# Untitled

Malick SENE

2025-04-11

```
library(tidyverse)
library(questionr)
library(forcats)
library(ggplot2)
library(dplyr)
```

# 5. Recoder des variables

## 5.1 Recoder une variable qualitative

Pour rappel, on appelle variable qualitative une variable pouvant prendre un nombre limité de modalités(de valeurs possibles).

### 5.1.1 Facteurs et forcats

Dans **R**, les variables qualitatives peuvent être de deux types :
- des vecteurs de type `character` (chaînes de caractères),
- ou des `factor` (facteurs).

Si vous utilisez les fonctions des extensions du **tidyverse** comme `readr`, `readxl` ou `haven` pour importer vos données, vos variables qualitatives seront importées sous forme de `character`.
Mais dans les autres cas comme avec les fonctions de base de R (`read.table()`, `read.csv()`...), ou lors de la création manuelle de `data.frame` sans spécifier `stringsAsFactors = FALSE` elles se retrouveront souvent sous forme de `factor`.

## Qu'est-ce qu'un facteur (`factor`) ?

Un **facteur** est une structure de données utilisée en R pour représenter des **variables qualitatives** (ou catégorielles).
Plutôt que de stocker les valeurs comme de simples chaînes de caractères, un facteur associe chaque valeur à une **modalité** (ou **niveau**).

Par exemple, une variable "sexe" avec les valeurs `"Homme"`, `"Femme"`, `"Homme"` sera stockée comme un facteur avec deux niveaux : `"Femme"` et `"Homme"`.

Les facteurs sont particulièrement utiles pour : - représenter des **catégories avec un nombre limité de modalités** ; - faciliter certaines **analyses statistiques** (comme les modèles linéaires ou les tableaux croisés) ; - garantir une **cohérence des données** (éviter les fautes de frappe ou incohérences dans les noms de catégories).

Techniquement, un facteur est un vecteur d'entiers qui pointent vers un vecteur de labels (les niveaux).

Vous pouvez créer un facteur à partir d'un vecteur `character` avec la fonction `factor()` :

```
sexe <- factor(c("Homme", "Femme", "Homme"))
class(sexe)
```

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

```
library(haven)
wf <- read_dta("ehcvm_welfare_sen2021.dta")
```

Dans l'exemple de notre base de donnée welfare de EHCVM 2021 (wf), la variable zone agroécologique est au format character. Nous allons le convertir en facteur :

```
wf$zae <- haven::as_factor(wf$zae)
class(wf$zae)
```

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

```
unique(wf$zae)
```

```
## [1] Dakar                          Ziguinchor-Tamba-Kolda-Sédhiou
## [3] Thies-Diourbel-Louga           Saint-Louis-Matam
## [5] Kaolack-Fatick-Kaffrine        Kédougou
## 6 Levels: Kédougou Saint-Louis-Matam ... Dakar
```

## 5.2 Modifier les modalité d'une variable qualitative

Ceci complique les opérations de recodage, car du coup l'opérations suivante, qui tente de modifier une modalité de la variable, aboutit à un avertissement, et l'opération n'est pas effectuée :

```
wf$zae[wf$zae=="Kaolack-Fatick-Kaffrine"]<-"Bassin arachidier"
```

```
## Warning in `[<-.factor`(`*tmp*`, wf$zae == "Kaolack-Fatick-Kaffrine", value =
## structure(c(6L, : niveau de facteur incorrect, NAs générés
```

Une opération courante consiste à modifier les valeurs d'une variable qualitative, que ce soit pour avoir des intitulés plus courts ou plus clairs, ou pour regrouper des modalités entre elles.

Il existe plusieurs possibilités pour effectuer ce type de recodage, mais ici on va utiliser la fonction `fct_recode` de l'extension **forcats**. Celle-ci prend en argument une liste de recodages sous la forme `"Nouvelle valeur" = "Ancienne valeur"`.

