Transformer des données dans R avec **dplyr**

Fousseynou Bah

29-Mar-2019

Introduction

Objectif de ce chapitre

Le *data scientist* a très rarement les données structurées dans la forme qui lui convient. Il lui revient de les mettres dans cette forme. De ce fait, il lui est indispensable de savoir manipuler les données. Il s'agit de la maîtrise d'une série de tâches parmi lesquelles nous pouvons citer:

- la simple sélection d'un sous-ensemble à l'intérieur d'un large groupe ;
- la sélection d'un nombre déterminé de variables (attributs) ;
- la combinaison d'informations conservées dans différentes `data frame`;
- la suppresion et la création variables;
- la réorganisation des données à l'intérieur d'un `data frame`.

Dans cette présentation, nous allons voir quelques techniques de manipulation de données avec le package **dplyr**.

Installer **dplyr**

Le package **dplyr** est un package créé par Hadley Wickham. Il fait partie de l'écosystème **tidyverse** qui est un ensemble de packages conçus pour la *data science* et partageant tous les mêmes philoshophie, grammaire et structure. Nous allons, avec l'exploration des packages du **tidyverse**, comprendre l'importance de cette grammaire. Elle traduit la volonté des auteurs de condenser dans le nom des fonctions l'idée de la tâche que celles-ci exécutent. Dans **dplyr**, les fonctions majeures sont des verbes (comme nous allons le voir).

Alors, qu'est-ce que **dplyr** fait? Voici une simple analogie. Si les données constituent un tissu, nous pouvons voir **dplyr** comme à la fois la paire de ciseaux pour en faire la coupe, et l'aiguille et le fil pour le coudre. Pour le prouver, voici le logo associé au package.

Installer **dplyr**

```
# Installer tidyverse
install.packages("tidyverse")
# ou dplyr tout seul, sa version stable depuis CRAN
install.packages("dplyr")
# ou installer la version en développement
# au cas où devtools n'est pas installé
install.packages("devtools")
# ensuite
devtools::install_github("tidyverse/dplyr")
# Une fois l'installation effectuée,
# charger le package.
library(dplyr)
```

Données

Les données utilisées sont accessibles à partir de cette adresse.

```
# Données
source_donnees <- paste0("https://raw.githubusercontent.com/fousseynoubah/dswr_slides/master/",
                         "5_Transformer_Donnees_dans_R_avec_dplyr/data/adm3_pop_2009.csv")
# Chargement du package "readr"
library(readr)
# Importation des données
adm3_pop_2009 <- read_csv(source_donnees)
# Classe
class(adm3_pop_2009)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```
# Aperçu
```

```
## Observations: 703
## Variables: 9
## $ id         <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...
## $ admin0_nom <chr> "Mali", "Mali", "Mali", "Mali", "Mali", "Mali", "Ma...
## $ admin1_nom <chr> "Kayes", "Kayes", "Kayes", "Kayes", "Kayes", "Kayes...
## $ admin2_nom <chr> "Kayes", "Kayes", "Kayes", "Kayes", "Kayes", "Kayes...
## $ admin3_nom <chr> "Bangassi", "Colimbine", "Diamou", "Djelebou", "Fal...
## $ annee      <dbl> 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 200...
## $ homme      <dbl> 6123, 6144, 7115, 11466, 5141, 1999, 3939, 1918, 97...
## $ femme      <dbl> 5974, 6353, 7015, 12091, 5017, 1934, 3927, 1903, 10...
## $ source     <chr> "RGPH", "RGPH", "RGPH", "RGPH", "RGPH", "RGPH", "RG...
```

Selection et/ou exclusion de variables: `select`

Aperçu des données

Il arrive souvent que l'on veuille sélectionner certaines variables d'un jeu de données. Dans notre cas, supposons que nous sommes seulement intéressés par les chiffres. Il s'agit des colonnes: `homme` et `femme`.

Rappelons notre jeu de données.

```
# Chargement du package "readr"
library(readr)
# Importation des données
adm3_pop_2009 <- read_csv(source_donnees)
# Impression
adm3_pop_2009
```

```
## # A tibble: 703 x 9
##       id admin0_nom admin1_nom admin2_nom admin3_nom annee homme femme
##    <dbl> <chr>      <chr>      <chr>      <chr>      <dbl> <dbl> <dbl>
##  1     1 Mali       Kayes      Kayes      Bangassi    2009  6123  5974
##  2     2 Mali       Kayes      Kayes      Colimbine   2009  6144  6353
##  3     3 Mali       Kayes      Kayes      Diamou      2009  7115  7015
##  4     4 Mali       Kayes      Kayes      Djelebou    2009 11466 12091
##  5     5 Mali       Kayes      Kayes      Faleme      2009  5141  5017
##  6     6 Mali       Kayes      Kayes      Fegui       2009  1999  1934
##  7     7 Mali       Kayes      Kayes      Gory Gope~  2009  3939  3927
##  8     8 Mali       Kayes      Kayes      Goumera     2009  1918  1903
##  9     9 Mali       Kayes      Kayes      Guidimaka~  2009  9798 10234
## 10    10 Mali       Kayes      Kayes      Hawa Demb~  2009  3406  3445
## # ... with 693 more rows, and 1 more variable: source <chr>
```

