Séance 2.2: Transformation des données

Visseho Adjiwanou, PhD.

30 May 2021

Plan

1. Créer manuellement une base de données
2. Télécharger des données des médias sociaux
3. Travailler avec des données

Effaçons notre environnement

Ce code permet d'effacer l'environnement.

```
rm(list = ls())
```

Entrons les données pour créer la base de données

- la section de code est obtenue par ce raccourci:
  - command + Option + i (mac)
- Pour exécuter une ligne de code, il faut se placer sur la ligne et faire
  - command + Entree (mac)

```
nom_classe <- c("guillaume", "amy", "andree", "john", "natacha")
nom_classe
```

```
## [1] "guillaume" "amy"       "andree"    "john"      "natacha"
```

```
sexe_classe <- c("h", "f", "f", "h", "f")
sexe_classe
```

```
## [1] "h" "f" "f" "h" "f"
```

```
sexe_classe1 <- c(1, 0, 0, 1, 0)
sexe_classe1
```

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

```
revenu_classe <- c(24000, 15000, 22000, 17000, 21000)
revenu_classe
```

```
## [1] 24000 15000 22000 17000 21000
```

```
age_classe <- c(19, 25, 30, 27, 32)
age_classe
```

```
## [1] 19 25 30 27 32
```

```
sexe_classe_factor <- factor(c(1, 0, 0, 1, 0), labels = c("Femme", "Homme"))
sexe_classe_factor
```

```
## [1] Homme Femme Femme Homme Femme
## Levels: Femme Homme
```

```
education_classe <- c("universite", "college", "secondaire", "universite", "college")
education_classe
```

```
## [1] "universite" "college"    "secondaire" "universite" "college"
```

Petite parenthèse sur les fonctions

Dans la création de la variable `sexe_classe_factor`, on a utilisé une fonction qui s'appelle **factor**. Si vous tapez **?factor** dans le chunk (c'est quoi un chunk?), il vous indiquera comment utiliser cette fonction.

```
?factor
```

On voit ainsi dans RStudio, dans la fenêtre **Help**, la description de cette fonction et son usage. Certains de ces arguments (ce qui se trouve dans la parenthèse) sont optionnelles, alors que d'autres sont obligatoires. La fonction factor permet de créer une variable **qualitative** ou **categorielle** ou **factorielle**, à partir d'une autre variable. Il faut donc lui indiquer cette variable à partir de laquelle on créer la nouvelle variable factorielle. Dans notre cas (chunk ligne 47, il s'agit de la variable c(1, 0, 0, 1, 0). De manière optionnelle, il faut lui indiquer à quoi représentent les valeurs de l'ancienne variable. ce;a se fait avec **labels**. Finalement, il faut indiquer l'ordre dans lequel les valeurs dela nouvelle variables vont être affichées. Si on ne lui indique rien, il va les afficher par ordre alphabétique. Autrement, il faut lui indiquer l'ordre que vous souhaitez, avec l'option **levels**. Vous avez plusieurs autres arguments que vous pouvez aller consulter vous-mêmes. Finalement, la description se termine toujours par des exemples d'utilisation.

Base de données

Une base de données n'est qu'une mise ensemble de plusieurs vecteurs ou variables.

```
base_classe <- data.frame(nom = nom_classe, age = age_classe, sexe = sexe_classe, sexe01 = sexe_classe1

base_classe
```

```
##         nom age sexe sexe01 sexefactor revenu  education
## 1 guillaume  19    h      1      Homme  24000 universite
## 2       amy  25    f      0      Femme  15000    college
## 3    andree  30    f      0      Femme  22000 secondaire
## 4      john  27    h      1      Homme  17000 universite
## 5   natacha  32    f      0      Femme  21000    college
```

Pour créer cette base de données, j'ai utilisé la fonction **data.frame**. cette fonction comme la fonction factor viennent de base R. Cela signifie qu'elles sont incorporées directement dans votre R, une fois que vous télécharger R pour la première fois. Par contre, plusieurs autres fonctions qui ne figurent pas dans la base R, vous devez télécharger leur package avant de les utiliser. C'est ce que nous avons fait avec le package **tidyverse**. Et la fonction de tidyverse pour créer une base de données est data_frame. Avez-vous remarqué là où se trouve la différence. Dans bien des cas, quand une fonction se trouve dans la base R et dans une autre package, la différence va être celle là (_ au lieu de .) ou sur la capitalisation de la première lettre (Factor au lieu de factor). Chaque fois que vous tapez une fonction, prenez la peine d'observer de quel packahe provient la fonction.

Comme dans une matrice

On peut travailler dans une base de données comme dans une matrice.

```
base_classe[1, "nom"]
```

```
## [1] guillaume
## Levels: amy andree guillaume john natacha
```

```
base_classe[, "nom"]
```

```
## [1] guillaume amy       andree    john      natacha
## Levels: amy andree guillaume john natacha
```

```
base_classe["nom"]
```

```
##         nom
## 1 guillaume
## 2       amy
## 3    andree
## 4      john
## 5   natacha
```

```
base_classe["age2"] <- base_classe["age"]*2
```

Quelle est la classe des éléments de cette base de données?

```
class(base_classe$nom)        # factor
```

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

```
class(base_classe$sexe)       # factor
```

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

```
class(base_classe$sexe01)         # numeric
```

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

```
class(base_classe$sexefactor)     # factor
```

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

```
class(base_classe$revenu)         # numeric
```

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

```
class(base_classe$age)            # numeric
```

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

```
class(base_classe$education)      # numeric
```

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

Maintenant, on va travailler avec une base de donnée plus simple.

- Il s'agit du CROP Socio-Cultural Survey de 1996
- Dans cette partie, nous allons apprendre à :

- Sélectionner les variables
- Sélectionnez les observations
- Réorganiser les données
- Créer de nouvelles variables avec des fonctions de variables existantes (mutate ())
- Recoder des variables existantes
- Calculer des statistiques univariées

Maintenant, on va travailler avec une base de donnée plus simple.