Un exemple : nous allons essayer encore de recoder encore "Kaolack-Fatick-Kaffrine" par son ancienne valeur "Bassin arachidier" la région de Kédougou par "Sud-Est du Sénégal"

```
wf$zae <- fct_recode(wf$zae,
                     "Bassin arachidier" = "Kaolack-Fatick-Kaffrine", "Sud-Est du Sénégal" = "Kédougou"
unique(wf$zae)
```

```
## [1] Dakar                          Ziguinchor-Tamba-Kolda-Sédhiou
## [3] Thies-Diourbel-Louga           Saint-Louis-Matam
## [5] Bassin arachidier              Sud-Est du Sénégal
## 6 Levels: Sud-Est du Sénégal Saint-Louis-Matam ... Dakar
```

**Attention** : les anciennes valeurs saisies doivent être **exactement égales** aux valeurs des modalités de la variable recodée.
Toute différence d'accent ou d'espace fera que ce recodage ne sera pas pris en compte. Dans ce cas, `forcats` affiche un **avertissement** nous indiquant qu'une valeur saisie **n'a pas été trouvée** dans les modalités de la variable.

Si l'on souhaite recoder les `NA` d'une variable, on utilisera la fonction `fct_explicit_na`,
qui convertit toutes les valeurs manquantes (`NA`) d'un facteur en une **modalité spécifique** : Ici, nous recoder les `NA` de la variable hhsectins (secteur institutionnel du chef de ménage)

```
wf$hsectins <- haven::as_factor(wf$hsectins)
cat("Avant recodage avec les NA\n")
```

```
## Avant recodage avec les NA
```

```
unique(wf$hsectins)
```

```
## [1] Ménage comme employeur de personnel domestique
## [2] Entreprise Privée
## [3] Entreprise publique/ parapublique
## [4] <NA>
## [5] Etat/Collectivités locales
## [6] Organisme international /Ambassade
## [7] Entreprise associative
## 6 Levels: Etat/Collectivités locales ... Organisme international /Ambassade
```

```
wf$hsectins <- fct_explicit_na(wf$hsectins ,na_level= "valeurs manquantes")
cat("Après recodage de de NA en valeurs manquantes\n")
```

```
## Après recodage de de NA en valeurs manquantes
```

```
unique(wf$hsectins)
```

```
## [1] Ménage comme employeur de personnel domestique
## [2] Entreprise Privée
## [3] Entreprise publique/ parapublique
## [4] valeurs manquantes
## [5] Etat/Collectivités locales
## [6] Organisme international /Ambassade
## [7] Entreprise associative
## 7 Levels: Etat/Collectivités locales ... valeurs manquantes
```

D'autres fonctions sont proposées par **forcats** pour faciliter certains recodages, comme `fct_collapse`,
qui propose une autre syntaxe pratique quand on doit regrouper ensemble des modalités : dans cet exemple, toujours avec la variable hsectins, nous allons regouper les modalités " Entreprise Privée", "Entreprise publique/ parapublique" et "Entreprise associative" en "Entreprise" tout court.

```r
wf$hsectins <- fct_collapse(wf$hsectins ,
 "Entreprises" =c("Entreprise Privée","Entreprise publique/ parapublique","Entreprise associative"))
unique(wf$hsectins)
```

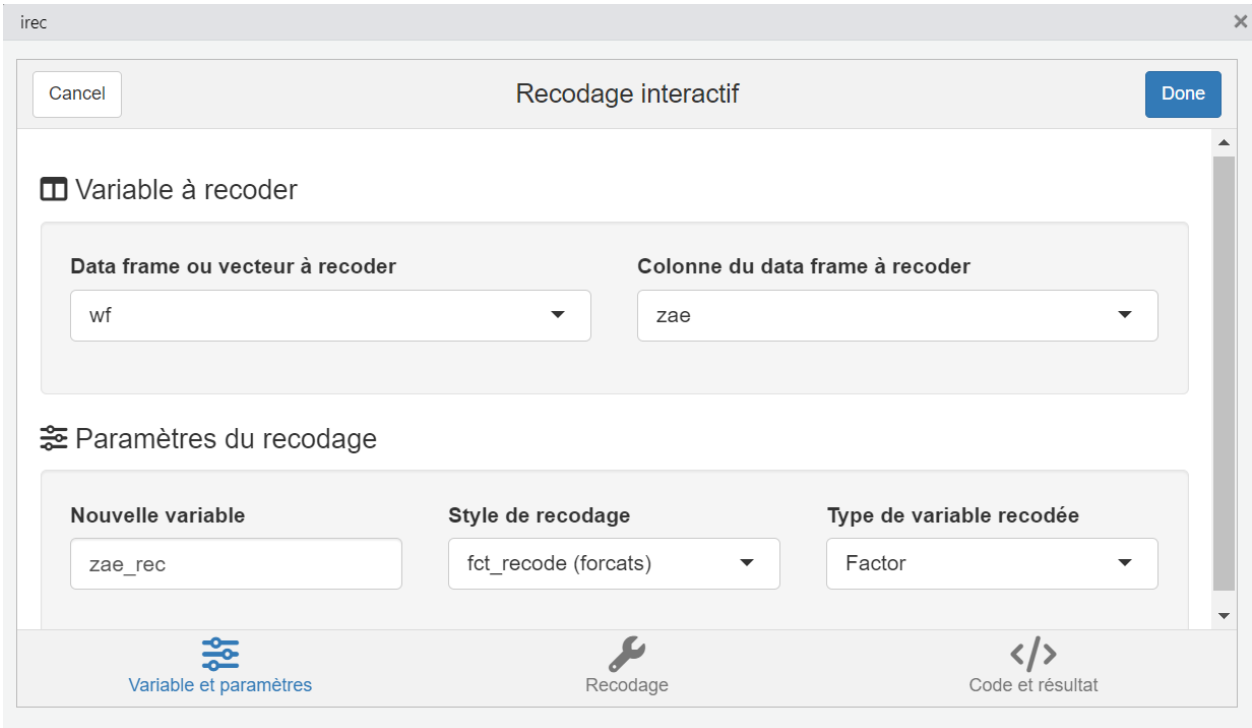
```
## [1] Ménage comme employeur de personnel domestique
## [2] Entreprises
## [3] valeurs manquantes
## [4] Etat/Collectivités locales
## [5] Organisme international /Ambassade
## 5 Levels: Etat/Collectivités locales ... valeurs manquantes
```

