Les factors dans R

Introduction

Les facteurs sont une classe de données importante dans R pour représenter et travailler avec des variables catégorielles. Dans cette leçon, nous allons apprendre à créer des facteurs et à les manipuler avec des fonctions du package forcats, qui fait partie du tidyverse. Plongeons-nous dedans !

Objectifs d'apprentissage

- Vous comprenez ce que sont les facteurs et en quoi ils diffèrent des caractères dans R.
- Vous êtes capable de modifier **l'ordre** des niveaux des facteurs.
- Vous êtes capable de modifier **la valeur** des niveaux des facteurs.

Packages

```
# Load packages
if(!require(pacman)) install.packages("pacman")
pacman::p_load(tidyverse,
               here)
```

Jeu de données : Mortalité VIH

Nous allons utiliser un jeu de données contenant des informations sur la mortalité VIH en Colombie de 2010 à 2016, hébergé sur la plateforme de données ouvertes 'Datos Abiertos Colombia'. Vous pouvez en savoir plus et accéder à l'ensemble du jeu de données ici.

Chaque ligne correspond à un individu décédé du SIDA ou de complications liées au SIDA.

```
        hiv_mort <-
read_csv(here("data/colombia_hiv_deaths_2010_to_2016.csv"))
```

```
## # A tibble: 5 × 25
##   municipality_type death_location  birth_date birth_year
##   <chr>             <chr>           <date>          <dbl>
## 1 Municipal head    Hospital/clinic 1956-05-26       1956
## 2 Municipal head    Hospital/clinic 1983-10-10       1983
## 3 Municipal head    Hospital/clinic 1967-11-22       1967
## 4 Municipal head    Home/address    1964-03-14       1964
## 5 Municipal head    Hospital/clinic 1960-06-27       1960
## # ℹ 21 more variables: birth_month <chr>, birth_day <dbl>,
## #   death_year <dbl>, death_month <chr>, death_day <dbl>, …
```

Voir l'annexe au bas pour le dictionnaire de données décrivant toutes les variables.

Qu'est-ce que les facteurs ?

Les facteurs sont une classe de données importante dans R utilisée pour représenter des variables catégorielles.

Une variable catégorielle prend un ensemble limité de valeurs ou niveaux possibles. Par exemple, pays, race ou affiliation politique. Celles-ci diffèrent des variables texte libre qui prennent des valeurs arbitraires, comme des noms de personnes, titres de livres ou commentaires de médecins.

RECAP

Rappel des principales classes de données dans R

- `Numérique` : Représente des données numériques continues, incluant des nombres décimaux.
- `Entier` : Spécifiquement pour les nombres entiers sans décimales.

RECAP

- `Caractère` : Utilisé pour les données textuelles ou chaînes de caractères.
- `Logique` : Représente des valeurs booléennes (VRAI ou FAUX).
- `Facteur` : Utilisé pour les données catégorielles avec des niveaux ou catégories prédéfinis.
- `Date` : Représente des dates sans heures.

Les facteurs ont quelques avantages clés par rapport aux vecteurs de caractères pour travailler avec des données catégorielles dans R :

- Les facteurs sont stockés dans R de manière légèrement plus efficace que les caractères.
- Certaines fonctions statistiques, comme `lm()`, nécessitent que les variables catégorielles soient passées en paramètre sous forme de facteurs.
- Les facteurs permettent de contrôler l'ordre des catégories ou niveaux. Cela permet de trier et tracer correctement les données catégorielles.

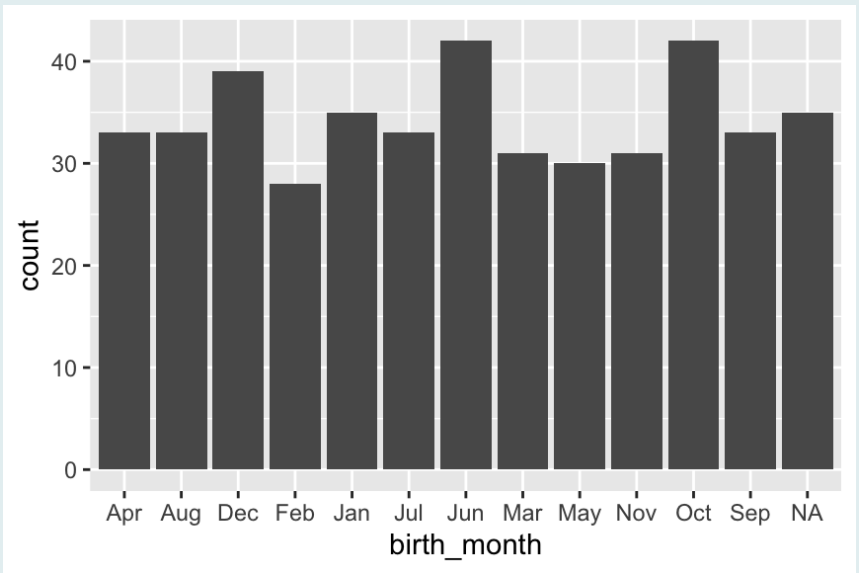
Ce dernier point, le contrôle de l'ordre des niveaux de facteurs, sera notre objectif principal.

Les facteurs en action

Voyons un exemple concret de l'intérêt des facteurs en utilisant le jeu de données `hiv_mort` que nous avons chargé précédemment.

Supposons que vous souhaitez visualiser les patients du jeu de données par leur mois de naissance. Nous pouvons le faire avec ggplot :

```
ggplot(hiv_mort) +
  geom_bar(aes(x = birth_month))
```



Cependant, il y a un problème : l'axe des x (qui représente les mois) est classé alphabétiquement, avec Avril en premier à gauche, puis Août, etc. Mais les mois devraient suivre un ordre chronologique spécifique !

Nous pouvons arranger le graphique dans l'ordre souhaité en créant un facteur avec la fonction `factor()` :

```
hiv_mort_modified <-
  hiv_mort %>%
  mutate(birth_month = factor(x = birth_month,
                              levels = c("Jan", "Feb", "Mar", "Apr",
                                         "May", "Jun", "Jul", "Aug",
                                         "Sep", "Oct", "Nov",
"Dec")))
```

La syntaxe est simple : l'argument `x` prend la colonne de caractères d'origine, `birth_month`, et l'argument `levels` prend la séquence désirée de mois.