Sélection simple avec R-base

```
adm3_pop_2009[ , c("homme", "femme")]
```

```
## # A tibble: 703 x 2
##    homme femme
##    <dbl> <dbl>
##  1  6123  5974
##  2  6144  6353
##  3  7115  7015
##  4 11466 12091
##  5  5141  5017
##  6  1999  1934
##  7  3939  3927
##  8  1918  1903
##  9  9798 10234
## 10  3406  3445
## # ... with 693 more rows
```

```
# ou
subset(x = adm3_pop_2009, select = c(homme, femme))
```

```
## # A tibble: 703 x 2
##    homme femme
##    <dbl> <dbl>
##  1  6123  5974
##  2  6144  6353
##  3  7115  7015
##  4 11466 12091
##  5  5141  5017
##  6  1999  1934
```

Sélection simple avec select

```
select(adm3_pop_2009, homme, femme)
```

```
## # A tibble: 703 x 2
##    homme femme
##    <dbl> <dbl>
##  1  6123  5974
##  2  6144  6353
##  3  7115  7015
##  4 11466 12091
##  5  5141  5017
##  6  1999  1934
##  7  3939  3927
##  8  1918  1903
##  9  9798 10234
## 10  3406  3445
## # ... with 693 more rows
```

- Simplicité par rapport aux solution de R-base
- Pas besoin de " " comme avec mes crochets
- Pas besoin de préciser le nombre d'observation à imprimer
- Affichage optimisées quand l'entrant un `tibble`: fonction `R-base` ou fonction **tidyverse**
- Quand l'entrant n'est pas un `tibble`, affichage optimisé par les fonctions **tidyverse**, mais pas par les fonctions R-base

L'opérateur %>%

```
adm3_pop_2009 %>%
  select(homme, femme)
```

```
## # A tibble: 703 x 2
##    homme femme
##    <dbl> <dbl>
##  1  6123  5974
##  2  6144  6353
##  3  7115  7015
##  4 11466 12091
##  5  5141  5017
##  6  1999  1934
##  7  3939  3927
##  8  1918  1903
##  9  9798 10234
## 10  3406  3445
## # ... with 693 more rows
```

- Verbalisation de l'idée de la chaîne
- Omission de l'argument de l'intrant (jeu de données) car déjà pris en charge

Sélection groupée: avec R-base

Un autre avantage de select est de permettre la sélection de plusieurs variables à travers leur communalité. Dans notre jeu, nous avons trois variables qui portent toutes le terme admin comme préfixe. Plutôt que de les sélectionner une à une nous pouvons les appeler toutes ensembles. Regardons avec R-base.

```
names_df <- names(adm3_pop_2009)
select_df <- startsWith(x = names_df, prefix = "adm")
adm3_pop_2009[, select_df]
```

```
## # A tibble: 703 x 4
##    admin0_nom admin1_nom admin2_nom admin3_nom
##    <chr>      <chr>      <chr>      <chr>
##  1 Mali       Kayes      Kayes      Bangassi
##  2 Mali       Kayes      Kayes      Colimbine
##  3 Mali       Kayes      Kayes      Diamou
##  4 Mali       Kayes      Kayes      Djelebou
##  5 Mali       Kayes      Kayes      Faleme
##  6 Mali       Kayes      Kayes      Fegui
##  7 Mali       Kayes      Kayes      Gory Gopela
##  8 Mali       Kayes      Kayes      Goumera
##  9 Mali       Kayes      Kayes      Guidimakan Keri Kaff
## 10 Mali       Kayes      Kayes      Hawa Dembaya
## # ... with 693 more rows
```

Sélection groupée: avec select

```
adm3_pop_2009 %>%
  select(starts_with("adm"))
```

```
## # A tibble: 703 x 4
##    admin0_nom admin1_nom admin2_nom admin3_nom
##    <chr>      <chr>      <chr>      <chr>
##  1 Mali       Kayes      Kayes      Bangassi
##  2 Mali       Kayes      Kayes      Colimbine
##  3 Mali       Kayes      Kayes      Diamou
##  4 Mali       Kayes      Kayes      Djelebou
##  5 Mali       Kayes      Kayes      Faleme
##  6 Mali       Kayes      Kayes      Fegui
##  7 Mali       Kayes      Kayes      Gory Gopela
##  8 Mali       Kayes      Kayes      Goumera
##  9 Mali       Kayes      Kayes      Guidimakan Keri Kaff
## 10 Mali       Kayes      Kayes      Hawa Dembaya
## # ... with 693 more rows
```

- Voir aussi avec les fonctions: `ends_with`, `contains`, `matches()`, `num_range()`, `one_of()`, `everything()`, etc.

Index et noms: avec R-base (1)

La sélection peut aussi se faire sur la base de l'index des colonnes, c'est-à-dire leur position. Sélectionnons de la deuxième à la cinquième colonne.

```
adm3_pop_2009[, c(2:5)]
```

```
## # A tibble: 703 x 4
##    admin0_nom admin1_nom admin2_nom admin3_nom
##    <chr>      <chr>      <chr>      <chr>
##  1 Mali       Kayes      Kayes      Bangassi
##  2 Mali       Kayes      Kayes      Colimbine
##  3 Mali       Kayes      Kayes      Diamou
##  4 Mali       Kayes      Kayes      Djelebou
##  5 Mali       Kayes      Kayes      Faleme
##  6 Mali       Kayes      Kayes      Fegui
##  7 Mali       Kayes      Kayes      Gory Gopela
##  8 Mali       Kayes      Kayes      Goumera
##  9 Mali       Kayes      Kayes      Guidimakan Keri Kaff
## 10 Mali       Kayes      Kayes      Hawa Dembaya
## # ... with 693 more rows
```