### 5.2.1 Interface graphique de recodage

L'extension **questionr** propose une interface graphique facilitant le recodage des valeurs d'une variable qualitative.
L'objectif est de permettre à l'utilisateur de saisir les nouvelles valeurs dans un formulaire,
et de générer ensuite le code R correspondant au recodage indiqué.

Pour utiliser cette interface, sous **RStudio**, vous pouvez aller dans le menu **Addins** (présent dans la barre d'outils principale),
puis choisir **Levels recoding**.

Sinon, vous pouvez lancer dans la console la fonction `irec()` en lui passant comme paramètre la variable à recoder.(ex :irec(wf$zae) )




L'interface se compose de trois onglets :

- L'onglet **Variable et paramètres** vous permet de sélectionner la variable à recoder, le nom de la nouvelle variable, ainsi que d'autres paramètres.
- L'onglet **Recodages** vous permet de saisir les nouvelles valeurs des modalités.

- L'onglet **Code et résultat** affiche le code R correspondant, ainsi qu'un tableau permettant de vérifier les résultats.

Une fois votre recodage terminé, cliquez sur le bouton **Done** :
le code R sera inséré dans votre script R ou affiché dans la console.

**Attention** : cette interface est prévue pour **ne pas modifier directement vos données**.
C'est donc à **vous d'exécuter manuellement le code généré** pour que le recodage soit réellement effectif.

