12, 13, 14), byrow = TRUE, ncol = 2))
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]   11   12
## [4,]   13   14
```

```r
# Création d'une matrice par remplissage par colonne
(Matrice3 <- matrix(c(1, 2, 3, 4, 11, 12, 13, 14), byrow = FALSE, ncol = 2))
```

```
##      [,1] [,2]
## [1,]    1   11
## [2,]    2   12
## [3,]    3   13
## [4,]    4   14
```

```r
# Création de noms de colonnes
colnames(Matrice3) <-
c("i", "10+i")
Matrice3
```

```
##      i 10+i
## [1,] 1   11
## [2,] 2   12
## [3,] 3   13
## [4,] 4   14
```

EXERCICES de manipulation de matrices :

1. Créez une nouvelle matrice de la même dimension que `Matrice3` ne contenant que des valeurs à 100.
2. Créez une nouvelle matrice comme la somme de `Matrice3` et de celle que vous venez de créer.
3. Utilisez la fonction `colnames()` pour changer les noms de colonnes de cette matrice "100+i" et "110+i".

Chunk à compléter

```
## 1

## 2

## 3
```

Indexation d'un vecteur ou d'une matrice

On peut indexer un vecteur (ou une matrice) par **position**, en indiquant le numéro de ligne (et de colonne pour une matrice) entre crochets (séparés par une virgule dans le cas d'une matrice), par **nom** (si les lignes - resp. les colonnes - sont nommées), ou par **condition logique**. Pour bien comprendre comment cela fonctionne explorez les codes suivants.

```
# Indexation par position d'un vecteur
Age # juste pour avoir en tête toutes les valeurs
```

```
## [1] 71 29 45 NA 81
```

```
Age[1]
```

```
## [1] 71
```

```
Age[2:4]
```

```
## [1] 29 45 NA
```

```
Age[length(Age)]
```

```
## [1] 81
```

```
Age[-4] # pour enlever la 4ème valeur
```

```
## [1] 71 29 45 81
```

```
Age[c(-1, -3)] # pour enlever la 1ère et la 3ème valeur
```

```
## [1] 29 NA 81
```

```
## Indexation par condition d'un vecteur
Age[Age > 30]
```

```
## [1] 71 45 NA 81
```

```
Age[Genre == "masculin"]
```

```
## [1] 45 NA
```

Pour une matrice on indique toujours entre les crochets ce qui correspond aux lignes, une virgule, puis ce qui correspond aux colonnes. La virgule doit être mise même si on ne fait une sélection que sur les lignes ou que sur les colonnes. Explorez les exemples ci-dessous.

```r
# Indexation par position d'une matrice
Matrice3 # pour avoir en tête la matrice en entier
```

```
##      i 10+i
## [1,] 1   11
## [2,] 2   12
## [3,] 3   13
## [4,] 4   14
```

```r
Matrice3[1, 2] # toujours [numéro de ligne, numéro de colonne]
```

```
## 10+i
##   11
```

```r
Matrice3[2, 1] # toujours [numéro de ligne, numéro de colonne]
```

```
## i
## 2
```

```r
Matrice3[1:3, ]
```

```
##      i 10+i
## [1,] 1   11
## [2,] 2   12
## [3,] 3   13
```

```r
Matrice3[2, ]
```

```
##    i 10+i
##    2   12
```

```r
Matrice3[, 2]
```

```
## [1] 11 12 13 14
```

```r
## Indexation par nom
Matrice3[, "10+i"]
```

```
## [1] 11 12 13 14
```

```r
## Indexation par nom et calcul d'indice
Matrice3[2+2, "10+i"]
```

```
## 10+i
##   14
```

EXERCICES d'indexation

1. Ecrire une ligne de code qui donne l'âge du deuxième individu de la variable `Age`.
2. Ecrire une ligne de code qui donne la valeur contenue dans `Matrice1` en première ligne et avant dernière colonne (en utilisant la fonction `ncol()` pour calculer le nombre de colonnes de la matrice dans l'indexation).
3. Ecrire une ligne de code qui donne le vecteur des genres des individus qui viennent du Danemark.
4. *En OPTIONNEL pour les plus rapides, créez un objet que vous nommerez pays_du_plus_jeune qui contiendra le pays du plus jeune individu (parmi ceux d'âge connu). Pour cela vous pouvez procéder en plusieurs instructions, en utilisant la fonction `min()` avec l'argument `na.rm` pour ignorer les données manquantes, et une condition logique pour l'indexation dans le vecteur `Pays`.*

Chunk à compléter