Index et noms: avec select (1)

```
adm3_pop_2009 %>%
  select(2:5)
```

```
## # A tibble: 703 x 4
##    admin0_nom admin1_nom admin2_nom admin3_nom
##    <chr>      <chr>      <chr>      <chr>
##  1 Mali       Kayes      Kayes      Bangassi
##  2 Mali       Kayes      Kayes      Colimbine
##  3 Mali       Kayes      Kayes      Diamou
##  4 Mali       Kayes      Kayes      Djelebou
##  5 Mali       Kayes      Kayes      Faleme
##  6 Mali       Kayes      Kayes      Fegui
##  7 Mali       Kayes      Kayes      Gory Gopela
##  8 Mali       Kayes      Kayes      Goumera
##  9 Mali       Kayes      Kayes      Guidimakan Keri Kaff
## 10 Mali       Kayes      Kayes      Hawa Dembaya
## # ... with 693 more rows
```

Index et noms: avec R-base (2)

La même chose peut se faire avec le nom des variables.

```
adm3_pop_2009[, c("admin0_nom", "admin1_nom", "admin2_nom", "admin3_nom")]
```

```
## # A tibble: 703 x 4
##    admin0_nom admin1_nom admin2_nom admin3_nom
##    <chr>      <chr>      <chr>      <chr>
##  1 Mali       Kayes      Kayes      Bangassi
##  2 Mali       Kayes      Kayes      Colimbine
##  3 Mali       Kayes      Kayes      Diamou
##  4 Mali       Kayes      Kayes      Djelebou
##  5 Mali       Kayes      Kayes      Faleme
##  6 Mali       Kayes      Kayes      Fegui
##  7 Mali       Kayes      Kayes      Gory Gopela
##  8 Mali       Kayes      Kayes      Goumera
##  9 Mali       Kayes      Kayes      Guidimakan Keri Kaff
## 10 Mali       Kayes      Kayes      Hawa Dembaya
## # ... with 693 more rows
```

Index et noms: avec select (2)

```
adm3_pop_2009 %>%
  select(admin0_nom, admin1_nom, admin2_nom, admin3_nom)
```

```
## # A tibble: 703 x 4
##    admin0_nom admin1_nom admin2_nom admin3_nom
##    <chr>      <chr>      <chr>      <chr>
##  1 Mali       Kayes      Kayes      Bangassi
##  2 Mali       Kayes      Kayes      Colimbine
##  3 Mali       Kayes      Kayes      Diamou
##  4 Mali       Kayes      Kayes      Djelebou
##  5 Mali       Kayes      Kayes      Faleme
##  6 Mali       Kayes      Kayes      Fegui
##  7 Mali       Kayes      Kayes      Gory Gopela
##  8 Mali       Kayes      Kayes      Goumera
##  9 Mali       Kayes      Kayes      Guidimakan Keri Kaff
## 10 Mali       Kayes      Kayes      Hawa Dembaya
## # ... with 693 more rows
```

Exclusion: avec R-base

Comme avec R-base, la sélection peut aussi se faire sur la base de l'exclusion. Imaginez que l'on souhaite exclure les quatre premières variables.

```
adm3_pop_2009[, -c(1:4)]
```

```
## # A tibble: 703 x 5
##    admin3_nom           annee homme femme source
##    <chr>                <dbl> <dbl> <dbl> <chr>
##  1 Bangassi              2009  6123  5974 RGPH
##  2 Colimbine             2009  6144  6353 RGPH
##  3 Diamou                2009  7115  7015 RGPH
##  4 Djelebou              2009 11466 12091 RGPH
##  5 Faleme                2009  5141  5017 RGPH
##  6 Fegui                 2009  1999  1934 RGPH
##  7 Gory Gopela           2009  3939  3927 RGPH
##  8 Goumera               2009  1918  1903 RGPH
##  9 Guidimakan Keri Kaff  2009  9798 10234 RGPH
## 10 Hawa Dembaya          2009  3406  3445 RGPH
## # ... with 693 more rows
```

Exclusion: avec select

```
adm3_pop_2009 %>%
  select(-c(1:4))
```

```
## # A tibble: 703 x 5
##    admin3_nom           annee homme femme source
##    <chr>                <dbl> <dbl> <dbl> <chr>
##  1 Bangassi              2009  6123  5974 RGPH
##  2 Colimbine             2009  6144  6353 RGPH
##  3 Diamou                2009  7115  7015 RGPH
##  4 Djelebou              2009 11466 12091 RGPH
##  5 Faleme                2009  5141  5017 RGPH
##  6 Fegui                 2009  1999  1934 RGPH
##  7 Gory Gopela           2009  3939  3927 RGPH
##  8 Goumera               2009  1918  1903 RGPH
##  9 Guidimakan Keri Kaff  2009  9798 10234 RGPH
## 10 Hawa Dembaya          2009  3406  3445 RGPH
## # ... with 693 more rows
```

Renommer des variables: avec R-base

dplyr compte une fonction `rename` qui permet de renommer les variables. Prise indépendemment, elle agit comme `select`. Supposons que nous voulions changer les noms de variables en majuscules.

```
# Sauvegarde des données dans un nouveau data frame
pop_df <- adm3_pop_2009[ , c("homme", "femme")]
# Voir les noms
names(pop_df)
```

```
## [1] "homme" "femme"
```

```
# Changer les noms
names(pop_df) <- c("HOMME", "FEMME")
# Vérification
names(pop_df)
```

```
## [1] "HOMME" "FEMME"
```