## 5.3 Ordonner les modalités d'une variable qualitative

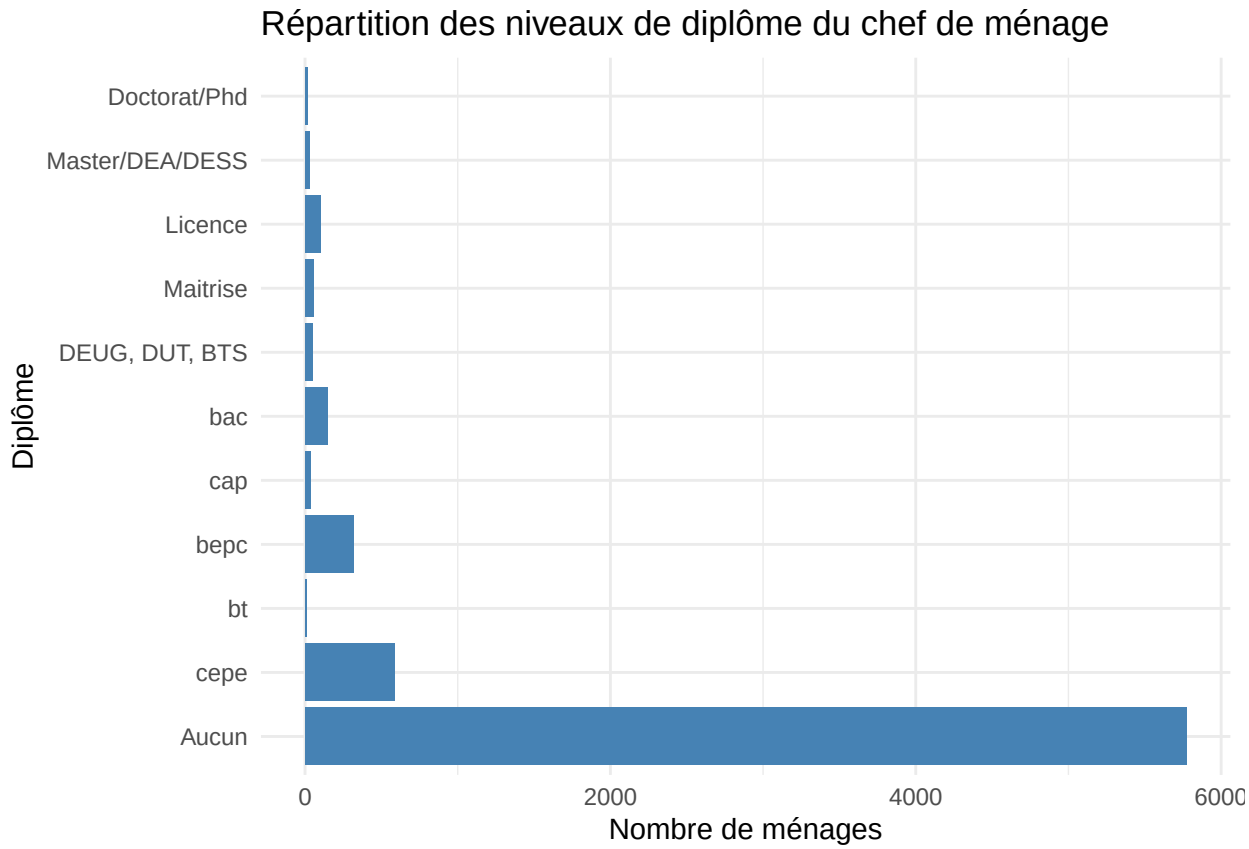
L'avantage des **facteurs** (par rapport aux vecteurs de type `character`) est que leurs **modalités peuvent être ordonnées**,
ce qui peut faciliter la lecture de tableaux ou de graphiques.

On peut ordonner les modalités d'un facteur **manuellement**, par exemple avec la fonction `fct_relevel()` de l'extension **forcats** : nous allons ordonner les modalités de la variable `hdiploma` dans un ordre progressif de niveau d'études,
puis afficher leur répartition sous forme de graphique.

```
wf$hdiploma <- haven::as_factor(wf$hdiploma)
unique(wf$hdiploma)
```

```
##  [1] Aucun           Licence         cepe            DEUG, DUT, BTS
##  [5] Doctorat/Phd    bepc            cap             bac
##  [9] Master/DEA/DESS Maitrise        bt
## 11 Levels: Aucun cepe bepc cap bt bac DEUG, DUT, BTS Licence ... Doctorat/Phd
```

```
# Reclasser les niveaux du diplôme
wf$hdiploma <- fct_relevel(wf$hdiploma,
  "Aucun",
  "cepe",
  "bt",
  "bepc",
  "cap",
  "bac",
  "DEUG, DUT, BTS",
  "Maitrise",
  "Licence",
  "Master/DEA/DESS",
  "Doctorat/PhD"
)
# Créer un graphique pour visualiser l'ordre des diplômes
wf %>%
  filter(!is.na(hdiploma)) %>%
  ggplot(aes(x = hdiploma)) +
  geom_bar(fill = "steelblue") +
  coord_flip() +
  labs(
    title = "Répartition des niveaux de diplôme du chef de ménage",
    x = "Diplôme",
    y = "Nombre de ménages"
  ) +
  theme_minimal()
```




Ce graphique permet de **voir l'effet de `fct_relevel()`**, car les barres sont maintenant affichées selon l'ordre logique d'étude — du niveau **"Aucun"** jusqu'à **"Doctorat/PhD"**.

Une autre possibilité utile est d'**ordonner les modalités d'un facteur selon les valeurs d'une autre variable numérique**.

Par exemple, on peut vouloir afficher les niveaux de diplôme (`hdiploma`) selon le **revenu moyen par niveau d'étude**, mesuré ici par `pcexp` (indicateur de bien-être par tête).