Lorsque nous inspectons le type de données de la variable birth_month, nous pouvons voir sa transformation :

```
# Modified dataset
class(hiv_mort_modified$birth_month)
```

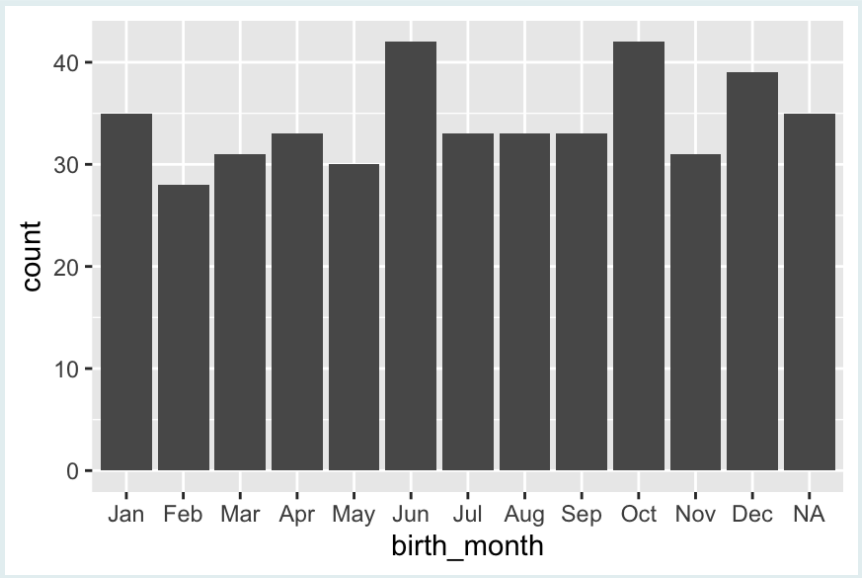
```
## [1] "factor"
```

```
# Original dataset
class(hiv_mort$birth_month)
```

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

Maintenant, nous pouvons régénérer le ggplot avec le jeu de données modifié :

```
ggplot(hiv_mort_modified) +
  geom_bar(aes(x = birth_month))
```



Les mois sur l'axe des x sont maintenant affichés dans l'ordre que nous avons spécifié.

La nouvelle variable de facteur respectera également l'ordre défini dans d'autres contextes. Par exemple, comparez comment la fonction `count()` affiche les deux tableaux de fréquences ci-dessous :

```
# Original dataset
count(hiv_mort, birth_month)
```

```
## # A tibble: 13 × 2
##    birth_month     n
##    <chr>       <int>
##  1 Apr            33
##  2 Aug            33
##  3 Dec            39
##  4 Feb            28
##  5 Jan            35
##  6 Jul            33
##  7 Jun            42
##  8 Mar            31
##  9 May            30
## 10 Nov            31
## 11 Oct            42
## 12 Sep            33
## 13 <NA>           35
```

```
# Modified dataset
count(hiv_mort_modified, birth_month)
```

```
## # A tibble: 13 × 2
##    birth_month     n
##    <fct>       <int>
##  1 Jan            35
##  2 Feb            28
##  3 Mar            31
##  4 Apr            33
##  5 May            30
##  6 Jun            42
##  7 Jul            33
##  8 Aug            33
##  9 Sep            33
## 10 Oct            42
## 11 Nov            31
## 12 Dec            39
## 13 <NA>           35
```

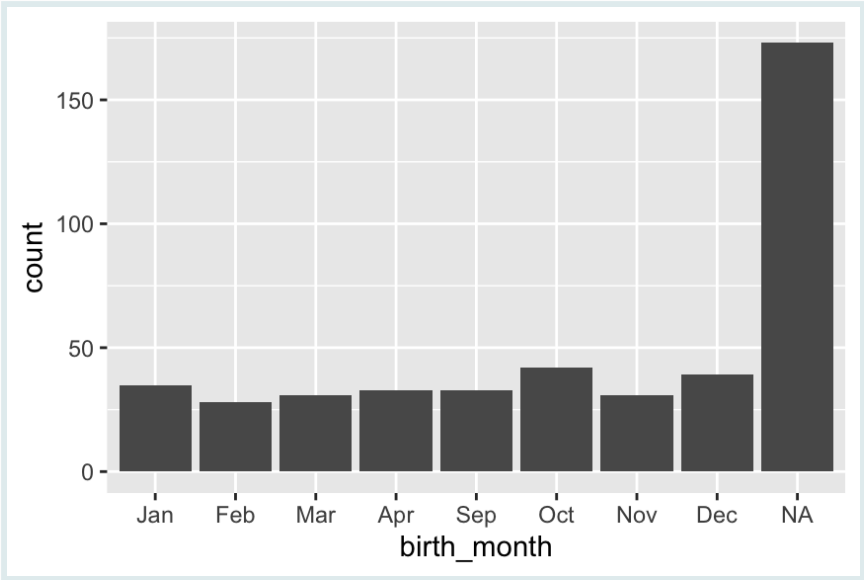
Soyez vigilant lorsque vous créez des niveaux de facteurs ! Toutes les valeurs de la variable **qui ne sont pas incluses** dans l'ensemble des niveaux fournis à l'argument `levels` seront converties en NA.