Nous allons travaillé avec les données issues des enquêtes socioculturelles. Les enquêtes socioculturelles font partie d'une grande série d'études internationales comparatives sur les valeurs fondamentales. Des enquêtes parallèles sont effectuées chaque année dans plusieurs pays européens et aux États-Unis. CROP Inc. a commencé cette série au Canada en 1983. Ces enquêtes portent sur un large éventail d'attitudes fondamentales - sociales, culturelles, économiques et politiques. Ces données sont disponibles sur le site suivant: https://www.queensu.ca/cora/our-data/data-holdings

Dressons la table

Avant de travailler avec ces données, il faut charger les packages que nous allons utiliser. Ici, nous ferons recours à `tidyverse` et à `summarytools`. Chargeons les deux packages. Si vous travailler sur vos machines, vous devez télécharger en premier lieu ces packaqes avec la commande : `install.packages("tidyverse")` et `install.packages("summarytools"). Cependant, un package s'installe une seule fois. Mais chaque fois que vous voulez les utilisez, vous devez juste les charger. On peut faire l'analogie avec Facebook. Pour pouvoir utiliser`facebook‘ sur votre téléphone, vous devez en premier lieu télécharger cette application. Pour les prochaines utilisations, vous n'avez pas besoin de le télécharger à nouveau. Vous devez juste l'ouvrir. C'est le même principe avec les packaqes de R.

Donc ici, dans la partie code, j'ai mis un hashtag au début des commentaires pour signifier que je ne veux pas les exécuter.

```
# Effacer votre environnement

rm(list = ls())

# Installer les package dont vous avez besoin

#install.packages("tidyverse")
#install.packages("summarytools")
#install.packages("tinytex")

# Charger les packages - Étape fondamentales

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.4.0
## Warning: package 'ggplot2' was built under R version 3.6.2
## Warning: package 'tibble' was built under R version 3.6.2
## Warning: package 'tidyr' was built under R version 3.6.2
```

```
## Warning: package 'readr' was built under R version 3.6.2
```

```
## Warning: package 'purrr' was built under R version 3.6.2
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(summarytools)
```

```
## Registered S3 method overwritten by 'pryr':
##   method      from
##   print.bytes Rcpp
```

```
## Warning in system2("/usr/bin/otool", c("-L", shQuote(DSO)), stdout = TRUE):
## running command ''/usr/bin/otool' -L '/Library/Frameworks/R.framework/Resources/
## library/tcltk/libs//tcltk.so'' had status 1
```

```
##
## Attaching package: 'summarytools'
```

```
## The following object is masked from 'package:tibble':
##
##     view
```

Une fois que les packages de tidyverse sont téléchargés, ils vous les montrent. Il vous indique les endroits où il y a conflit avec d'autres packages.

Téléchargement de la base de données

Maintenant, il faut charger les données que vous voulez utiliser dans R. Ici, on fait recours à la fonction `read_csv` qui est issue du package `tidyverse`. Remarquer l'usage de `_`. La même fonction existe dans base R avec un `.` (read.csv). cette fonction lit les fichiers csv. Si vous avez d'autres types de fichiers, vous devez utiliser d'autres fonctions. On verra certains plus tard dans le cours.

```
crsc96 <- read_csv("../Données/cora-crsc1996-E-1996_F1.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use `spec()` for the full column specifications.
```

Regardons ce que contient cette base de données

```
# trois manière de faire

#View(crsc96)

head(crsc96)
```

```
## # A tibble: 6 x 416
##    sexq  ageq commsize region   age    q1    q2    q3    q4    q5    q6    q7
##   <dbl> <dbl>    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
## 1     2     3        1      9    33     1     5     5     5     5     5     5
## 2     2     3        1      9    34     2     5     4     5     1     5     5
## 3     2     4        1      9    56     2     2     4     5     5     5     5
## 4     1     5        1      9    69     1     4     2     4     2     4     4
## 5     1     3        1      9    43     4     4     4     5     2     5     5
## 6     2     2        1      9    28     4     5     4     5     1     5     5
## # ... with 404 more variables: q8 <dbl>, q9 <dbl>, q10 <dbl>, q11 <dbl>,
## #   q12 <dbl>, q13 <dbl>, q14 <dbl>, q15 <dbl>, q16 <dbl>, q17 <dbl>,
## #   q18 <dbl>, q19 <dbl>, q20 <dbl>, q21 <dbl>, q22 <dbl>, q23 <dbl>,
## #   q24 <dbl>, q25 <dbl>, q26 <dbl>, q27 <dbl>, q28 <dbl>, q29 <dbl>,
## #   q30 <dbl>, q31 <dbl>, q32 <dbl>, q33 <dbl>, q34 <dbl>, q35 <dbl>,
## #   q36 <dbl>, q37 <dbl>, q38 <dbl>, q39 <dbl>, q40 <dbl>, q41 <dbl>,
## #   q42 <dbl>, q43 <dbl>, q44 <dbl>, q45 <dbl>, q46 <dbl>, q47 <dbl>,
## #   q48 <dbl>, q49 <dbl>, q50 <dbl>, q51 <dbl>, q52 <dbl>, q53 <dbl>,
## #   q54 <dbl>, q55 <dbl>, q56 <dbl>, q57 <dbl>, q58 <dbl>, q59 <dbl>,
## #   q60 <dbl>, q61 <dbl>, q62 <dbl>, q63 <dbl>, q64 <dbl>, q65 <dbl>,
## #   q66 <dbl>, q67 <dbl>, q68 <dbl>, q69 <dbl>, q70 <dbl>, q71 <dbl>,
## #   q72 <dbl>, q73 <dbl>, q74 <dbl>, q75 <dbl>, q76 <dbl>, q77 <dbl>,
## #   q78 <dbl>, q79 <dbl>, q80 <dbl>, q81 <dbl>, q82 <dbl>, q83 <dbl>,
## #   q84 <dbl>, q85 <dbl>, q86 <dbl>, q87 <dbl>, q88 <dbl>, q89 <dbl>,
## #   q90 <dbl>, q91 <dbl>, q92 <dbl>, q93 <dbl>, q94 <dbl>, q95 <dbl>,
## #   q96 <dbl>, q97 <dbl>, q98 <dbl>, q99 <dbl>, q100 <dbl>, q101 <dbl>,
## #   q102 <dbl>, q103 <dbl>, q104 <dbl>, q105 <dbl>, q106 <dbl>, q107 <dbl>, ...
```

```
#glimpse(crsc96)
```

Taille échantillon : 2859 Nombre de variables : 416

Travailler avec les données.