Renommer des variables: avec select (1)

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(homme, femme) %>%
  # Modification des noms de variables
  rename(HOMME = homme, FEMME = femme)
```

```
## # A tibble: 703 x 2
##    HOMME FEMME
##    <dbl> <dbl>
##  1  6123  5974
##  2  6144  6353
##  3  7115  7015
##  4 11466 12091
##  5  5141  5017
##  6  1999  1934
##  7  3939  3927
##  8  1918  1903
##  9  9798 10234
## 10  3406  3445
## # ... with 693 more rows
```

Renommer des variables: avec `select` (2)

Nous voyons qu'une telle opération qu'avec R-base requiert la création d'un objet intermédiaire tandis qu'avec **dplyr** elle s'insère tout simplement dans la séquence. Mais `select` peut elle-même prendre en charge la tâche de changement de noms.

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt et changement de noms
  select(HOMME = homme, FEMME = femme)
```

```
## # A tibble: 703 x 2
##    HOMME FEMME
##    <dbl> <dbl>
##  1  6123  5974
##  2  6144  6353
##  3  7115  7015
##  4 11466 12091
##  5  5141  5017
##  6  1999  1934
##  7  3939  3927
##  8  1918  1903
##  9  9798 10234
## 10  3406  3445
## # ... with 693 more rows
```

L'on indique le nouveau nom suivi du signe = et ensuite le nouveau nom.

Création et/ou suppression de variables: mutate

Création de variable: avec R-base (1)

La sélection et la suppression de variables peuvent s'inscrire dans le cadre d'une stratégie d'exploration plus large qui peut elle-même impliquer la création de nouvelles variables.

Explorons la supériorité numérique entre hommes et femmes. Supposons que nous souhaitions:

- sélectionner les populations pour les hommes et les femmes pour chaque commune;
- calculer un ratio femme/homme pour chaque commune.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable
pop_df$ratio <- pop_df$femme / pop_df$homme
# Aperçu
pop_df
```

```
## # A tibble: 703 x 4
##    admin3_nom           homme femme ratio
##    <chr>                <dbl> <dbl> <dbl>
##  1 Bangassi              6123  5974 0.976
##  2 Colimbine             6144  6353 1.03
##  3 Diamou                7115  7015 0.986
##  4 Djelebou             11466 12091 1.05
##  5 Faleme                5141  5017 0.976
##  6 Fegui                 1999  1934 0.967
##  7 Gory Gopela           3939  3927 0.997
##  8 Goumera               1918  1903 0.992
##  9 Guidimakan Keri Kaff  9798 10234 1.04
## 10 Hawa Dembaya          3406  3445 1.01
## # ... with 693 more rows
```

Création de variable: avec mutate (1)

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(ratio = homme/femme)
```

```
## # A tibble: 703 x 4
##    admin3_nom           homme femme ratio
##    <chr>                <dbl> <dbl> <dbl>
##  1 Bangassi              6123  5974 1.02
##  2 Colimbine             6144  6353 0.967
##  3 Diamou                7115  7015 1.01
##  4 Djelebou             11466 12091 0.948
##  5 Faleme                5141  5017 1.02
##  6 Fegui                 1999  1934 1.03
##  7 Gory Gopela           3939  3927 1.00
##  8 Goumera               1918  1903 1.01
##  9 Guidimakan Keri Kaff  9798 10234 0.957
## 10 Hawa Dembaya          3406  3445 0.989
## # ... with 693 more rows
```

Le résultat est le même, mais le gain avec **dplyr** est visible. Le séquençage rend la lecture du code facile. Il évite aussi la création d'un objet intermédiaire, comme ce fut le cas de `pop_df` avec R-base.

Création de variable: avec R-base (2)

Dans le cas précédent, la variable était numérique. Elle peut aussi prendre la forme catégorielle. Considérons par exemple des intervalles de population qu'on souhaiterait créer pour séparer les communes en catégories. Nous allons d'abord faire la somme des deux groupes, hommes et femmes, et ensuite créer la variable catégorielle.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Création d'une nouvelle variable: catégorielle
pop_df$pop_cat <- ifelse(pop_df$total < 10000, "<10000", ">=10000")
# Aperçu
pop_df
```

```
## # A tibble: 703 x 5
##    admin3_nom           homme femme total pop_cat
##    <chr>                <dbl> <dbl> <dbl> <chr>
##  1 Bangassi              6123  5974 12097 >=10000
##  2 Colimbine             6144  6353 12497 >=10000
##  3 Diamou                7115  7015 14130 >=10000
##  4 Djelebou             11466 12091 23557 >=10000
##  5 Faleme                5141  5017 10158 >=10000
##  6 Fegui                 1999  1934  3933 <10000
##  7 Gory Gopela           3939  3927  7866 <10000
##  8 Goumera               1918  1903  3821 <10000
##  9 Guidimakan Keri Kaff  9798 10234 20032 >=10000
## 10 Hawa Dembaya          3406  3445  6851 <10000
## # ... with 693 more rows
```

Création de variable: avec mutate (2)

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable: numérique
  mutate(total = homme + femme) %>%
  # Création d'une nouvelle variable: catégorielle
  mutate(pop_cat = ifelse(total < 10000, "<10000", ">=10000"))
```

```
## # A tibble: 703 x 5
##    admin3_nom           homme femme total pop_cat
##    <chr>                <dbl> <dbl> <dbl> <chr>
##  1 Bangassi              6123  5974 12097 >=10000
##  2 Colimbine             6144  6353 12497 >=10000
##  3 Diamou                7115  7015 14130 >=10000
##  4 Djelebou             11466 12091 23557 >=10000
##  5 Faleme                5141  5017 10158 >=10000
##  6 Fegui                 1999  1934  3933 <10000
##  7 Gory Gopela           3939  3927  7866 <10000
##  8 Goumera               1918  1903  3821 <10000
##  9 Guidimakan Keri Kaff  9798 10234 20032 >=10000
## 10 Hawa Dembaya          3406  3445  6851 <10000
## # ... with 693 more rows
```

Création de variable: avec mutate (3)

A l'instar de select qui accepte la liste de toutes les variables à sélection, mutate aussi peut, avec un seul appel, exécuter plusieurs opérations de création de variables.