```
## 1

## 2

## 3
```

Les jeux de données

Création d'un jeu de données directement dans R

UTILISATION des fonctions `data.frame()`, `str()`, `is.data.frame()`, `as.data.frame()`

Un jeu de données est un **objet R de dimension 2**, comme une matrice, mais à la différence d'une matrice il ne contient **pas forcément des éléments tous du même type**. Il est composé de vecteurs colonnes qui peuvent être de types différents. Un jeu de données est une juxtaposition de vecteurs dans lesquels sont stockées en colonne les valeurs de différentes variables (qualitatives ou quantitatives), chaque ligne correspondant aux valeurs des ces variables pour un même individu, ou plus généralement une même observation. Ci-dessous nous allons créer un petit jeu de données "jouet" à partir des vecteur créés précédemment et utiliser quelques fonctions. Explorez ce code.

```
# Création d'un jeu de données à partir des vecteurs définis en amont
(d_jouet <- data.frame(age = Age, genre = Genre, pays = Pays))
```

```
##   age    genre     pays
## 1  71  feminin Autriche
## 2  29  feminin Danemark
## 3  45 masculin Danemark
## 4  NA masculin       UK
## 5  81  feminin Autriche
```

```
class(d_jouet)
```

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

```
str(d_jouet) # structure du jeu de données
```

```
## 'data.frame':    5 obs. of  3 variables:
##  $ age  : num  71 29 45 NA 81
##  $ genre: chr  "feminin" "feminin" "masculin" "masculin" ...
##  $ pays : chr  "Autriche" "Danemark" "Danemark" "UK" ...
```

```r
is.data.frame(d_jouet)
```

```
## [1] TRUE
```

```r
# Création d'un jeu de données à partir d'une matrice
# dans ce cas le jeu de données ne contiendra soit que des variables quantitatives
# soit que des variables qualitatives
is.data.frame(Matrice3)
```

```
## [1] FALSE
```

```r
(d_matrice3 <- as.data.frame(Matrice3))
```

```
##   i 10+i
## 1 1   11
## 2 2   12
## 3 3   13
## 4 4   14
```

```r
str(d_matrice3)
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ i   : num  1 2 3 4
##  $ 10+i: num  11 12 13 14
```

Ajout de colonnes calculées à partir des colonnes préexistantes

Il est assez fréquent, quand on manipule des données, que l'on souhaite ajouter des variables calculées à partir des colonnes de base (*i.e.* des données dites brutes), et il vaut bien mieux le faire dans R, que dans le tableur (pour des raisons de traçabilité notamment). Pour créer une nouvelle variable dans le jeu de données, il suffit de lui donner comme nom dans l'affection le nom du jeu de données suivi de $ suivi du nom que l'on veut donner à la cette variable, comme ci-dessous :

```r
d_jouet$age_log_10 <- log10(d_jouet$age)
d_jouet$au_moins_50ans <- d_jouet$age >= 50
d_jouet
```

```
##   age    genre     pays age_log_10 au_moins_50ans
## 1  71  feminin Autriche   1.851258          TRUE
## 2  29  feminin Danemark   1.462398         FALSE
## 3  45 masculin Danemark   1.653213         FALSE
## 4  NA masculin       UK         NA            NA
## 5  81  feminin Autriche   1.908485          TRUE
```

EXERCICES d'ajout de colonnes à un jeu de données :

1. Ajouter une colonne à `d_jouet` nommée `age_mois` contenant l'âge en mois des individus, calculé à partir de leur âge en année codé dans `d_jouet$age`.

2. Ajouter une colonne àd_jouet nommée autrichienne qui aura pour valeur TRUE si l'individu est une femme autrichienne.

Chunk à compléter

```
## 1

## 2

## Visualisation de d_jouet pour vérifier
d_jouet
```

```
##   age    genre     pays age_log_10 au_moins_50ans
## 1  71  feminin Autriche   1.851258           TRUE
## 2  29  feminin Danemark   1.462398          FALSE
## 3  45 masculin Danemark   1.653213          FALSE
## 4  NA masculin       UK         NA             NA
## 5  81  feminin Autriche   1.908485           TRUE
```

Indexation d'un jeu de données

La façon la plus courante de sélectionner une colonne (donc une variable) d'un jeu de données est d'utiliser le nom du jeu de données et d'y accoler $ suivi du nom de la variable, comme ci-dessous, mais on peut aussi l'indexer de la même façon qu'une matrice pour sélectionner ligne(s) ou colonne(s). Testez les lignes de code ci-dessous pour bien comprendre.