Nous utiliserons surtout ici le package **dplyr** pour manipuler les données: - sélectionner les variables avec `select` - sélectionner les individus avec `filter` - déterminer le type de variable avec `class` - recoder et créer de nouvelles variables avec `mutate`

Sélectionnons les données qui nous intéressent

q1 : - I hate being bossed around: I must feel that I have total control over all the different areas of my life - **Je déteste être patronisé: je dois sentir que j'ai un contrôle total sur tous les différents domaines de ma vie**

q2: - An unmarried girl of 18 should not have sexual relations - *Une fille non mariée de 18 ans ne devrait pas avoir de relations sexuelles*

q3: - The best way to get something from someone is by putting your foot down - *La meilleure façon d'obtenir quelque chose de quelqu'un est de mettre le pied à terre (dialoguer)*

q4: - In a household where both partners are working, is not right for the wife to earn more than the husband - *Dans un ménage où les deux partenaires travaillent, il n'est pas normal que la femme gagne plus que le mari*

q44: - Overpopulation in third world countries doesn't really affect our country - *La surpopulation dans les pays du tiers monde n'affecte pas vraiment notre pays*

q95: - An extramarital affair from time to time is not that serious - *Une liaison extraconjugale de temps en temps n'est pas si grave*

q96: - I would like to have a religious service at my funeral - *J'aimerais avoir un service religieux à mes funérailles*

```
crsc96_small <-
  crsc96 %>%
  select(sexq, region, age, ageq, q1, q2, q3, q4, q44, q95, q96)

crsc96_small
```

```
## # A tibble: 2,859 x 11
##     sexq region   age  ageq    q1    q2    q3    q4   q44   q95   q96
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     2      9    33     3     1     5     5     5     4     5     1
##  2     2      9    34     3     2     5     4     5     5     5     5
##  3     2      9    56     4     2     2     4     5     5     4     5
##  4     1      9    69     5     1     4     2     4     5     5     2
##  5     1      9    43     3     4     4     4     5     5     4     4
##  6     2      9    28     2     4     5     4     5     5     5     2
##  7     1      9    27     2     2     4     2     4     4     5     4
##  8     1      9    51     4     1     4     4     5     5     4     2
##  9     1      9    41     3     1     5     5     5     4     4     5
## 10     1      9    39     3     4     2     5     5     4     5     1
## # ... with 2,849 more rows
```

Signification du code:

- La première ligne du code indique le nom du nouveau fichier
- La deuxième ligne indique le fichier qui va être utilisé. Cette ligne est terminée par le symbole %>% qu'on appelle **pipe**. Ce symbole n'a aucune signification statistique ou mathématique. Il indique juste une succession d'opération.
- la dernière ligne donne les indications sur la manière dont vous voulez créer le fichier de la **première ligne**.

Sélectionnons les données les données qui nous intéressent

- Select

```
crsc96_small <-
  crsc96 %>%
  select(sexq, region, age, ageq, q1, q2, q3, q4, q44, q95)

crsc96_small
```

```
## # A tibble: 2,859 x 10
##     sexq region   age  ageq    q1    q2    q3    q4   q44   q95
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     2      9    33     3     1     5     5     5     4     5
##  2     2      9    34     3     2     5     4     5     5     5
##  3     2      9    56     4     2     2     4     5     5     4
##  4     1      9    69     5     1     4     2     4     5     5
##  5     1      9    43     3     4     4     4     5     5     4
##  6     2      9    28     2     4     5     4     5     5     5
##  7     1      9    27     2     2     4     2     4     4     5
##  8     1      9    51     4     1     4     4     5     5     4
##  9     1      9    41     3     1     5     5     5     4     4
## 10     1      9    39     3     4     2     5     5     4     5
```

```
## # ... with 2,849 more rows
```

- Première ligne: Ceci signifie que je crée un nouvel objet (ici une base de données) que je nomme `crsc96_small`
- Deuxième ligne: Je le crée à partir de la base de donnée que je viens de charger `crsc96`. Le signe `%>%` s'appelle `pipes` et signifie que j'applique ce qui viens après le symbole à ce qui est à sa droite.
- Troisième ligne: les variables que je veux sélectionner.

Le code est très bien éclairé avec des espaces aux bons endroits avant les symboles `<-` et `%>%`. Remarquer aussi l'indentation après `<-` sur la nouvelle ligne.

Un bon codage permet de vous faire lire facilement. Adopter une bonne manière dès maintenant.

Sélectionner les observations

```
crsc96_small_homme <-
  crsc96_small %>%
  filter(sexq == 1 & age >= 35)

crsc96_small_homme
```

```
## # A tibble: 804 x 10
##     sexq region   age  ageq    q1    q2    q3    q4   q44   q95
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     1      9    69     5     1     4     2     4     5     5
##  2     1      9    43     3     4     4     4     5     5     4
##  3     1      9    51     4     1     4     4     5     5     4
##  4     1      9    41     3     1     5     5     5     4     4
##  5     1      9    39     3     4     2     5     5     4     5
##  6     1      9    86     5     5     2     5     5     5     5
##  7     1      9    37     3     2     5     4     5     4     5
##  8     1      9    43     3     1     5     1     5     5     5
##  9     1      9    67     5     2     4     4     5     5     5
## 10     1      9    46     4     2     4     2     5     4     5
## # ... with 794 more rows
```

Toutes ces étapes peuvent se réduire à:

On n'est pas obliger de séparer ces deux étapes. On peut les combiner ensemble.

```
crsc96_small_homme_general <-
  crsc96 %>%
  select(sexq, region, age, ageq, q1, q2, q3, q4, q44, q95) %>%
  filter(sexq == 1 & age >= 35)
```

Subset dans base R vous permet de faire les deux choses en méme temps

Voici enfin une autre manière de faire en utilisant la fonction subset de base R.

```
crsc_base <-
crsc96 %>%
  subset(sexq == 1 & age >= 35, select = c(sexq, region, age, ageq, q1, q4, q44, q95))
```

Information sur une variable

Maintenant, nous allons travailler avec la base de données `crsc_small`. Nous introduisons ici un nouvel symbole `$`. `$` fait référence à une colonne (une variable) spécifique dans une base de données. C'est la même chose que nous avons appris précédemment avec les matrices en utilisant [,].

```
crsc96_small$sexq
```

```
##    [1] 2 2 2 1 1 2 1 1 1 1 1 2 2 2 2 2 1 2 1 1 2 2 2 1 1 2 2 2 2 1 1 1 1 1 1 2 2
##   [38] 1 1 1 1 2 2 2 1 1 2 2 2 2 1 2 1 1 1 2 2 1 1 2 2 1 1 1 2 1 2 2 2 2 1 1 2 1
##   [75] 2 1 2 1 1 2 2 2 1 2 2 1 2 1 2 1 1 2 2 2 1 2 1 2 1 2 1 2 2 1 1 1 1 1 1 2 2
##  [112] 2 1 1 2 2 2 2 2 2 1 1 1 1 2 2 2 1 1 1 1 2 1 2 1 1 1 1 2 1 2 1 1 2 1 1 2 2
##  [149] 1 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 1 1 2 1 2 2 2 1 1 2 1 1 2 2 2 1 1 2
##  [186] 2 2 1 1 1 2 1 1 2 2 2 2 1 2 2 1 1 1 1 1 1 1 2 2 1 2 2 1 2 2 2 2 2 1 1 2 2
##  [223] 2 1 2 1 1 2 2 1 2 2 1 1 2 1 1 1 1 1 2 2 1 2 1 2 2 2 2 2 2 2 1 1 2 1 2 2 1
##  [260] 2 1 2 1 2 2 2 1 2 1 2 2 1 1 1 2 2 1 1 1 2 2 2 2 1 1 2 2 1 1 1 1 2 2 1 1 2
##  [297] 2 1 2 1 1 2 1 2 2 1 2 2 2 1 1 1 2 1 2 1 2 2 2 2 2 1 2 1 1 1 1 2 2 1 1 1 2
##  [334] 2 1 2 2 1 1 2 2 2 2 2 2 1 1 2 1 2 1 1 1 2 1 2 2 2 2 2 1 1 1 1 1 2 2 2 1 2
##  [371] 2 1 1 1 1 1 1 2 1 1 2 2 2 2 1 1 2 1 2 1 2 1 2 2 2 1 1 2 2 2 1 1 2 1 2 1 1
##  [408] 2 1 1 1 1 2 2 2 1 2 1 1 1 2 2 2 1 2 1 2 2 2 2 1 1 2 1 1 2 1 1 1 2 1 2 1 1
##  [445] 1 2 2 1 1 2 2 2 2 1 2 2 1 2 1 2 2 1 2 2 1 2 2 2 2 2 1 1 2 1 1 2 2 2 1 1 1
##  [482] 1 2 2 2 2 2 2 2 1 2 2 1 1 1 1 1 2 2 1 1 2 1 1 2 1 2 2 1 2 2 2 2 1 2 1 1 1
##  [519] 1 2 1 2 1 2 1 2 1 2 2 2 2 2 2 1 2 1 1 2 1 1 1 1 2 1 1 2 2 1 1 2 2 1 1 2 2
##  [556] 1 2 2 1 1 1 1 2 1 2 2 1 2 2 2 1 1 2 1 1 2 2 1 2 1 2 1 1 2 2 1 2 2 2 1 1 2
##  [593] 2 1 1 2 2 1 2 1 2 1 2 1 1 2 1 2 2 1 1 2 2 2 1 2 2 1 2 1 1 1 1 2 2 2 1 2 2
##  [630] 2 2 1 2 2 2 2 2 2 1 2 1 2 1 1 1 1 2 2 1 2 1 1 2 2 2 1 1 2 1 2 2 2 1 1 2 2
##  [667] 1 2 1 2 1 1 2 2 1 2 2 2 2 1 1 2 1 2 2 2 2 2 1 1 1 1 1 2 2 1 1 2 1 1 1 2 2
##  [704] 1 2 2 1 2 2 1 1 2 2 2 2 2 1 1 2 1 1 1 2 1 1 1 2 2 2 2 1 2 2 1 1 1 2 2 1 2 2
##  [741] 1 1 2 2 1 2 2 1 1 1 1 1 1 2 2 1 2 1 1 2 1 2 2 1 2 1 2 2 2 1 2 1 2 2 1 2 2
##  [778] 1 2 1 2 2 1 1 2 2 1 2 1 1 2 2 2 2 2 1 2 1 1 1 1 2 1 1 2 2 2 2 2 2 1 1 1 1
##  [815] 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 1 2 1 1 1 2 2 2 1 2 2 1 2 2 1 2 1 2 1 1 1
##  [852] 2 1 1 2 2 2 2 2 2 2 1 2 1 1 2 2 1 1 2 2 2 1 1 1 1 1 2 2 2 2 1 2 2 1 1 2 1
##  [889] 1 2 2 1 2 2 1 2 2 1 1 2 2 2 1 1 1 1 1 1 2 2 2 1 1 2 2 2 1 2 1 1 1 2 2 2 1
##  [926] 1 2 1 1 2 2 1 1 1 2 1 2 2 1 2 1 1 2 2 1 2 2 1 2 2 1 2 1 2 1 2 2 1 1 1 1 1
##  [963] 2 2 2 2 2 1 1 1 2 1 2 2 1 2 1 2 2 1 2 1 1 2 1 1 1 2 1 1 2 2 2 2 1 2 1 1 2
## [1000] 1 1 2 2 2 1 2 2 1 1 1 2 2 2 1 2 1 2 1 2 1 2 2 2 1 1 2 2 1 1 1 1 2 2 2 2 1
## [1037] 2 1 2 1 2 2 1 2 2 1 1 1 2 1 2 2 1 2 1 1 1 2 1 2 2 2 1 2 2 2 2 1 1 2 1 2 1
## [1074] 2 1 1 2 1 2 1 2 1 1 2 2 1 2 2 2 1 2 2 2 2 1 2 2 1 2 1 2 1 1 1 1 1 1 1 2 2
## [1111] 2 2 1 2 1 2 2 1 2 1 1 2 2 2 2 1 2 1 2 2 1 2 1 2 2 1 2 1 1 1 2 1 1 2 1 1 2
## [1148] 1 1 2 2 2 2 1 1 2 1 2 1 2 2 2 1 1 2 2 1 2 2 2 1 2 1 1 2 1 2 2 1 1 2 1 1 1
## [1185] 2 2 2 2 2 2 1 1 1 2 1 2 2 1 1 1 2 1 2 2 2 2 2 1 1 1 2 2 2 1 1 2 2 2 1 1 2 2
## [1222] 1 1 2 1 2 2 2 1 2 1 2 2 2 2 2 2 2 1 2 1 2 1 2 1 1 2 2 1 1 1 1 2 2 1 1 2 2
## [1259] 2 2 1 1 1 1 2 2 1 1 2 1 1 2 2 2 2 1 1 1 2 2 1 1 2 2 1 2 1 1 1 1 2 1 2 2 2
## [1296] 1 1 2 2 1 2 1 2 1 2 1 2 1 1 2 1 2 2 2 1 2 1 1 1 1 2 2 1 1 1 2 1 2 2 2 1 2
## [1333] 