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création de deux nouvelles variables: numérique et catégorielle
  mutate(total = homme + femme,
         pop_cat = ifelse(total < 10000, "<10000", ">=10000"))
```

```
## # A tibble: 703 x 5
##    admin3_nom           homme femme total pop_cat
##    <chr>                <dbl> <dbl> <dbl> <chr>
##  1 Bangassi              6123  5974 12097 >=10000
##  2 Colimbine             6144  6353 12497 >=10000
##  3 Diamou                7115  7015 14130 >=10000
##  4 Djelebou             11466 12091 23557 >=10000
##  5 Faleme                5141  5017 10158 >=10000
##  6 Fegui                 1999  1934  3933 <10000
##  7 Gory Gopela           3939  3927  7866 <10000
##  8 Goumera               1918  1903  3821 <10000
##  9 Guidimakan Keri Kaff  9798 10234 20032 >=10000
## 10 Hawa Dembaya          3406  3445  6851 <10000
## # ... with 693 more rows
```

Suppression de variables: avec mutate

Comme dans R-base, l'on supprime une variable en lui affectant la valeur NULL.

```
  # Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  mutate(
    # Création de deux nouvelles variables: numérique et catégorielle
    total = homme + femme,
    pop_cat = ifelse(total < 10000, "<10000", ">=10000"),
    # Suppression de variables
    homme = NULL,
    femme = NULL
    )
```

```
## # A tibble: 703 x 3
##    admin3_nom           total pop_cat
##    <chr>                <dbl> <chr>
##  1 Bangassi             12097 >=10000
##  2 Colimbine            12497 >=10000
##  3 Diamou               14130 >=10000
##  4 Djelebou             23557 >=10000
##  5 Faleme               10158 >=10000
##  6 Fegui                 3933 <10000
##  7 Gory Gopela           7866 <10000
##  8 Goumera               3821 <10000
##  9 Guidimakan Keri Kaff 20032 >=10000
## 10 Hawa Dembaya          6851 <10000
## # ... with 693 more rows
```

Ne garder que le résultat: avec R-base

Une variante de la fonction mutate est la fonction transmute qui a la particularité de ne garder que les résultats issus des tâches qui lui ont été confiées. Elle présente certes une approche radicale par rapport à une suppression sélective des variables, mais elle est très commode pour celui que n'est intéressé que par les résultats. Elle se distingue ainsi de mutate qui, quant à elle, préserve les variables préexistantes à l'opération de création de nouvelles colonnes

Reprenons le dernier exemple.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Création d'une nouvelle variable: catégorielle
pop_df$pop_cat <- ifelse(pop_df$total < 10000, "<10000", ">=10000")
# Suppression de variables
pop_df$admin3_nom <- NULL
pop_df$homme <- NULL
pop_df$femme <- NULL
# Alternative: sélection des variables créées
# Aperçu
pop_df
```

```
## # A tibble: 703 x 2
##    total pop_cat
##    <dbl> <chr>
##  1 12097 >=10000
##  2 12497 >=10000
##  3 14130 >=10000
##  4 23557 >=10000
##  5 10158 >=10000
##  6  3933 <10000
##  7  7866 <10000
```

Ne garder que le résultat: avec mutate

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création de deux nouvelles variables: numérique et catégorielle
  transmute(total = homme + femme,
            pop_cat = ifelse(total < 10000, "<10000", ">=10000"))
```

```
## # A tibble: 703 x 2
##    total pop_cat
##    <dbl> <chr>
##  1 12097 >=10000
##  2 12497 >=10000
##  3 14130 >=10000
##  4 23557 >=10000
##  5 10158 >=10000
##  6  3933 <10000
##  7  7866 <10000
##  8  3821 <10000
##  9 20032 >=10000
## 10  6851 <10000
## # ... with 693 more rows
```

Comme résultat, nous avons les deux nouvelles colonnes. Toutes les autres ont été omises. Toutefois, le nombre d'observations n'a pas varié.

Sélection d'observations: `filter`

Avant de commencer

Regardons à nouveau notre jeu de données.