Par exemple, si nous avons manqué certains mois dans notre exemple :

```
hiv_mort_missing_months <-
  hiv_mort %>%
  mutate(birth_month = factor(x = birth_month,
                              levels = c("Jan", "Feb",
"Mar", "Apr",
                                         # missing
months
                                        "Sep", "Oct",
"Nov", "Dec")))
```

Nous finissons avec beaucoup de valeurs NA :

```
ggplot(hiv_mort_missing_months) +
  geom_bar(aes(x = birth_month))
```

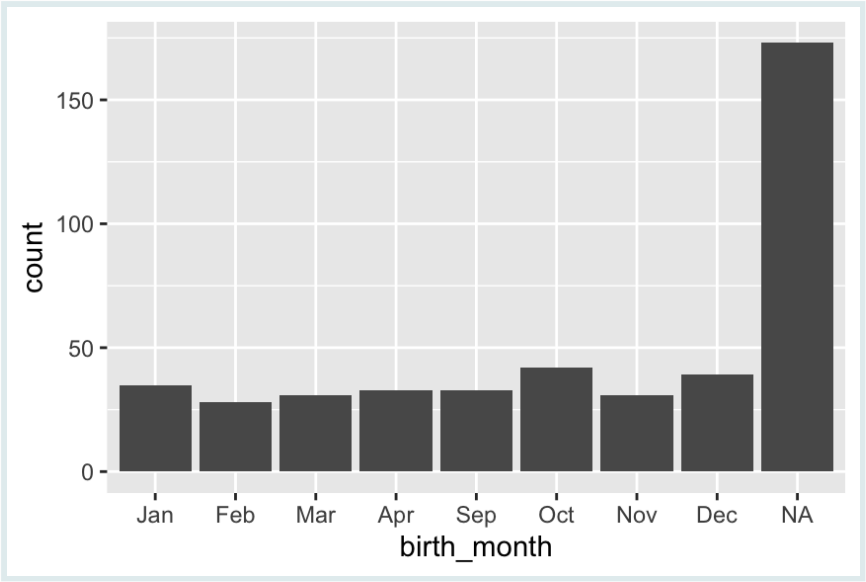


Vous aurez le même problème s'il y a des erreurs de frappe :

```
hiv_mort_with_typos <-
  hiv_mort %>%
  mutate(birth_month = factor(x = birth_month,
                              levels = c("Jan", "Feb",
"Mar", "Apr",
                                         "Moy", "Jon",
"Jol", "Aog", # typos
                                         "Sep", "Oct",
"Nov", "Dec")))
```

```
ggplot(hiv_mort_with_typos) +
  geom_bar(aes(x = birth_month))
```

WATCH OUT



SIDE NOTE

Vous pouvez utiliser le facteur sans niveaux. Il utilise simplement l'arrangement par défaut (alphabétique) des niveaux.

```
hiv_mort_default_factor <- hiv_mort %>%
  mutate(birth_month = factor(x = birth_month))
```

```
class(hiv_mort_default_factor$birth_month)
```

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

```
levels(hiv_mort_default_factor$birth_month)
```

```
##  [1] "Apr" "Aug" "Dec" "Feb" "Jan" "Jul" "Jun" "Mar" "May"
"Nov" "Oct" "Sep"
```

Q: Facteur de genre

En utilisant le jeu de données `hiv_mort`, convertissez la variable `gender` en un facteur avec les niveaux "Femme" et "Homme", dans cet ordre.

Q: Repérage des erreurs

Quelles erreurs pouvez-vous repérer dans l'extrait de code suivant ? Quelles sont les conséquences de ces erreurs ?

```
hiv_mort <-
  hiv_mort %>%
  mutate(birth_month = factor (x = birth_month,
                               levels = c("Jan", "Feb", "Mar", "Apr",
                                          "Mai", "Jun", "Jul", "Sep",
                                          "Oct", "Nov.", "Dec")))
```

Q: Avantage des facteurs

Quel est l'avantage principal de l'utilisation de facteurs par rapport aux caractères pour les données catégorielles dans R ?

a. Il est plus facile d'effectuer des manipulations de chaînes sur les facteurs.
b. Les facteurs permettent un meilleur contrôle de l'ordre des données catégorielles.
c. Les facteurs augmentent la précision des modèles statistiques.

Manipuler les facteurs avec `forcats`

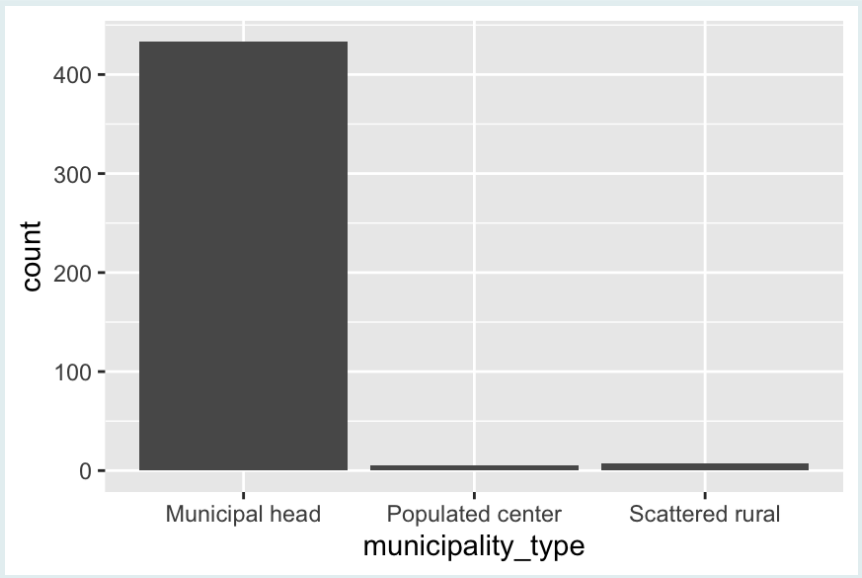
Les facteurs sont très utiles, mais ils peuvent parfois être un peu fastidieux à manipuler uniquement avec les fonctions de base R. Heureusement, le package `forcats`, membre du tidyverse, propose un ensemble de fonctions qui simplifient grandement la manipulation des facteurs. Nous allons examiner quatre fonctions ici, mais il y en a beaucoup d'autres, donc nous vous encourageons à explorer le site web de forcats par vous-même ici!

fct_relevel

La fonction `fct_relevel()` est utilisée pour changer manuellement l'ordre des niveaux de facteurs.

Par exemple, disons que nous voulons visualiser la fréquence des individus de notre jeu de données par type de municipalité. Lorsque nous créons un graphique en barres, les valeurs sont classées par ordre alphabétique par défaut :

```
ggplot(hiv_mort) +
  geom_bar(aes(x = municipality_type))
```



Mais que se passerait-il si nous voulions qu'une valeur spécifique, disons "Populated center", apparaisse en premier dans le graphique ?