1 2 2 2 2 2 2 1 1 2 1 1 1 1 2 1 2 1 1 2 2 2 2 2 2 2 1 1 2 1 1 2 1 2 2 2 1 1 2
## [1370] 2 1 2 2 1 1 2 2 1 2 2 1 1 2 1 1 1 2 2 1 1 2 1 2 2 2 2 1 1 2 1 1 1 1 2 1 2 2
## [1407] 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 1 2 2 2 2 2 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 2 2
## [1444] 2 1 2 2 2 2 2 1 1 2 2 2 1 1 1 1 2 1 1 2 2 2 1 1 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2
## [1481] 1 2 2 1 1 2 2 2 1 1 2 1 1 2 2 2 1 1 2 2 1 2 1 1 1 2 2 2 2 1 1 2 1 1 1 2 1
## [1518] 2 1 1 2 1 1 2 2 2 1 1 1 1 2 1 1 1 2 1 2 2 2 2 2 1 1 2 1 2 1 1 1 1 2 2 2 1 2
## [1555] 1 2 1 1 1 2 2 2 1 2 1 2 1 2 1 2 1 2 2 1 2 1 1 1 2 2 2 2 1 2 2 1 1 2 1 1 2
## [1592] 1 1 1 2 2 1 1 2 2 2 2 1 2 1 2 2 1 2 2 2 2 2 1 2 1 1 2 1 2 1 2 1 2 1 1 1 2 2
## [1629] 2 1 1 2 2 1 1 1 1 2 2 2 2 1 1 2 1 1 1 2 2 2 1 2 1 2 1 2 2 2 1 2 2 1 2 1 1
## [1666] 1 1 2 2 1 2 1 2 2 1 1 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 1 2 1 1 2 1 2 2 2 2
## [1703] 1 1 1 1 1 2 1 1 1 1 2 1 2 2 2 1 1 2 1 2 1 2 2 2 2 2 1 2 1 2 2 2 2 1 1 1 2
```

```
## [1740] 2 2 1 2 1 2 2 2 2 1 2 2 1 1 2 2 2 1 1 1 2 2 2 1 1 1 1 2 2 1 1 1 1 2 1 1 1
## [1777] 2 2 1 1 1 2 2 2 2 2 2 2 1 2 1 2 1 2 2 2 1 1 1 1 2 1 2 2 2 2 2 2 1 1 2 2 2
## [1814] 2 1 1 1 2 1 1 1 1 2 2 2 1 2 1 2 1 2 2 2 1 2 2 2 1 2 1 1 1 1 1 2 1 2 1 2 2
## [1851] 2 1 1 1 2 2 2 1 1 2 1 2 2 2 2 1 1 1 2 2 1 1 1 2 2 1 1 1 2 1 1 2 2 2 2 2 1
## [1888] 2 1 2 2 2 1 2 1 1 2 2 1 1 1 2 1 2 2 2 1 1 2 2 2 1 1 2 2 2 1 1 1 2 1 2 2 2
## [1925] 2 2 1 2 1 1 2 1 1 1 1 2 2 2 2 1 1 2 1 1 1 2 2 2 1 2 1 2 1 1 2 2 1 2 1 1 1
## [1962] 2 1 1 1 2 1 2 2 2 1 2 1 2 1 2 1 2 2 1 2 1 1 2 2 1 1 2 1 1 2 2 2 1 2 2 2 1
## [1999] 1 1 2 2 1 1 2 2 1 2 2 1 1 2 1 2 2 1 1 2 2 1 1 2 1 2 1 2 2 2 2 1 2 2 1 1 1
## [2036] 2 1 2 1 1 2 2 2 2 1 1 1 1 2 2 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 2 2 2 2 1
## [2073] 2 2 2 2 2 1 1 1 2 1 2 1 2 2 2 1 1 2 2 1 2 1 2 1 1 2 2 1 2 2 1 1 1 2 1 1 2
## [2110] 2 2 2 1 1 2 2 2 1 2 2 2 1 1 1 1 1 2 1 2 2 1 1 2 1 1 2 1 2 2 2 1 1 2 1 1 2
## [2147] 1 2 2 1 2 2 2 1 2 2 1 2 2 2 1 1 1 1 2 1 2 1 2 1 2 2 1 1 1 1 1 2 2 1 2 1 1
## [2184] 2 2 2 1 1 2 2 2 1 2 2 2 2 1 2 1 2 2 1 2 2 1 2 2 1 1 1 1 1 2 1 2 2 2 1 2 2
## [2221] 1 2 1 2 2 1 1 2 1 2 1 2 2 2 2 1 1 2 1 1 1 1 1 1 2 2 2 1 1 2 2 2 2 1 2 2 2
## [2258] 1 2 1 1 2 2 1 2 2 1 2 1 1 2 2 1 1 2 1 1 2 2 1 1 2 2 1 1 1 2 1 1 1 2 1 2 2
## [2295] 2 2 1 1 2 2 2 1 2 2 1 2 2 1 1 1 1 2 2 2 2 2 1 1 1 2 1 1 1 1 2 1 2 2 2 2 2
## [2332] 2 1 2 2 2 2 1 2 1 1 2 1 2 2 1 2 2 1 2 1 2 2 1 1 1 1 1 2 2 2 1 1 1 2 2 1 2
## [2369] 1 2 1 2 1 1 2 2 2 2 1 2 1 2 2 1 2 2 1 1 2 2 1 2 1 1 1 2 1 1 2 2 2 2 1 1 2
## [2406] 1 1 1 2 2 2 2 1 2 2 2 1 1 1 1 1 2 2 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 1 1 1 1
## [2443] 1 2 2 2 1 2 1 1 2 2 2 1 2 1 2 2 1 1 2 1 2 2 1 2 2 1 1 2 1 1 1 2 1 1 2 1 2
## [2480] 2 1 1 1 2 2 2 2 1 2 1 1 2 2 1 2 1 2 2 2 2 1 1 2 1 1 1 1 1 2 2 2 2 2 1 2 1 1 2
## [2517] 2 1 2 1 2 1 1 1 2 2 2 1 2 2 2 1 1 2 1 2 1 1 2 1 1 2 1 2 1 2 2 1 2 2 1 2 1
## [2554] 2 1 1 2 1 1 2 1 1 2 2 2 2 1 2 1 2 2 2 1 1 2 2 1 2 2 1 1 2 1 2 2 2 1 1 2 1
## [2591] 1 1 2 1 1 2 1 2 2 2 1 1 2 1 2 2 2 2 2 1 1 1 2 1 1 2 2 1 1 1 2 2 2 2 2 2 1
## [2628] 2 1 1 2 2 2 1 2 2 1 2 2 2 1 1 1 1 2 2 1 1 1 2 1 1 1 1 1 2 1 2 1 1 2 2 2 1
## [2665] 1 2 1 2 1 2 1 2 2 2 1 1 1 2 2 2 1 2 2 1 1 1 2 2 2 1 2 2 1 2 1 1 1 2 2 2 2 1 2 1
## [2702] 2 1 1 2 2 2 1 1 2 2 2 2 1 1 2 1 1 1 2 1 1 1 1 2 1 1 2 2 2 2 1 1 1 2 1 2 1
## [2739] 2 1 1 2 1 1 2 2 2 2 2 1 2 1 1 1 2 2 1 2 2 1 1 1 1 2 1 2 2 1 2 2 2 1 2 1 2
## [2776] 2 1 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 1 1 1 2 1 1 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2
## [2813] 1 1 1 1 2 1 2 1 2 2 2 2 2 1 1 1 2 1 1 2 2 2 1 2 2 1 2 1 1 1 1 1 2 1 2 1 1 1
## [2850] 2 2 1 2 2 2 1 2 1 2
```

```
crsc96_small[, 1]
```

```
## # A tibble: 2,859 x 1
##     sexq
##    <dbl>
##  1     2
##  2     2
##  3     2
##  4     1
##  5     1
##  6     2
##  7     1
##  8     1
##  9     1
## 10     1
## # ... with 2,849 more rows
```