```
# Sélection d'une colonne (variable)
d_jouet$age
```

```
## [1] 71 29 45 NA 81
```

```
d_jouet$pays
```

```
## [1] "Autriche" "Danemark" "Danemark" "UK"       "Autriche"
```

```
d_jouet[, 2]
```

```
## [1] "feminin"  "feminin"  "masculin" "masculin" "feminin"
```

```
# Sélection de lignes
d_jouet[2:4, ]
```

```
##   age    genre     pays age_log_10 au_moins_50ans
## 2  29  feminin Danemark   1.462398          FALSE
## 3  45 masculin Danemark   1.653213          FALSE
## 4  NA masculin       UK         NA             NA
```

```r
d_jouet[d_jouet$age > 30, ]
```

```
##    age    genre     pays age_log_10 au_moins_50ans
## 1   71  feminin Autriche   1.851258           TRUE
## 3   45 masculin Danemark   1.653213          FALSE
## NA  NA     <NA>     <NA>         NA             NA
## 5   81  feminin Autriche   1.908485           TRUE
```

```r
# Sélection de colonnes par noms
d_jouet[, c("age", "pays")]
```

```
##   age     pays
## 1  71 Autriche
## 2  29 Danemark
## 3  45 Danemark
## 4  NA       UK
## 5  81 Autriche
```

```r
# Sélection sur lignes et colonnes
d_jouet[d_jouet$age > 30, ]$pays
```

```
## [1] "Autriche" "Danemark" NA         "Autriche"
```

Les facteurs

UTILISATION des fonctions `is.factor()`, `as.factor()`, `factor()`, `levels()`

Pour que R considère les vecteurs de chaînes de caractères (type `char`) comme des variables qualitatives (type `factor`), il faut les tranformer en facteurs. Ensuite il est souvent utile de **modifier l'ordre des modalités d'un facteur** (par défaut il les ordonne par ordre alphabétique) et/ou de **changer le nom des modalités**. Découvrez ces fonctionalités en testant le code ci-dessous.

```r
# Transformation en facteur
class(d_jouet$pays)
```

```
## [1] "character"
```

```r
d_jouet$pays <- as.factor(d_jouet$pays)
class(d_jouet$pays)
```

```
## [1] "factor"
```

```r
levels(d_jouet$pays)
```

```
## [1] "Autriche" "Danemark" "UK"
```

```r
# Changement des noms des modalités
levels(d_jouet$pays) <- c("Autriche", "Danemark", "Royaume-Uni")

# Changement de l'ordre des modalités
d_jouet$pays <- factor(d_jouet$pays, levels = c("Danemark", "Royaume-Uni", "Autriche"))
levels(d_jouet$pays)
```