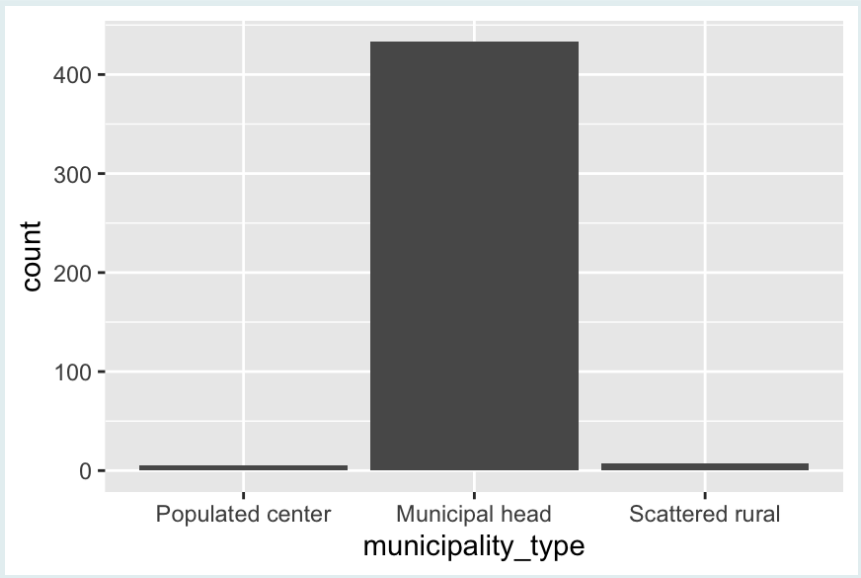
Cela peut être réalisé en utilisant `fct_relevel()`. Voici comment :

```
          hiv_mort_pop_center_first <-
            hiv_mort %>%
            mutate(municipality_type = fct_relevel(municipality_type,
 "Populated center"))
```

La syntaxe est simple : nous passons la variable factorielle en premier argument, et le niveau que nous voulons déplacer au début en second argument.

Maintenant lorsque nous traçons :

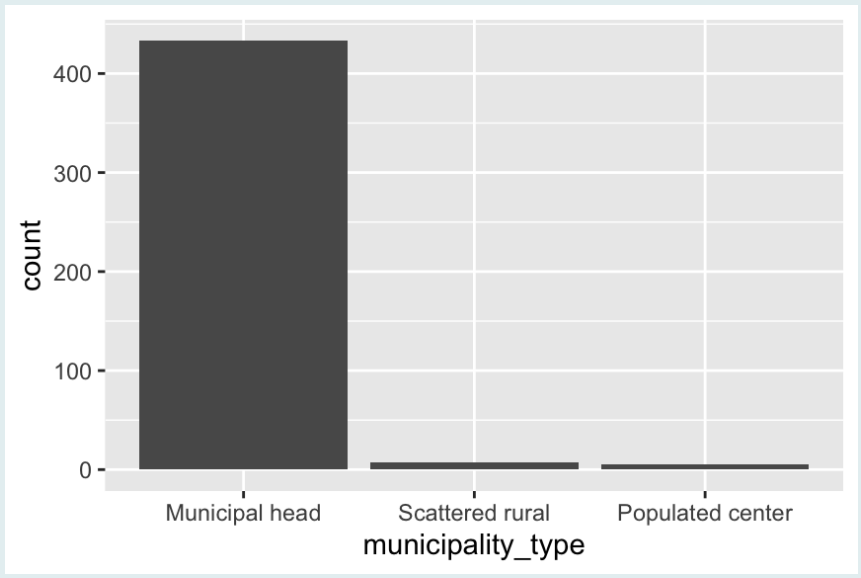
```
          ggplot(hiv_mort_pop_center_first) +
            geom_bar(aes(x = municipality_type))
```



Le niveau "Centre peuplé" est maintenant le premier.

Nous pouvons déplacer le niveau "Populated center" à une position différente avec l'argument `after` :

```
         hiv_mort %>%
           mutate(municipality_type = fct_relevel(municipality_type,
"Populated center",
                                                  after = 2)) %>%
          # pipe directly into to plot to visualize change
           ggplot() +
           geom_bar(aes(x = municipality_type))
```

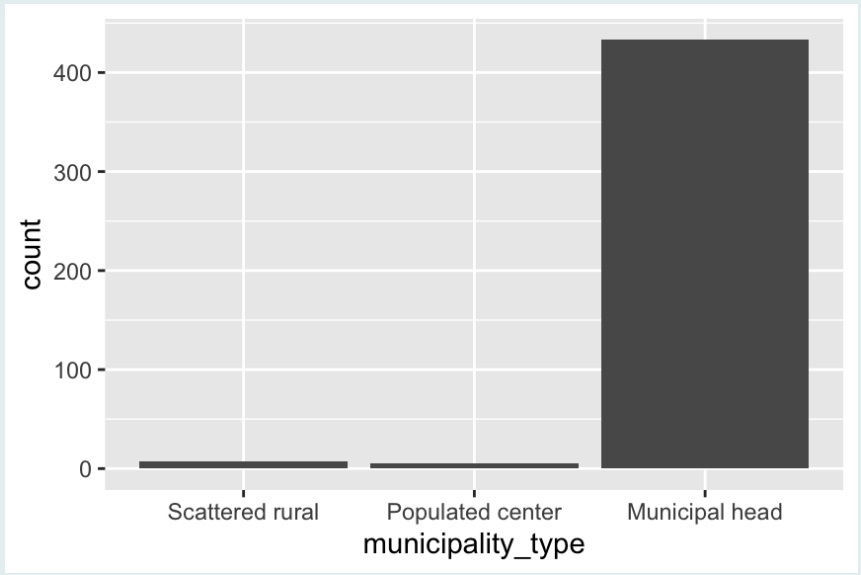


La syntaxe est : spécifier le facteur, le niveau à déplacer, et utiliser l'argument `after` pour définir à quelle position le placer après.