Cela permet de **rendre les visualisations plus informatives** : au lieu d'un ordre alphabétique ou arbitraire, les catégories sont organisées selon leur valeur moyenne (du plus faible au plus élevé). Pour cela, on utilise la fonction `fct_reorder()`

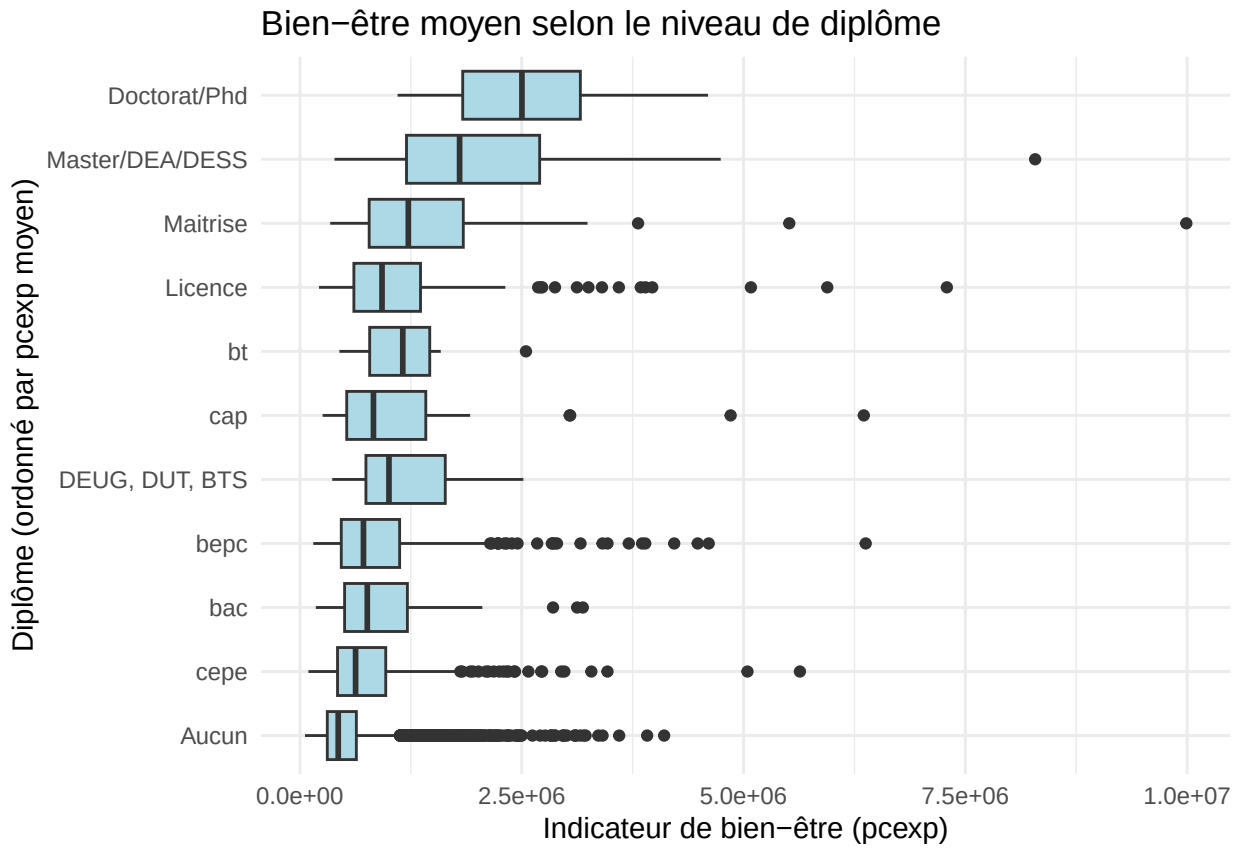
La fonction `fct_reorder()` de l'extension `forcats` est utilisée pour cela.
Elle prend 3 arguments :

1. Le facteur à réordonner (ex. `hdiploma`)
2. La variable numérique à utiliser comme critère d'ordre (ex. `pcexp`)
3. Une fonction d'agrégation (ex. `mean`, `median`, `sum`, etc.)

```r
# Créer une nouvelle variable réordonnée
wf$hdiploma_reord <- fct_reorder(wf$hdiploma, wf$pcexp, mean)


ggplot(wf, aes(x = hdiploma_reord, y = pcexp)) +
  geom_boxplot(fill = "lightblue") +
  coord_flip() +
  labs(
```

```
    title = "Bien-être moyen selon le niveau de diplôme",
    x = "Diplôme (ordonné par pcexp moyen)",
    y = "Indicateur de bien-être (pcexp)"
  ) +
  theme_minimal()
```