```
## [1] "Danemark"    "Royaume-Uni" "Autriche"
```

EXERCICES de manipulation des facteurs :

1. Transformez `d_jouet$genre` en facteur et appliquez la fonction `table()` à celui-ci.
2. Utilisez la fonction `levels()` à `d_jouet$genre` pour voir les noms et ordre des modalités du genre
3. Modifiez les noms des modalités du genre en "femme" et "homme".
4. Modifiez l'ordre des modalités du genre en mettant "homme" en première modalité et réappliquez la fonction `table()` à `d_jouet$genre` pour vérifier que vous n'avez pas fait de bêtise.

Chunk à compléter

Insérez vous-même ici le chunk (dans l'onglet supérieur de Rstudio Code / Insert chunk) et complétez-le.

Importation et exploration rapide d'un jeu de données

UTILISATION des fonctions `read.table()`, `str()`, `head()`, `nrow()`, `dim()`, `summary()`.

Généralement les jeux de données ne sont pas créés directement dans R mais importés par exemple à partir d'un fichier texte exporté depuis un tableur. Avant d'importer un fichier contenant le jeu de données à importer, il est important de savoir :

- **s'il comporte ou non une en-tête** (première ligne qui correspond aux noms des variables, pour définir l'**argument `header`** de la fonction d'import),
- quel **séparateur de colonnes** a été utilisé (le point virgule `";"`, l'espace `" "`, la tabulation `"\t"`, ..., pour définir l'**argument `sep`** de la fonction d'import),
- quel **séparateur de décimales** a été utilisé, pour définir l'**argument `dec`** de la fonction d'import. (ATTENTION, dans R le point `.` est utilisé, alors que sur la plupart des ordinateurs paramétrés en France, la virgule `", "` est utilisée. Il faudra donc indiquer `dec = ","` pour importer correctement un fichier venant d'un tel ordinateur paramétré en français pour le séparateur de décimales).

Pour importer des données à partir d'un fichier .txt exporté classiquement avec comme **séparateur de colonnes la tabulation**, nous utiliserons la fonction `read.table()` avec ses arguments `header = TRUE` si le fichier comporte une en-tête, et `stringsAsFactors = TRUE` pour que les vecteurs de chaînes de caractères soient automatiquement transformés en facteurs. Dans la plupart des jeux de données mis à disposition sur VetAgroTice le séparateur de décimales est déjà un point, donc il n'est pas nécessaire de préciser l'argument `dec`.

Importez et explorez un jeu de données qui nous servira plus tard à l'aide des lignes de codes suivantes :

```r
d <- read.table("ENQ2223.txt", header = TRUE, stringsAsFactors = TRUE)
str(d)
```

```
## 'data.frame':    149 obs. of  19 variables:
##  $ ident           : Factor w/ 149 levels "Anonyme1","Anonyme10",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ civilite        : Factor w/ 2 levels "Madame","Monsieur": 2 1 1 1 1 1 1 2 1 2 ...
##  $ cursus          : Factor w/ 3 levels "A1","Autre","BCPST": 1 1 3 3 3 3 3 3 3 3 ...
##  $ tps_travail     : int  2 10 0 5 8 8 15 35 7 0 ...
##  $ nb_CM           : int  0 10 0 10 20 20 10 17 8 0 ...
##  $ apprentissage   : Factor w/ 3 levels "intermediaire",..: 2 2 2 1 1 3 1 3 2 2 ...
##  $ tps_job         : num  0 0 5 0 0 0 0 0 0 3 ...
##  $ tps_loisirs     : int  25 10 14 6 13 7 6 6 5 20 ...
##  $ nb_rattrapages  : int  0 0 0 0 0 0 1 0 2 0 ...
##  $ niveau_global   : int  13 11 15 10 7 13 14 15 5 15 ...
##  $ memoire         : int  16 13 20 15 15 15 14 13 8 18 ...
##  $ orthographe     : int  18 9 19 17 18 16 18 13 19 19 ...
##  $ biostat_proba   : int  7 1 5 12 5 16 17 18 2 16 ...
##  $ biostat_info    : int  9 9 5 8 10 16 17 19 10 17 ...
##  $ reorientation   : Factor w/ 2 levels "non","oui": 1 1 1 1 1 1 1 2 2 1 ...
##  $ bien_etre       : int  17 15 20 10 19 10 10 15 1 19 ...
##  $ serenite_examens: int  15 11 19 8 15 13 8 8 3 18 ...
##  $ interet_cours   : int  13 10 18 12 15 14 12 14 15 15 ...
##  $ filiere         : Factor w/ 5 levels "AC","autre","equine",..: 1 5 4 4 4 1 4 1 5 1 ...
```

```
head(d)
```

```
##        ident civilite cursus tps_travail nb_CM apprentissage tps_job
## 1   Anonyme1 Monsieur     A1           2     0          tard       0
## 2  Anonyme10   Madame     A1          10    10          tard       0
## 3 Anonyme100   Madame  BCPST           0     0          tard       5
## 4 Anonyme101   Madame  BCPST           5    10 intermediaire       0
## 5 Anonyme102   Madame  BCPST           8    20 intermediaire       0
## 6 Anonyme103   Madame  BCPST           8    20           tot       0
##   tps_loisirs nb_rattrapages niveau_global memoire orthographe biostat_proba
## 1          25              0            13      16          18             7
## 2          10              0            11      13           9             1
## 3          14              0            15      20          19             5
## 4           6              0            10      15          17            12
## 5          13              0             7      15          18             5
## 6           7              0            13      15          16            16
##   biostat_info reorientation bien_etre serenite_examens interet_cours filiere
## 1            9           non        17               15            13      AC
## 2            9           non        15               11            10  rurale
## 3            5           non        20               19            18   mixte
## 4            8           non        10                8            12   mixte
## 5           10           non        19               15            15   mixte
## 6           16           non        10               13            14      AC
```