```
print(adm3_pop_2009)
```

```
## # A tibble: 703 x 9
##       id admin0_nom admin1_nom admin2_nom admin3_nom annee homme femme
##    <dbl> <chr>      <chr>      <chr>      <chr>      <dbl> <dbl> <dbl>
##  1     1 Mali       Kayes      Kayes      Bangassi    2009  6123  5974
##  2     2 Mali       Kayes      Kayes      Colimbine   2009  6144  6353
##  3     3 Mali       Kayes      Kayes      Diamou      2009  7115  7015
##  4     4 Mali       Kayes      Kayes      Djelebou    2009 11466 12091
##  5     5 Mali       Kayes      Kayes      Faleme      2009  5141  5017
##  6     6 Mali       Kayes      Kayes      Fegui       2009  1999  1934
##  7     7 Mali       Kayes      Kayes      Gory Gope~  2009  3939  3927
##  8     8 Mali       Kayes      Kayes      Goumera     2009  1918  1903
##  9     9 Mali       Kayes      Kayes      Guidimaka~  2009  9798 10234
## 10    10 Mali       Kayes      Kayes      Hawa Demb~  2009  3406  3445
## # ... with 693 more rows, and 1 more variable: source <chr>
```

Nous avons 703 observations, donc 703 communes. Regardons les plus grandes en matière de population. Disons, celles qui sont au dessus du seuil de 100000 habitants.

Sur la base de critères numériques: avec R-base (1)

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Critère de sélection: valeurs logiques (TRUE/FALSE)
pop_100000_plus <- pop_df$total > 100000
# Mise en oeuvre de la sélection
pop_df[pop_100000_plus, ]
```

```
## # A tibble: 12 x 4
##    admin3_nom        homme  femme  total
##    <chr>             <dbl>  <dbl>  <dbl>
##  1 Kayes Commune     65135  61184 126319
##  2 Kalabancoro       81018  80864 161882
##  3 Koutiala Commune  70905  70539 141444
##  4 Sikasso Commune  114171 112447 226618
##  5 Segou Commune     66819  66682 133501
##  6 Mopti Commune     60080  60706 120786
##  7 Commune I        168308 166587 334895
##  8 Commune II        78690  80670 159360
##  9 Commune III       63792  64874 128666
## 10 Commune IV       152153 152373 304526
## 11 Commune V        206749 206517 413266
## 12 Commune VI       237951 231702 469653
```

Sur la base de critères numériques: avec filter (1)

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(total > 100000)
```

```
## # A tibble: 12 x 4
##    admin3_nom         homme  femme  total
##    <chr>              <dbl>  <dbl>  <dbl>
##  1 Kayes Commune      65135  61184 126319
##  2 Kalabancoro        81018  80864 161882
##  3 Koutiala Commune   70905  70539 141444
##  4 Sikasso Commune   114171 112447 226618
##  5 Segou Commune      66819  66682 133501
##  6 Mopti Commune      60080  60706 120786
##  7 Commune I         168308 166587 334895
##  8 Commune II         78690  80670 159360
##  9 Commune III        63792  64874 128666
## 10 Commune IV        152153 152373 304526
## 11 Commune V         206749 206517 413266
## 12 Commune VI        237951 231702 469653
```

Là aussi, nous voyons que la séquence est plus économe en écriture.

Sur la base de critères numériques: avec R-base (2)

Il arrive souvent aussi que la sélection porte sur un interval ou une région. Dans ce genre de cas, **dplyr** a des fonctions spécialisées comme `between` ou `near`. Si l'on cherche les communes dont la population est comprise entre 50000 et 60000.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Critère de sélection: valeurs logiques (TRUE/FALSE)
pop_50k_60k <- pop_df$total >= 50000 & pop_df$total <= 60000
# Mise en oeuvre de la sélection
pop_df[pop_50k_60k, ]
```

```
## # A tibble: 13 x 4
##    admin3_nom       homme femme total
##    <chr>            <dbl> <dbl> <dbl>
##  1 Massigui         26249 28003 54252
##  2 Baguineda Camp   26406 25415 51821
##  3 Mande            28734 28615 57349
##  4 Ouelessebougou   24694 25345 50039
##  5 Kadiolo          25394 26622 52016
##  6 Koury            26842 27448 54290
##  7 Kolondieba       25976 27404 53380
##  8 Wassoulou Balle  25532 25941 51473
##  9 Bougouni Commune 29581 28957 58538
## 10 Koumantou        24828 26381 51209
## 11 Pelengana        27796 28051 55847
## 12 Tombouctou       27915 26714 54629
## 13 Tonka            25698 27578 53276
```

Sur la base de critères numériques: avec filter (2)

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(between(x = total, left = 50000, right = 60000))
```

```
## # A tibble: 13 x 4
##    admin3_nom       homme femme total
##    <chr>            <dbl> <dbl> <dbl>
##  1 Massigui         26249 28003 54252
##  2 Baguineda Camp   26406 25415 51821
##  3 Mande            28734 28615 57349
##  4 Ouelessebougou   24694 25345 50039
##  5 Kadiolo          25394 26622 52016
##  6 Koury            26842 27448 54290
##  7 Kolondieba       25976 27404 53380
##  8 Wassoulou Balle  25532 25941 51473
##  9 Bougouni Commune 29581 28957 58538
## 10 Koumantou        24828 26381 51209
## 11 Pelengana        27796 28051 55847
## 12 Tombouctou       27915 26714 54629
## 13 Tonka            25698 27578 53276
```

Sur la base de critères numériques: avec R-base (3)

Et pour les communes autour de 50000 habitants. Disons que nous prendrons en compte les communes dans les valeurs environnantes et ce jusqu'au 2500 personnes.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Critère de sélection: valeurs logiques (TRUE/FALSE)
pop_50k_tol2500 <- pop_df$total >= 50000 - 2500 & pop_df$total <= 50000 + 2500
# Mise en oeuvre de la sélection
pop_df[pop_50k_tol2500, ]
```

```
## # A tibble: 9 x 4
##   admin3_nom      homme femme total
##   <chr>           <dbl> <dbl> <dbl>
## 1 Kita Commune    24054 24989 49043
## 2 Guegneka        23949 24092 48041
## 3 Baguineda Camp  26406 25415 51821
## 4 Ouelessebougou  24694 25345 50039
## 5 Kadiolo         25394 26622 52016
## 6 Misseni         28443 19230 47673
## 7 Wassoulou Balle 25532 25941 51473
## 8 Koumantou       24828 26381 51209
## 9 Sony Aliber     23472 24146 47618
```