Nous pouvons également déplacer plusieurs niveaux à la fois en fournissant ces niveaux à `fct_relevel()` :

Ci-dessous, nous disposons tous les niveaux de facteurs pour le type de municipalité dans l'ordre souhaité :

```
hiv_mort %>%
  mutate(municipality_type = fct_relevel(municipality_type,
                                         "Scattered rural",
                                         "Populated center",
                                         "Municipal head")) %>%
  ggplot() +
  geom_bar(aes(x = municipality_type))
```



C'est similaire à la création d'un facteur depuis le début avec des niveaux dans cet ordre :

```
hiv_mort %>%
  mutate(municipality = factor(municipality_type,
                               levels = c("Scattered rural",
                                          "Populated center",
                                          "Municipal head")))
```

Q: Utiliser fct_relevel

En utilisant le jeu de données `hiv_mort`, convertissez la variable `death_location` en facteur de sorte que 'Home/address' soit le premier

niveau. Ensuite, créez un graphique en barres montrant le décompte des individus du jeu de données par `death_location`.

fct_reorder

`fct_reorder()` est utilisé pour réorganiser les niveaux d'un facteur en fonction des valeurs d'une autre variable.

Pour illustrer, créons un tableau récapitulatif avec le nombre de décès, l'âge moyen et médian au décès pour chaque municipalité :

```
summary_per_muni <-
  hiv_mort %>%
  group_by(municipality_name) %>%
  summarise(n_deceased = n(),
            mean_age_death = mean(age_at_death, na.rm = T),
            med_age_death = median(age_at_death, na.rm = T))

summary_per_muni
```

```
## # A tibble: 25 × 4
##    municipality_name n_deceased mean_age_death med_age_death
##    <chr>                  <int>          <dbl>         <dbl>
##  1 Aguadas                    2           42            42
##  2 Anserma                   15           37.4          37.5
##  3 Aranzazu                   2           37.5          37.5
##  4 Belalcázar                 4           38.8          41
##  5 Chinchiná                 62           43.6          42.5
##  6 Filadelfia                 5           42.6          43
##  7 La Dorada                 46           41.0          41
##  8 La Merced                  3           27            28
##  9 Manizales                199           41.0          41
## 10 Manzanares                 3           38.3          34
## # ℹ 15 more rows
```

Lorsque nous traçons l'une des variables, nous voudrons peut-être arranger les niveaux de facteurs par cette variable numérique. Par exemple, pour ordonner la municipalité par la colonne de l'âge moyen :

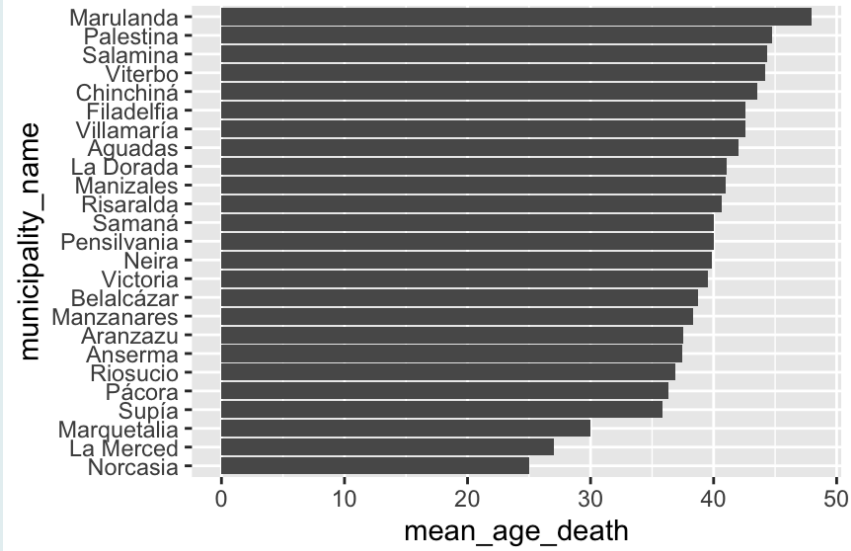
```
summary_per_muni_reordered <-
  summary_per_muni %>%
  mutate(municipality_name = fct_reorder(.f = municipality_name,
                                         .x = mean_age_death))
```

La syntaxe est :

- `.f` - le facteur à réorganiser
- `.x` - le vecteur numérique déterminant le nouvel ordre

Nous pouvons maintenant tracer un joli graphique en barres :

```
ggplot(summary_per_muni_reordered) +
  geom_col(aes(y = municipality_name, x = mean_age_death))
```



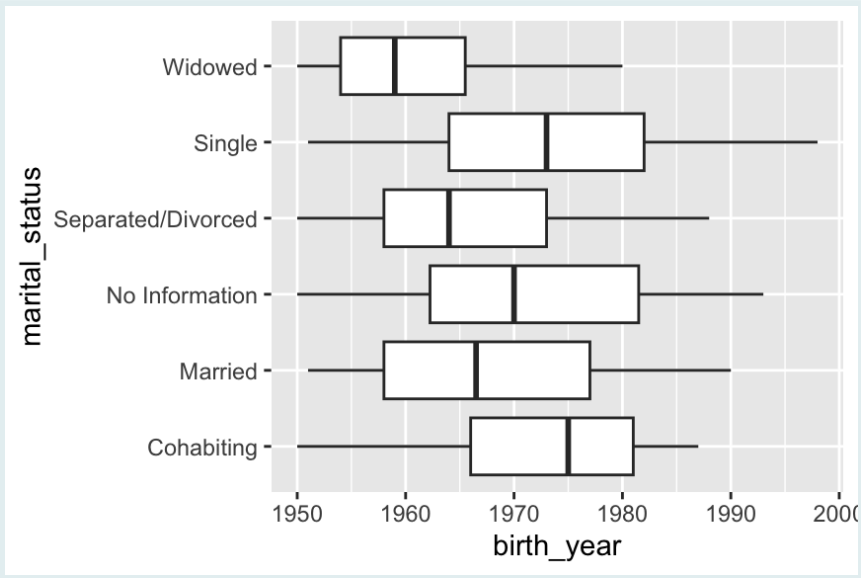
Q: Utiliser fct_reorder

En partant du dataframe `summary_per_muni`, réorganisez la municipalité (`municipality_name`) par la colonne `med_age_death` et tracez le graphique en barres réorganisé.