Classe des variables

Avant de travailler avec une variable, nous devons savoir de quel type elle est. Nous savons par exemple que nous ne pouvons pas faire des calculs avec des variables nominales. La fonction class, nous permet de déterminer le type d'une variable.

```
class(crsc96_small$q1)
```

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

```
class(crsc96_small$q2)
```

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

```
class(crsc96_small$sexq)
```

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

```
class(crsc96_small$age)
```

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

- Allons voir si cette variable est vraiment numérique. Cette information se trouve dans le codebook (dictionnaire) de cette étude. Ces variables ont donc besoin d'être recodé correctement si elles ne sont pas numériques. Le recodage d'une variable signifie qu'on change les modalités de cette variable. Mais, une bonne pratique consiste `toujours` à créer une nouvelle variable à partir de l'ancienne variable et à recoder cette dernière. de ce fait, on ne risque pas de commettre des erreurs et d'écraser par mégarde une variable de notre base de données.

Avant d'aller recoder/créer des variables factorielles, voyons les chagements qu'on peut faire à partir d'une variable conitibue

Changement d'échelle d'une variable numérique

Pour toute variable continue, on peut utiliser les opérations mathématiques pour créer une nouvelle variable. Comme par exemple, le double de l'âge, ou l'écat de l'age de chaque individu par rapport à la moyenne. Comment pensez-vous qu'on va créer cette variable `age_ecart`?

```
crsc96_small <-
  crsc96_small %>%
  mutate(age_double =  age*2,
         age_écart = age - mean(age))
```

L'une des modification les plus importantes pour une variable continue est le calcul du score z (z-score en anglais), ou score standardisé. Il désigne le nombre d'écarts-types par rapport à la moyenne d'un point de données. Par exemple, un individu qui a une valeur de z-score de 2, signifie qu'il se trouve à deux écarts-types de la moyenne. Le calcul des z-scores permet de comparer les populations entre elle, en enlevant l'unité de mesure spécifique dans laquelle les mesures ont été faites. Les tests standardisés, les poids et taille standardisés sont des z-scores. Il se calcule de la manière suivante:

```
crsc96_small <-
  crsc96_small %>%
  mutate(age_standardise = (age - mean(age))/sqrt(age))
```