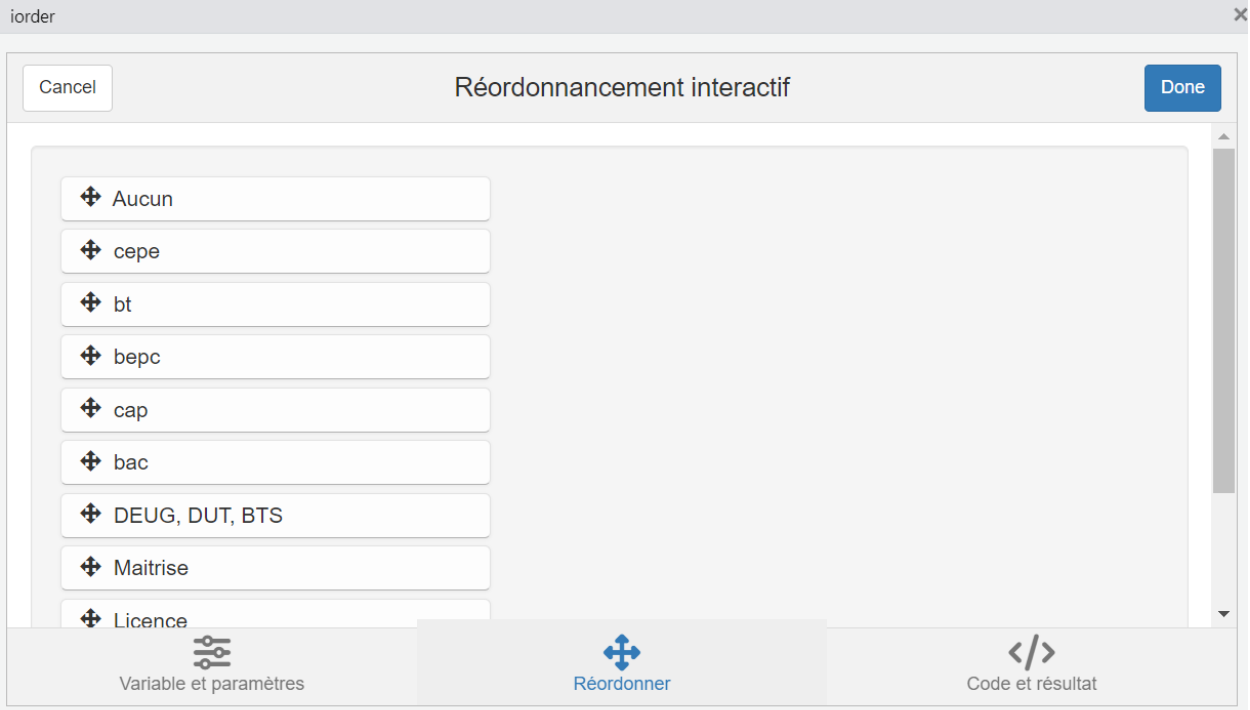

Ce graphique illustre la relation entre le **niveau de diplôme du chef de ménage** (`hdiploma`) et l'**indicateur de bien-être par tête** (`pcexp`).

Les modalités de la variable `hdiploma` ont été **ordonnées selon la moyenne de `pcexp`**, ce qui permet d'observer facilement les tendances.

### 5.3.1 Interface graphique

On peut aussi utiliser l'**interface graphique** proposée par l'extension **questionr** afin de faciliter cette opération de réordonnancement.

Pour la lancer, vous pouvez : - sélectionner le menu **Addins** dans RStudio, puis choisir **Levels ordering**, - ou exécuter la fonction `iorder()` en lui passant comme paramètre le facteur à réordonner : ex(iorder(wf$hdiploma))




Le fonctionnement de l'interface est similaire à celui de l'interface de recodage.

Vous pouvez **réordonner les modalités** en les faisant glisser avec la souris,
puis **récupérer et exécuter le code R généré** pour que le réordonnancement soit effectif.

## 5.4 Combiner plusieurs variables : if_else, case_when

Parfois, on souhaite créer une **nouvelle variable** à partir des valeurs d'une ou plusieurs autres variables. Dans ce cas, on peut utiliser :

- la fonction `if_else()` pour les **cas simples** (conditions binaires),
- ou la fonction `case_when()` pour les **cas plus complexes** impliquant plusieurs conditions.