Sur la base de critères numériques: avec filter (3)

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(near(x = total, y = 50000, tol = 2500))
```

```
## # A tibble: 9 x 4
##   admin3_nom      homme femme total
##   <chr>           <dbl> <dbl> <dbl>
## 1 Kita Commune    24054 24989 49043
## 2 Guegneka        23949 24092 48041
## 3 Baguineda Camp  26406 25415 51821
## 4 Ouelessebougou  24694 25345 50039
## 5 Kadiolo         25394 26622 52016
## 6 Misseni         28443 19230 47673
## 7 Wassoulou Balle 25532 25941 51473
## 8 Koumantou       24828 26381 51209
## 9 Sony Aliber     23472 24146 47618
```

Sur la base de critères catégoriels: avec R-base

La sélection peut aussi porter sur des variables en caractères ou catégorielles. Supposons que nous souhaitons connaître la population d'une commune dont nous avons le nom: *Bambara Maoudé*.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Critère de sélection: valeurs logiques (TRUE/FALSE)
bambara_maoude <- pop_df$admin3_nom == "Bambara Maoude"
# Mise en oeuvre de la sélection
pop_df[bambara_maoude, ]
```

```
## # A tibble: 1 x 4
##   admin3_nom     homme femme total
##   <chr>          <dbl> <dbl> <dbl>
## 1 Bambara Maoude  8315  8170 16485
```

Sur la base de critères catégoriels: avec mutate

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(admin3_nom == "Bambara Maoude")
```

```
## # A tibble: 1 x 4
##   admin3_nom     homme femme total
##   <chr>          <dbl> <dbl> <dbl>
## 1 Bambara Maoude  8315  8170 16485
```

Moins de lignes! Même résultat!

Sur la base d'expressions régulières: avec R-base (1)

Comme avec `select`, les fonctions relatives aux préfixes et suffixes (et similaires) peuvent être mobilisées dans `filter` aussi. Suppossons que l'on veuille connaître la population de toutes les communes dont le nom se termine par *dougou*.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Sélection des communes répondant au critère
select_nom <- endsWith(x = pop_df$admin3_nom, suffix = "dougou")
# Mise en oeuvre de la sélection
pop_df[select_df, ]
```

```
## # A tibble: 312 x 4
##    admin3_nom        homme femme  total
##    <chr>             <dbl> <dbl>  <dbl>
##  1 Colimbine          6144  6353  12497
##  2 Diamou             7115  7015  14130
##  3 Djelebou          11466 12091  23557
##  4 Faleme             5141  5017  10158
##  5 Karakoro           7436  7770  15206
##  6 Kayes Commune     65135 61184 126319
##  7 Kemene Tambo       8381  8574  16955
##  8 Khouloum           9734  9260  18994
##  9 Marena Diombougou  8996  9676  18672
## 10 Marintoumania      4001  4060   8061
## # ... with 302 more rows
```

Sur la base d'expressions régulières: avec filter (1)

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(endsWith(x = admin3_nom, suffix = "dougou"))
```

```
## # A tibble: 28 x 4
##    admin3_nom    homme femme total
##    <chr>         <dbl> <dbl> <dbl>
##  1 Gomitradougou  3638  3658  7296
##  2 Dinandougou   10571 10789 21360
##  3 Diedougou     17541 18351 35892
##  4 Dolendougou    6831  7389 14220
##  5 Kaladougou    19429 19742 39171
##  6 Kilidougou     7578  7930 15508
##  7 N'dlondougou  10326 10667 20993
##  8 N'garadougou   7633  7960 15593
##  9 Tenindougou    7681  7725 15406
## 10 Maramandougou  7237  7302 14539
## # ... with 18 more rows
```

La fonction `filter` épouse aussi bien les fonctions spécifiques aux expressions régulières dans R-base - comme dans l'exemple précédent - que celles présentes dans les packages dédiés du **tidyverse**.

Sur la base d'expressions régulières: avec R-base (2)

Les fonctions de **stringr** peuvent s'avérer très commode dans la sélection d'observations. Considérons les observations pour les communes qui ont la lettre *z* dans leur nom.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Sélection des communes répondant au critère
detect_z <- grepl(pattern = "z", x = tolower(pop_df$admin3_nom))
# Mise en oeuvre de la sélection
pop_df[detect_z, ]
```

```
## # A tibble: 21 x 4
##    admin3_nom    homme femme total
##    <chr>         <dbl> <dbl> <dbl>
##  1 Zan Coulibaly  9235  9255 18490
##  2 Zegoua        16480 16638 33118
##  3 Zanfigue       7492  7747 15239
##  4 Zangasso       9391 10105 19496
##  5 Zanina         3619  3806  7425
##  6 Zebala         8293  8997 17290
##  7 Zantiebougou  17667 18100 35767
##  8 Kafouziela     3108  3304  6412
##  9 Sanzana        5479  5820 11299
## 10 Zanferebougou  2237  2416  4653
## # ... with 11 more rows
```