```
nrow(d)
```

```
## [1] 149
```

```
dim(d)
```

```
## [1] 149  19
```

```r
summary(d)
```

```
##         ident          civilite     cursus    tps_travail        nb_CM
##  Anonyme1  :  1   Madame  :117   A1   :33   Min.   : 0.00   Min.   : 0.00
##  Anonyme10 :  1   Monsieur: 32   Autre:34   1st Qu.: 6.00   1st Qu.: 0.00
##  Anonyme100:  1                  BCPST:82   Median :10.00   Median :10.00
##  Anonyme101:  1                             Mean   :12.17   Mean   :10.58
##  Anonyme102:  1                             3rd Qu.:16.00   3rd Qu.:20.00
##  Anonyme103:  1                             Max.   :60.00   Max.   :35.00
##  (Other)   :143
##        apprentissage    tps_job        tps_loisirs     nb_rattrapages
##  intermediaire:89    Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000
##  tard         :27    1st Qu.: 0.000   1st Qu.: 5.000   1st Qu.: 0.0000
##  tot          :33    Median : 0.000   Median : 7.000   Median : 0.0000
##                      Mean   : 1.983   Mean   : 8.825   Mean   : 0.7181
##                      3rd Qu.: 4.000   3rd Qu.:10.000   3rd Qu.: 1.0000
##                      Max.   :20.000   Max.   :40.000   Max.   :15.0000
##
##  niveau_global     memoire       orthographe    biostat_proba    biostat_info
##  Min.   : 0.0   Min.   : 0.00   Min.   : 2.00   Min.   : 0.00   Min.   : 0.00
##  1st Qu.:10.0   1st Qu.:10.00   1st Qu.:15.00   1st Qu.: 8.00   1st Qu.: 8.00
##  Median :12.0   Median :14.00   Median :18.00   Median :12.00   Median :11.00
##  Mean   :11.8   Mean   :12.73   Mean   :16.13   Mean   :10.82   Mean   :11.13
##  3rd Qu.:14.0   3rd Qu.:15.00   3rd Qu.:19.00   3rd Qu.:14.00   3rd Qu.:15.00
##  Max.   :18.0   Max.   :20.00   Max.   :20.00   Max.   :20.00   Max.   :20.00
##
##  reorientation   bien_etre    serenite_examens interet_cours     filiere
##  non:133       Min.   : 0.0   Min.   : 0.00    Min.   : 5.00   AC    :24
##  oui: 16       1st Qu.:10.0   1st Qu.: 9.00    1st Qu.:13.00   autre :11
##                Median :14.0   Median :12.00    Median :15.00   equine:13
##                Mean   :13.1   Mean   :11.38    Mean   :14.62   mixte :85
##                3rd Qu.:17.0   3rd Qu.:15.00    3rd Qu.:17.00   rurale:16
##                Max.   :20.0   Max.   :19.00    Max.   :20.00
##
```

Automatisation des calculs

Ecriture d'une fonction et application d'une fonction par groupe à l'aide de la fonction `tapply()`

Afin de comprendre comment on écrit une fonction et comment on peut l'appliquer par groupe à l'aide de `tapply()` testez le code suivant :

```r
# Création d'une fonction
Q1 <- function(x){
  quartiles <- quantile(x, probs = c(0.25, 0.5, 0.75), na.rm = TRUE)
  return(as.vector(quartiles))
}

# Application de cette fonction au temps de loisirs
# (codé dans la variable `tps_loisirs`) de tous les répondants
```

```r
# puis manuellement par genre (codé dans la variable  `civilite`)

Q1(d$tps_loisirs)
```

```
## [1]  5  7 10
```

```r
Q1(d$tps_loisirs[d$civilite == "Madame"])
```

```
## [1]  5  6 10
```

```r
Q1(d$tps_loisirs[d$civilite == "Monsieur"])
```

```
## [1]  6.75 10.00 13.50
```

```r
# Application automatique de la fonction par genre
tapply(d$tps_loisirs, d$civilite, Q1)
```