Recodage et création de variables factorielles

- La création de nouvelles variables se fait avec la commande `mutate`

```
crsc96_small <-
  crsc96_small %>%
  mutate(q1_new = q1)

class(crsc96_small$q1_new)
```

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

```
head(crsc96_small)
```

```
## # A tibble: 6 x 14
##    sexq region   age  ageq    q1    q2    q3    q4   q44   q95 age_double
##   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>      <dbl>
## 1     2      9    33     3     1     5     5     5     4     5         66
## 2     2      9    34     3     2     5     4     5     5     5         68
## 3     2      9    56     4     2     2     4     5     5     4        112
## 4     1      9    69     5     1     4     2     4     5     5        138
## 5     1      9    43     3     4     4     4     5     5     4         86
## 6     2      9    28     2     4     5     4     5     5     5         56
## # ... with 3 more variables: age_écart <dbl>, age_standardise <dbl>,
## #   q1_new <dbl>
```

On voit que q1_new est exactement comme q1. Mais ce n'est pas ce que nous voulons. Nous voulons que ce soit une variable factorielle.

Recodage et création de variables facorielles

```
crsc96_small <-
  crsc96_small %>%
  mutate(q1_new = as.factor(q1))

class(crsc96_small$q1_new)
```

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

```
head(crsc96_small)
```

```
## # A tibble: 6 x 14
##    sexq region   age  ageq    q1    q2    q3    q4   q44   q95 age_double
##   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>      <dbl>
## 1     2      9    33     3     1     5     5     5     4     5         66
## 2     2      9    34     3     2     5     4     5     5     5         68
## 3     2      9    56     4     2     2     4     5     5     4        112
## 4     1      9    69     5     1     4     2     4     5     5        138
## 5     1      9    43     3     4     4     4     5     5     4         86
## 6     2      9    28     2     4     5     4     5     5     5         56
## # ... with 3 more variables: age_écart <dbl>, age_standardise <dbl>,
## #   q1_new <fct>
```

Une autre façon de créer une variable factorielle est d'utiliser la fonction **factor**.

Si vous tapez **?factor** dans le chunk (c'est quoi un chunk?), il vous indiquera comment utiliser cette fonction.

```
?factor
```

On voit ainsi dans RStudio, dans la fenêtre **Help**, la description de cette fonction et son usage. Certains de ces arguments (ce qui se trouve dans la parenthèse) sont optionnelles, alors que d'autres sont obligatoires. La fonction factor permet de créer une variable **qualitative** ou **categorielle** ou **factorielle**, à partir d'une autre variable. Il faut donc lui indiquer cette variable à partir de laquelle on créer la nouvelle variable factorielle. Dans notre cas (chunk ligne 47, il s'agit de la variable c(1, 0, 0, 1, 0). De manière optionnelle, il faut lui indiquer à quoi représentent les valeurs de l'ancienne variable. ce;a se fait avec **labels**.

Recodage et création de variables facorielles

Finalement, bien qu'il soit une variable factorielle, on ne sait pas ce que signifie ses modalités. Il nous faut chaque fois retourner dans le codebook pour ce faire. On peut faire simplement avec la fonction factor qui prend au moins deux arguments: la variable et le labels des modalités. Observer que ce label se fait selon l'ordre des modalités.

```
crsc96_small <-
  crsc96_small %>%
  mutate(q1_new = factor(q1, labels = c("totally agree", "agree somewhat", "DK/NA", "disagree somewhat"

class(crsc96_small$q1_new)
```

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

```
head(crsc96_small)
```

```
## # A tibble: 6 x 14
##    sexq region   age  ageq    q1    q2    q3    q4   q44   q95 age_double
##   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>      <dbl>
## 1     2      9    33     3     1     5     5     5     4     5         66
## 2     2      9    34     3     2     5     4     5     5     5         68
## 3     2      9    56     4     2     2     4     5     5     4        112
## 4     1      9    69     5     1     4     2     4     5     5        138
## 5     1      9    43     3     4     4     4     5     5     4         86
## 6     2      9    28     2     4     5     4     5     5     5         56
## # ... with 3 more variables: age_écart <dbl>, age_standardise <dbl>,
## #   q1_new <fct>
```

```
freq(crsc96_small$q1_new)
```

```
## Frequencies
## crsc96_small$q1_new
## Type: Factor
##
##                          Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
## ----------------------- ------ --------- -------------- --------- --------------
##           totally agree   1065     37.25          37.25     37.25          37.25
##          agree somewhat   1410     49.32          86.57     49.32          86.57
##                   DK/NA      9      0.31          86.88      0.31          86.88
##       disagree somewhat    326     11.40          98.29     11.40          98.29
##        totally disagree     49      1.71         100.00      1.71         100.00
##                    <NA>      0                                0.00         100.00
##                   Total   2859    100.00         100.00    100.00         100.00
```

Finalement, il faut indiquer l'ordre dans lequel les valeurs dela nouvelle variables vont être affichées. Si on ne lui indique rien, il va les afficher par ordre alphabétique. Autrement, il faut lui indiquer l'ordre que vous souhaitez, avec l'option **levels**. Vous avez plusieurs autres arguments que vous pouvez aller consulter vous-mêmes. Finalement, la description se termine toujours par des exemples d'utilisation.