L'argument `.fun`

Parfois, nous voulons que les catégories de notre graphique apparaissent dans un ordre spécifique déterminé par une statistique sommaire. Par exemple, considérons le diagramme en boîte de `birth_year` par `marital_status` :

```
ggplot(hiv_mort, aes(y = marital_status, x = birth_year)) +
  geom_boxplot()
```



Le diagramme en boîte affiche la médiane `birth_year` pour chaque catégorie de marital status comme une ligne au milieu de chaque boîte. Nous voudrions peut-être arranger les catégories marital_status dans l'ordre de ces médianes. Mais si nous créons un tableau récapitulatif avec les médianes, comme nous l'avons fait précédemment avec `summary_per_muni`, nous ne pouvons pas créer de diagramme en boîte avec lui (allez regarder le dataframe `summary_per_muni` pour le vérifier vous-même).

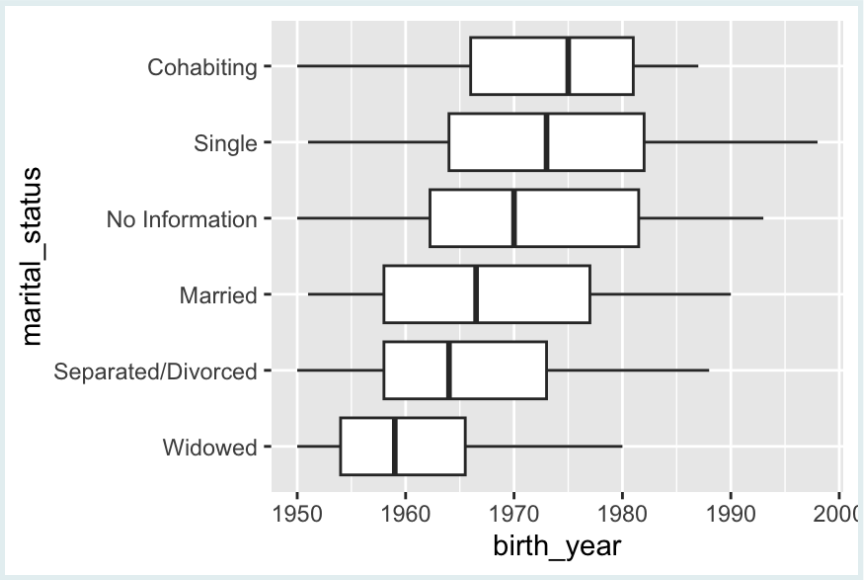
C'est là que intervient l'argument `.fun` de `fct_reorder()`. L'argument `.fun` nous permet de spécifier une fonction de résumé qui sera utilisée pour calculer le nouvel ordre des niveaux :

```
hiv_mort_arranged_marital <-
  hiv_mort %>%
  mutate(marital_status = fct_reorder(.f = marital_status,
                                      .x = birth_year,
                                      .fun = median,
                                      na.rm = TRUE))
```

Dans ce code, nous réorganisons le facteur `marital_status` en fonction de la médiane de `birth_year`. Nous incluons l'argument `na.rm = TRUE` pour ignorer les valeurs NA lors du calcul de la médiane.

Maintenant, lorsque nous créons notre diagramme en boîte, les catégories `marital_status` sont classées par la médiane `birth_year` :

```
ggplot(hiv_mort_arranged_marital, aes(y = marital_status, x =
birth_year)) +
  geom_boxplot()
```



Nous pouvons voir que les individus ayant le statut marital "cohabiting" ont tendance à être les plus jeunes (ils sont nés les dernières années).

Q: Utiliser .fun

En utilisant le jeu de données `hiv_mort`, faites un diagramme en boîte de `birth_year` par `health_insurance_status`, où les catégories `health_insurance_status` sont disposées par la médiane `birth_year`.

fct_recode

La fonction `fct_recode()` nous permet de modifier manuellement les valeurs des niveaux de facteurs. Cette fonction peut être particulièrement utile lorsque vous devez renommer des catégories ou lorsque vous souhaitez fusionner plusieurs catégories en une seule.

Par exemple, nous pouvons renommer 'Municipal head' en 'City' dans la variable `municipality_type` :

```
hiv_mort_muni_recode <- hiv_mort %>%
  mutate(municipality_type = fct_recode(municipality_type,
                                        "City" = "Municipal head"))

# View the change
levels(hiv_mort_muni_recode$municipality_type)
```

```
## [1] "City"            "Populated center" "Scattered rural"
```

Dans le code ci-dessus, `fct_recode()` prend deux arguments : la variable factorielle que vous souhaitez modifier (`municipality_type`) et l'ensemble des paires nom-valeur qui définissent le recodage. Le nouveau niveau ("City") est à gauche du signe égal et l'ancien niveau ("Municipal head") est à droite.

`fct_recode()` est particulièrement utile pour compresser plusieurs catégories en moins de niveaux.

Nous pouvons explorer cela en utilisant la variable education_level. Actuellement, elle possède six catégories :

```
        count(hiv_mort, education_level)
```

```
## # A tibble: 6 × 2
##   education_level     n
##   <chr>           <int>
## 1 No information     88
## 2 None               22
## 3 Post-secondary     29
## 4 Preschool           3
## 5 Primary           187
## 6 Secondary         116
```

Par souci de simplicité, regroupons-les en seulement trois catégories - "primary & below", "secondary & above" et "others":

```
        hiv_mort_educ_simple <-
          hiv_mort %>%
          mutate(education_level = fct_recode(education_level,
                                              "primary & below" = "Primary",
                                              "primary & below" =
"Preschool",
                                              "secondary & above" =
"Secondary",
                                              "secondary & above" = "Post-
secondary",
                                              "others" = "No information",
                                              "others" = "None"))
```

Cela condense joliment les catégories :

```
        count(hiv_mort_educ_simple, education_level)
```

```
## # A tibble: 3 × 2
##   education_level       n
##   <fct>             <int>
## 1 others              110
## 2 secondary & above   145
## 3 primary & below     190
```

Par mesure de précaution, nous pouvons arranger les niveaux dans un ordre raisonnable, avec "others" comme dernier niveau :

```
hiv_mort_educ_sorted <-
  hiv_mort_educ_simple %>%
  mutate(education_level = fct_relevel(education_level,
                                       "primary & below",
                                       "secondary & above",
                                       "others"))
```

Cela condense joliment les catégories :

```
count(hiv_mort_educ_sorted, education_level)
```

```
## # A tibble: 3 × 2
##   education_level       n
##   <fct>             <int>
## 1 primary & below     190
## 2 secondary & above   145
## 3 others              110
```

Q: Utiliser fct_recode

En utilisant le jeu de données `hiv_mort`, convertissez death_location en facteur.