```
## $Madame
## [1]  5  6 10
##
## $Monsieur
## [1]  6.75 10.00 13.50
```

EXERCICES d'utilisation de la fonction tapply :

1. A l'aide de la fonction `tapply()`, calculez la médiane de l'estimation par les étudiants de leur niveau global par genre.
2. A l'aide de la fonction `tapply()`, calculez la médiane de l'estimation par les étudiants de leur sérénité aux examens par cursus d'origine.
3. *OPTIONNEL pour les plus avancés : faites une petite fonction qui affiche les quartiles, le min et le max, en utilisant les fonctions `quantile()`, `min()` et `max()` et appliquez cette fonction pour calculer ces statistiques sur le score de bien_etre par cursus d'origine en utilisant la fonction `tapply()`.*

Chunk à compléter

Insérez vous-même ici le chunk (dans l'onglet supérieur de Rstudio Code / Insert chunk) et complétez-le.

Structures conditionnelles `if` et itératives `for` - OPTIONNEL, pour les plus à l'aise en programmation

Afin de comprendre comment on utilise les structures conditionnelles et itératives, testez le code suivant.

```r
# Utilisation de la boucle for
for (i in 1:5){
  print(paste("Itération", i, sep = " ")) # paste() colle ensemble des chaînes de caractères
}
```

```
## [1] "Itération 1"
## [1] "Itération 2"
## [1] "Itération 3"
## [1] "Itération 4"
## [1] "Itération 5"
```

```r
# Utilisation d'une condition if
valeur <- 40
if (valeur > 50){
  print("OK")
} else {
  print("KO")
}
```

```
## [1] "KO"
```

```r
# Boucle contenant une condition
valeurs <- c(12, 40, 45, 52, 67, 78)
for (i in 1:length(valeurs)){
  if (valeurs[i] > 50){
    print("OK")
  } else {
    print("KO")
  }
}
```

```
## [1] "KO"
## [1] "KO"
## [1] "KO"
## [1] "OK"
## [1] "OK"
## [1] "OK"
```

Sauvegarde d'un jeu de données

A l'aide de la fonction `write.table()`, sauvegardez le jeu de données `d_jouet`, avec toutes les nouvelles variables que vous avez créées, dans un fichier que l'on nommera `Exemple_jouet.txt` (à l'aide du code suivant) et importez-le dans Excel pour l'y retrouver.

```r
# Petit regard sur la structure du jeu de données augmenté avant sa sauvegarde
str(d_jouet)
```

```
## 'data.frame':    5 obs. of  5 variables:
##  $ age           : num  71 29 45 NA 81
##  $ genre         : chr  "feminin" "feminin" "masculin" "masculin" ...
##  $ pays          : Factor w/ 3 levels "Danemark","Royaume-Uni",..: 3 1 1 2 3
##  $ age_log_10    : num  1.85 1.46 1.65 NA 1.91
##  $ au_moins_50ans: logi  TRUE FALSE FALSE NA TRUE
```

```r
# Sauvegarde du jeu de données en format texte
write.table(d_jouet, file = "Exemple_jouet.txt", row.names = FALSE)
```

Si vous souhaitez avoir la virgule comme séparateur de décimales (important ici uniquement pour la variable AgeCR), il faut mettre l'argument `dec = ","`, mais dans ce cas il faudra bien penser à faire de même si vous voulez importer à nouveau ce jeu de données dans R avec la fonction `read.table()`.

Quelques liens vers des documents utiles

Pour l'utilisation de R

- la "cheat sheet" de la base de R : https://iqss.github.io/dss-workshops/R/Rintro/base-r-cheat-sheet.pdf
- la carte de référence de R en français : https://www.apmep.fr/IMG/pdf/R_RefCard.pdf
- la carte de référence de R (version 2 plus moderne) en anglais : https://cran.r-project.org/doc/contrib/Baggott-refcard-v2.pdf
- Le "cookbook" de R : http://www.cookbook-r.com/

Pour l'utilisation de rmarkdown pour inclure le code R (et les sorties) dans un rapport d'analyse

- le "cookbook" de rmarkdown : https://bookdown.org/yihui/rmarkdown-cookbook/
- la "cheat sheet" de rmardown : https://rstudio.github.io/cheatsheets/rmarkdown.pdf