La fonction `if_else()` prend trois arguments :

1. Un **test logique** (`condition`)
2. Une **valeur à renvoyer si le test est vrai**
3. Une **valeur à renvoyer si le test est faux**

On souhaite créer une nouvelle variable `categorie_age` qui catégorise les chefs de ménage comme "Jeune" ou "Âgé" selon leur âge (`hage`).

```
wf <- wf %>%
  mutate(categorie_age = if_else(hage < 40, "Jeune", "Âgé"))
table(wf$categorie_age)
```

```
##
##   Âgé Jeune
##  6037  1083
```

La fonction `case_when()` est une **généralisation** de `if_else()` qui permet d'indiquer **plusieurs tests logiques** et leurs valeurs associées.

Imaginons que l'on souhaite créer une nouvelle variable qui identifie :

- les **hommes de plus de 60 ans**,
- les **femmes de plus de 60 ans**,
- et **tous les autres**.

```
wf$hgender <- as_factor(wf$hgender)

wf$profil_age60 <-  case_when(
    wf$hgender == "Masculin" & wf$hage > 60 ~ "Homme +60 ans",
    wf$hgender == "Féminin" & wf$hage > 60 ~ "Femme +60 ans",
    TRUE ~ "Autres"
  )
table(wf$profil_age60)
```

```
##
##        Autres Femme +60 ans Homme +60 ans
##          4754           707          1659
```

On peut aussi porceder ainsi.

```
wf <- wf %>%
  mutate(profil_age60_2 = case_when(
    hgender == "Masculin" & hage > 60 ~ "Homme +60 ans",
    hgender == "Féminin" & hage > 60 ~ "Femme +60 ans",
    TRUE ~ "Autres"
  ))
table(wf$profil_age60_2)
```

```
##
##        Autres Femme +60 ans Homme +60 ans
##          4754           707          1659
```

La fonction `case_when()` prend en arguments une série d'instructions sous la forme : condition ~ valeur

**Attention** : comme les conditions sont testées **l'une après l'autre**, la valeur renvoyée est celle correspondant à la **première condition vraie**.

Cela signifie que **l'ordre des conditions est très important** :
il faut **absolument aller du plus spécifique au plus général** pour éviter que des cas généraux capturent des situations particulières par erreur.

```
wf$statut <- case_when(
  wf$hgender == "Masculin" ~ "Masculin",
  wf$hgender == "Masculin" & wf$hage > 60 ~ "Homme de plus de 60 ans",
  TRUE ~ "Autre"
)

# Résumé des résultats
table(wf$statut)
```

```
##
##    Autre Masculin
##     2025     5095
```

Dans ce code, la première condition `sexe == "Homme"` capture **tous les individus de sexe masculin**, ce qui empêche la deuxième condition (`sexe == "Homme" & age > 60`) de s'exécuter, car elle **n'est jamais atteinte**.

```
wf$statut_2 <- case_when(
  wf$hgender == "Masculin" & wf$hage > 60 ~ "Homme de plus de 60 ans",
    wf$hgender == "Masculin" ~ "Masculin",
  TRUE ~ "Autre"
)

# Résumé des résultats
table(wf$statut_2)
```

```
##
##                   Autre Homme de plus de 60 ans                 Masculin
##                    2025                    1659                    3436
```

Grâce à cet ordre corrigé, les **hommes de plus de 60 ans** sont correctement identifiés **avant d'assigner les autres hommes** dans la catégorie plus générale `"Homme"`.

## 5.5 Découper une variable numérique en classe

Une opération relativement courante consiste à **découper une variable numérique en classes**. Ici, on crée une variable agecl, qui regroupe l'age des chefs de ménages par classe d'interval 5.

```
wf$agecl <- cut(wf$hage,breaks = 5)
table(wf$agecl )
```

```
##
## (15.9,33]   (33,50]   (50,67]   (67,84]  (84,101]
##       412      2563      2890      1161        94
```

Si l'on donne un **nombre entier** à l'argument `breaks`, un nombre correspondant de **classes d'amplitudes égales** est automatiquement calculé.