Ensuite, utilisez `fct_recode()` pour renommer 'Public way' dans death_location en 'Public place'. Tracez les fréquences de la variable mise à jour.

fct_recode vs case_when/if_else

Vous vous demandez peut-être pourquoi nous avons besoin de `fct_recode()` alors que nous pouvons utiliser `case_when()` ou `if_else()` voire même `recode()` pour substituer des valeurs spécifiques. Le problème est que ces autres fonctions peuvent perturber votre variable factorielle.

Pour illustrer, disons que nous choisissons d'utiliser `case_when()` pour apporter une modification à la variable `education_level` du dataframe `hiv_mort_educ_sorted`.

Pour rappel, cette variable est un facteur avec trois niveaux, arrangés dans un ordre spécifié, avec "primary & below" en premier et "others" en dernier :

```
count(hiv_mort_educ_sorted, education_level)
```

```
## # A tibble: 3 × 2
##   education_level       n
##   <fct>             <int>
## 1 primary & below     190
## 2 secondary & above   145
## 3 others              110
```

SIDE NOTE

Disons que nous voulions remplacer "others" par "other", en enlevant le "s". Nous pouvons écrire :

```
hiv_mort_educ_other <-
  hiv_mort_educ_sorted %>%
  mutate(education_level = if_else(education_level ==
"others",
                                   "other",
education_level))
```

Après cette opération, la variable n'est plus un facteur :

```
class(hiv_mort_educ_other$education_level)
```

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

niveau :

```
count(hiv_mort_educ_other, education_level)
```

```
## # A tibble: 3 × 2
##   education_level       n
##   <chr>             <int>
## 1 other               110
## 2 primary & below     190
## 3 secondary & above   145
```

Cependant, si nous avions utilisé `fct_recode()` pour le recodage, nous n'aurions pas ce problème :

```
hiv_mort_educ_other_fct <-
  hiv_mort_educ_simple %>%
  mutate(education_level =
fct_recode(education_level, "other" = "others"))
```

SIDE NOTE

La variable reste un facteur :

```
class(hiv_mort_educ_other_fct$education_level)
```

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

Et si nous créons un tableau ou un graphique, notre ordre est préservé : "primary", "secondary", puis "other":

```
count(hiv_mort_educ_other_fct, education_level)
```

```
## # A tibble: 3 × 2
##   education_level       n
##   <fct>             <int>
## 1 other               110
## 2 secondary & above   145
## 3 primary & below     190
```

fct_lump

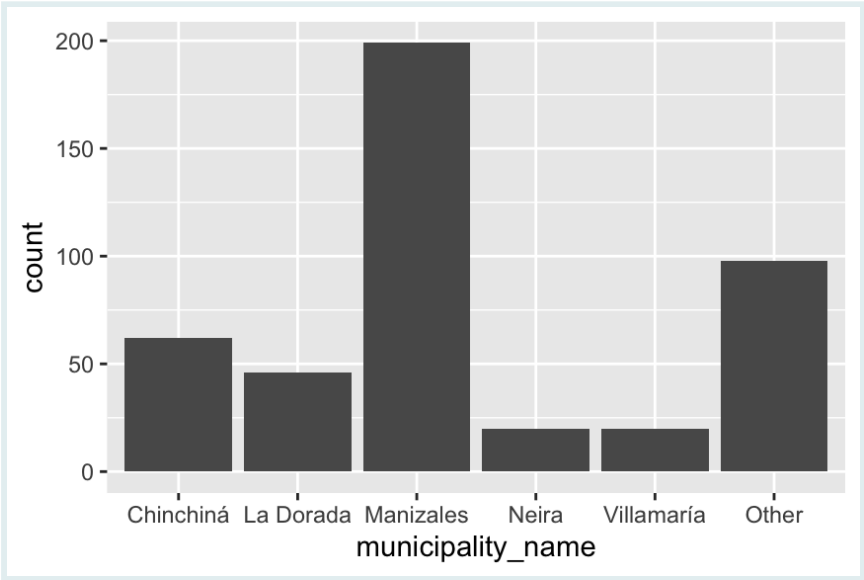
Parfois, nous avons trop de niveaux pour un tableau d'affichage ou un graphique, et nous voulons regrouper les niveaux les moins fréquents dans une seule catégorie, généralement appelée 'Other'.

C'est là que la fonction pratique `fct_lump()` intervient.

Dans l'exemple ci-dessous, nous regroupons les municipalités les moins fréquentes dans 'Other', en ne conservant que les 5 municipalités les plus fréquentes :

```
hiv_mort_lump_muni <- hiv_mort %>%
  mutate(municipality_name = fct_lump(municipality_name, n = 5))

ggplot(hiv_mort_lump_muni, aes(x = municipality_name)) +
  geom_bar()
```