Si on veut changer toutes les variables qui sont catégorielles

```
qlabel <- c("totally agree", "agree somewhat", "DK/NA", "disagree somewhat", "totally disagree")

crsc96_small <-
```

```
  crsc96_small %>%
  mutate(q2_new = factor(q2, labels = qlabel),
         q3_new = factor(q3, labels = qlabel))

freq(crsc96_small$q2)
```

```
## Frequencies
## crsc96_small$q2
## Type: Numeric
##
##               Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
## ----------- ------ --------- -------------- --------- --------------
##           1    516     18.05          18.05     18.05          18.05
##           2    636     22.25          40.29     22.25          40.29
##           3     26      0.91          41.20      0.91          41.20
##           4    894     31.27          72.47     31.27          72.47
##           5    787     27.53         100.00     27.53         100.00
##        <NA>      0                                0.00         100.00
##       Total   2859    100.00         100.00    100.00         100.00
```

If_else pour créer des variables binaires ou dichotomiques

Supposons que nous voulons scinder la variable age en deux catégories, alors on peut utiliser la commande if_else

```
crsc96_small <-
  crsc96_small %>%
  mutate(age2 = if_else(age >= 35, "adulte", "jeune"))

class(crsc96_small$age2)
```

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

```
# Transformer en facteur

crsc96_small <-
  crsc96_small %>%
  mutate(age2bis = as.factor(if_else(age >= 35, "adulte", "jeune")))

# utiliser factor

crsc96_small <-
  crsc96_small %>%
  mutate(ageter = if_else(age >= 35, 1, 2))

class(crsc96_small$ageter)
```

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

Pour le transformer en une variable factorielle, il faut juste utiliser factor au début de la création de la variable:

Commande case_when pour des cas plus généraux

```
crsc96_small <-
  crsc96_small %>%
  mutate(age4 = case_when(
    age < 20 ~ "adolescent",
    age >= 20 & age < 34 ~ "jeune",
    age >= 35 & age <59 ~ "adulte",
    age >=60 ~ "ainé"
  ))

class(crsc96_small$age4)
```

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

```
head(crsc96_small)
```

```
## # A tibble: 6 x 20
##    sexq region   age  ageq    q1    q2    q3    q4   q44   q95 age_double
##   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>      <dbl>
## 1     2      9    33     3     1     5     5     5     4     5         66
## 2     2      9    34     3     2     5     4     5     5     5         68
## 3     2      9    56     4     2     2     4     5     5     4        112
## 4     1      9    69     5     1     4     2     4     5     5        138
## 5     1      9    43     3     4     4     4     5     5     4         86
## 6     2      9    28     2     4     5     4     5     5     5         56
## # ... with 9 more variables: age_écart <dbl>, age_standardise <dbl>,
## #   q1_new <fct>, q2_new <fct>, q3_new <fct>, age2 <chr>, age2bis <fct>,
## #   ageter <dbl>, age4 <chr>
```

Si je veux que ce soit une variable factorielle, que dois-je faire?

Pour le rendre comme une variable catégorielle

```
crsc96_small <-
  crsc96_small %>%
  mutate(age5 = factor(case_when(
    age < 20 ~ 1,
    age >= 20 & age < 34 ~ 2,
    age >= 35 & age <59 ~ 3,
    age >=60 ~ 4), labels = c("adolescent", "jeune", "adulte", "ainé")
  ))

class(crsc96_small$age5)
```

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

Finalement, on peut créer la variable age en le scindant en 5 catégories

```
crsc96_small <-
  crsc96_small %>%
  mutate(age6 = ntile(age, 5))
```

```r
class(crsc96_small$age6)
```

```
## [1] "integer"
```

```r
freq(crsc96_small$age6)
```

```
## Frequencies
## crsc96_small$age6
##
##               Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
## ----------- ------ --------- -------------- --------- --------------
##           1    572     20.01          20.01     20.01          20.01
##           2    572     20.01          40.01     20.01          40.01
##           3    572     20.01          60.02     20.01          60.02
##           4    572     20.01          80.03     20.01          80.03
##           5    571     19.97         100.00     19.97         100.00
##        <NA>      0                                0.00         100.00
##       Total   2859    100.00         100.00    100.00         100.00
```

Je peux donc utiliser ntile en combinaison avec factor pour créer une variable factorielle

```r
crsc96_small <-
  crsc96_small %>%
  mutate(age6 = factor(ntile(age, 5), labels = c("g1", "g2", "g3", "g4", "g5")))

class(crsc96_small$age6)
```

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

```r
head(crsc96_small)
```

```
## # A tibble: 6 x 22
##    sexq region   age  ageq    q1    q2    q3    q4   q44   q95 age_double
##   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>      <dbl>
## 1     2      9    33     3     1     5     5     5     4     5         66
## 2     2      9    34     3     2     5     4     5     5     5         68
## 3     2      9    56     4     2     2     4     5     5     4        112
## 4     1      9    69     5     1     4     2     4     5     5        138
## 5     1      9    43     3     4     4     4     5     5     4         86
## 6     2      9    28     2     4     5     4     5     5     5         56
## # ... with 11 more variables: age_écart <dbl>, age_standardise <dbl>,
## #   q1_new <fct>, q2_new <fct>, q3_new <fct>, age2 <chr>, age2bis <fct>,
## #   ageter <dbl>, age4 <chr>, age5 <fct>, age6 <fct>
```

Application

- Créer la variable age au carré nommé `age_square`
- Recoder la variable q2 en trois catégories (`agree`, `dk`, et `disagree`) (variable factorielle) que vous nommez q2_3
- Créer une nouvelle variable qui permet de savoir combien de personne sont dans le groupe d'âge [25, 35], que vous nommez age_25_35
- Créer une variable scalaire avec l'âge

- De quel type est chaque variable?

```
crsc96_small <-
  crsc96_small %>%
  mutate(age_square = age^2,
         q2_3 = factor(case_when(
           q2 == 1 | q2 == 2 ~ 1,
           q2 == 3 ~ 2,
           q2 == 4 | q2 == 5 ~ 3), labels = c("agree", "dk", "disagree")),
         age_25_35 = between(age, 25, 34))

class(crsc96_small$age_25_35)
```

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

Autres fonctions pour construire des variables

```
crsc96_small <-
  crsc96_small %>%
  mutate(var1 = lead(age, n = 2),                    # Remplace var la valeur de rang 2
         var2 = lag(age, n = 1))
```

Exercices

- Explorer les autres fonctions de `Fonction "window"` de la feuille cheatsheet.