Il est cependant souvent **préférable d'avoir des limites "rondes"**.
Dans ce cas, on peut les spécifier manuellement en passant un **vecteur de coupures** à l'argument `breaks` :

```
wf$agecl <- cut(wf$hage,breaks= c(15.9,25, 35,45,55,65,97),
 include.lowest= TRUE)
table(wf$agecl )
```

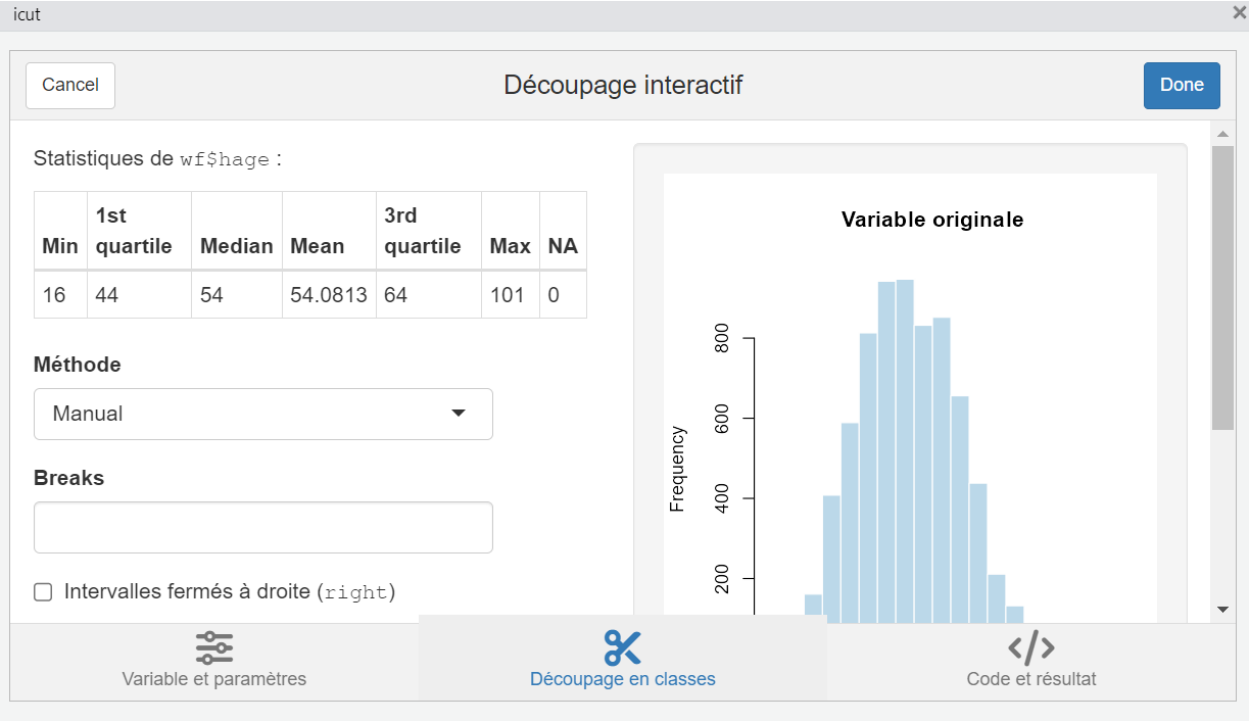
```
##
## [15.9,25]   (25,35]   (35,45]   (45,55]   (55,65]   (65,97]
##        62       569      1402      1889      1684      1512
```

Ici, on a été obligé d'ajouter l'argument `include.lowest = TRUE`,
car sinon la valeur `15.9` n'aurait pas été incluse dans la première classe,
et cela aurait entraîné la présence de **valeurs manquantes (`NA`)** dans la variable découpée.

### 5.5.1 Interface graphique

L'extension **questionr** propose une **interface graphique** facilitant cette opération de **découpage en classes** d'une variable numérique.

Pour lancer cette interface sous **RStudio**, vous pouvez : - ouvrir le menu **Addins** et sélectionner **Numeric range dividing**, - ou exécuter la fonction `icut()` dans la console en lui passant comme argument la variable quantitative à découper : icut(wf$hage)




Vous pouvez alors :

- choisir la variable à découper dans l'onglet **Variable et paramètres**,
- indiquer les **limites de vos classes** ainsi que quelques **options complémentaires** dans l'onglet **Découpage en classes**,
- et vérifier le résultat dans l'onglet **Code et résultat**.

Une fois le découpage satisfaisant, cliquez sur **Done** :

- Si vous êtes sous **RStudio**, le **code généré** sera directement inséré dans votre **script actuel** à l'emplacement du curseur.
- Sinon, ce code sera affiché dans la **console**, et vous pourrez le **copier/coller** pour l'inclure dans votre script.