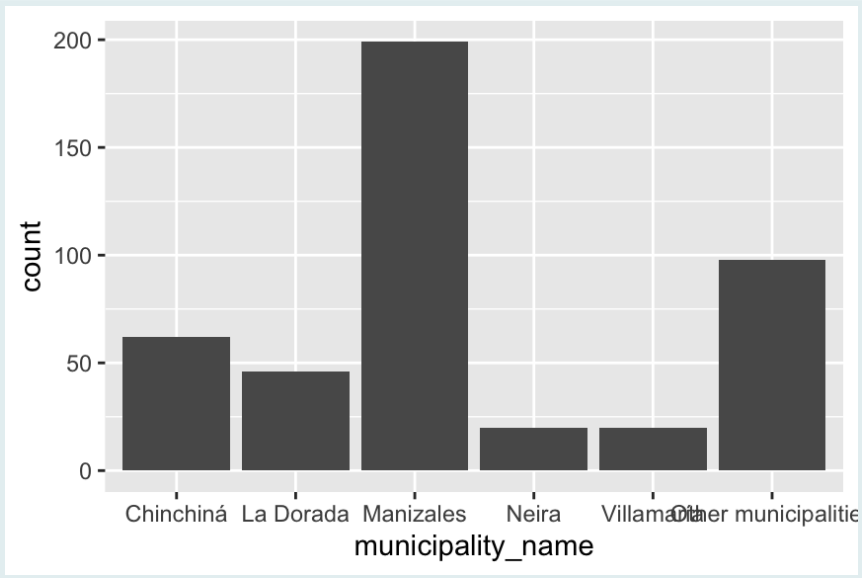


Dans l'utilisation ci-dessus, le paramètre `n = 5`signifie que les cinq municipalités les plus fréquentes sont conservées, et le reste est regroupé dans 'Other'.

Nous pouvons fournir un nom personnalisé pour l'autre catégorie avec l'argument `other_level`. Ci-dessous, nous utilisons le nom `"Other municipalities"`.

```
hiv_mort_lump_muni_other_name <- hiv_mort %>%
  mutate(municipality_name = fct_lump(municipality_name, n = 5,
                                      other_level = "Other
municipalities"))

ggplot(hiv_mort_lump_muni_other_name, aes(x = municipality_name)) +
  geom_bar()
```



De cette façon, `fct_lump()` est un outil pratique pour condenser les facteurs avec de nombreux niveaux peu fréquents en un nombre plus gérable de catégories.

Q: Utiliser fct_lump

En partant du jeu de données `hiv_mort`, utilisez `fct_lump()` pour créer un diagramme en barres avec la fréquence des 10 occupations les plus courantes.

Regroupez les occupations restantes dans une catégorie 'Other'.

Mettez `occupation` sur l'axe des y, et non sur l'axe des x, pour éviter le chevauchement des étiquettes.

Conclusion

Félicitations d'être arrivé jusqu'au bout. Dans cette leçon, vous avez appris les détails sur la classe de données, **les facteurs**, et comment les manipuler en utilisant des opérations de base comme `fct_relevel()`, `fct_reorder()`, `fct_recode()` et `fct_lump()`.

Bien qu'elles couvrent des tâches courantes comme le réordonnancement, le recodage et la fusion de niveaux, cette introduction ne fait qu'effleurer la surface de ce qui est

possible avec le package forcats. N'hésitez pas à explorer davantage sur le site web forcats website.

Maintenant que vous comprenez les bases du travail avec les facteurs, vous êtes équipé pour représenter correctement vos données catégorielles dans R pour l'analyse et la visualisation en aval.

Corrigé

Q: Facteur de genre

```
hiv_mort_q1 <- hiv_mort %>%
  mutate(gender = factor(x = gender,
                         levels = c("Female", "Male")))
```

Q: Repérage des erreurs

Erreurs : - "Mai" devrait être "May". - "Nov." a un point en trop. - "Aug" est manquant dans la liste des mois.

Conséquences :

Toutes les lignes avec les valeurs "May", "Nov" ou "Aug" pour death_month seront converties en NA dans la nouvelle variable death_month. Si vous créez des graphiques, ggplot supprimera ces niveaux avec seulement des valeurs NA.

Q: Avantage des facteurs

b. Les facteurs permettent un meilleur contrôle de l'ordre des données catégorielles.

Les deux autres déclarations ne sont pas vraies.

Si vous voulez appliquer des opérations sur les chaînes de caractères comme substr(), strsplit(), paste(), etc., il est en fait plus simple d'utiliser des vecteurs de caractères que des facteurs.

Et bien que de nombreuses fonctions statistiques attendent des facteurs, et non des caractères, pour les prédicteurs catégoriels, cela ne les rend pas plus "précises".

Q: Utiliser fct_relevel

Q: Utiliser fct_reorder

Q: Utiliser .fun

Q: Utiliser fct_recode

Q: Utiliser fct_lump

Annexe : Codebook

Les variables du jeu de données sont :

- `municipality` : localisation municipale générale du patient [chr]
- `death_location` : lieu où le patient est décédé [chr]
- `birth_date` : date de naissance complète, formatée “YYYY-MM-DD” [date]
- `birth_year` : année de naissance du patient [dbl]
- `birth_month` : mois de naissance du patient [chr]
- `birth_day` : jour de naissance du patient [dbl]
- `death_year` : année de décès du patient [dbl]
- `death_month` : mois de décès du patient [chr]
- `death_day` : jour de décès du patient [dbl]
- `gender` : genre du patient [chr]
- `education_level` : plus haut niveau d’études atteint par le patient [chr]
- `occupation` : profession du patient [chr]
- `racial_id` : race du patient [chr]
- `municipality_code` : localisation municipale spécifique du patient [chr]
- `primary_cause_death_description` : cause primaire de décès du patient [chr]
- `primary_cause_death_code` : code de la cause primaire de décès [chr]
- `secondary_cause_death_description` : cause secondaire de décès du patient [chr]
- `secondary_cause_death_code` : code de la cause secondaire de décès [chr]
- `tertiary_cause_death_description` : cause tertiaire de décès du patient [chr]
- `tertiary_cause_death_code` : code de la cause tertiaire de décès [chr]
- `quaternary_cause_death_description` : cause quaternaire de décès du patient [chr]
- `quaternary_cause_death_code` : code de la cause quaternaire de décès [chr]

Contributeurs

Les membres de l’équipe suivants ont contribué à cette leçon :

CAMILLE BEATRICE VALERA

Project Manager and Scientific Collaborator, The GRAPH Network

KENE DAVID NWOSU

Data analyst, the GRAPH Network
Passionate about world improvement