Sur la base d'expressions régulières: avec filter (2)

```
# Chargement du package stringr
library(stringr)
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(str_detect(string = tolower(admin3_nom), pattern = "z"))
```

```
## # A tibble: 21 x 4
##    admin3_nom    homme femme total
##    <chr>         <dbl> <dbl> <dbl>
##  1 Zan Coulibaly  9235  9255 18490
##  2 Zegoua        16480 16638 33118
##  3 Zanfigue       7492  7747 15239
##  4 Zangasso       9391 10105 19496
##  5 Zanina         3619  3806  7425
##  6 Zebala         8293  8997 17290
##  7 Zantiebougou  17667 18100 35767
##  8 Kafouziela     3108  3304  6412
##  9 Sanzana        5479  5820 11299
## 10 Zanferebougou  2237  2416  4653
## # ... with 11 more rows
```

Sur la base d'index: avec R-base

Comme nous l'avons vu pour les variables, avec les observations aussi, la sélection peut se faire à partir de l'index. Chercons à afficher les observations pour les 10 premières observations impaires.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Mise en oeuvre de la sélection
pop_df[seq(from = 1, to = 19, by = 2), ]
```

```
## # A tibble: 10 x 4
##    admin3_nom           homme femme total
##    <chr>                <dbl> <dbl> <dbl>
##  1 Bangassi              6123  5974 12097
##  2 Diamou                7115  7015 14130
##  3 Faleme                5141  5017 10158
##  4 Gory Gopela           3939  3927  7866
##  5 Guidimakan Keri Kaff  9798 10234 20032
##  6 Karakoro              7436  7770 15206
##  7 Kemene Tambo          8381  8574 16955
##  8 Koniakary             4061  4080  8141
##  9 Koussane             10472 11118 21590
## 10 Logo                  6022  5967 11989
```

Sur la base d'index: avec filter

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(row_number() %in% seq(from = 1, to = 19, by = 2))
```

```
## # A tibble: 10 x 4
##    admin3_nom           homme femme total
##    <chr>                <dbl> <dbl> <dbl>
##  1 Bangassi              6123  5974 12097
##  2 Diamou                7115  7015 14130
##  3 Faleme                5141  5017 10158
##  4 Gory Gopela           3939  3927  7866
##  5 Guidimakan Keri Kaff  9798 10234 20032
##  6 Karakoro              7436  7770 15206
##  7 Kemene Tambo          8381  8574 16955
##  8 Koniakary             4061  4080  8141
##  9 Koussane             10472 11118 21590
## 10 Logo                  6022  5967 11989
```

Filtres multiples: avec R-base

en matière de sélections d'observations, il y a plusieurs façons de combiner des critères. On peut :

- les ajouter avec le signe &;
- les présenter comme des alternatives avec le signe |; ou
- opérer avec une logique de négation avec le signe !.

Ces principes sont aussi bien valides avec R-base que **dplyr**.

Supposons que nous voulons connaître les communes de plus de 100000 où il y a plus de femmes que d'hommes. Ici, les deux critères s'accumulent.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Critère de sélection (1): valeurs logiques (TRUE/FALSE)
pop_100000_plus <- pop_df$total > 100000
# Critère de sélection (1): valeurs logiques (TRUE/FALSE)
femmes_sup_hommes <- pop_df$femme > pop_df$homme
# Mise en oeuvre de la sélection
pop_df[pop_100000_plus & femmes_sup_hommes, ]
```

```
## # A tibble: 4 x 4
##   admin3_nom     homme  femme  total
##   <chr>          <dbl>  <dbl>  <dbl>
## 1 Mopti Commune  60080  60706 120786
## 2 Commune II     78690  80670 159360
## 3 Commune III    63792  64874 128666
## 4 Commune IV    152153 152373 304526
```

Filtres multiples: avec filter

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(
    # Critère 1: population > 100000
    total > 100000,
    # Critère 2: femme > homme
    femme > homme
    )
```

```
## # A tibble: 4 x 4
##   admin3_nom     homme  femme  total
##   <chr>          <dbl>  <dbl>  <dbl>
## 1 Mopti Commune  60080  60706 120786
## 2 Commune II     78690  80670 159360
## 3 Commune III    63792  64874 128666
## 4 Commune IV    152153 152373 304526
```

Tri d'observations: arrange

Triage: avec R-base

Souvent, il arrive qu'à la suite d'opérations de sélections (aussi bien de variables que de colonnes) que l'on souhaite ordonner les résultats selon un ordre bien précis. Ceci peut servir souvent en matière d'affichage ou même servir de base pour des sélections.

Considérons qu'après avoir filtré pour ne garder que les communes qui ont plus de 100000 habitants et plus de femmes que d'hommes, que nous souhaitions ordonner la population.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin3_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Critère de sélection (1): valeurs logiques (TRUE/FALSE)
pop_100000_plus <- pop_df$total > 100000
# Critère de sélection (1): valeurs logiques (TRUE/FALSE)
femmes_sup_hommes <- pop_df$femme > pop_df$homme
# Mise en oeuvre de la sélection
pop_df_filtre <- pop_df[pop_100000_plus | femmes_sup_hommes, ]
# Tri: ordre croissant
pop_decroissant <- order(pop_df_filtre$total)
# Mise en oeuvre du tri
pop_df_filtre[pop_decroissant, ]
```

```
## # A tibble: 538 x 4
##    admin3_nom            homme femme total
##    <chr>                 <dbl> <dbl> <dbl>
##  1 Adarmalane              463   492   955
##  2 Bassirou                825   893  1718
##  3 Arham                  1358  1461  2819
##  4 Soignebougou           1547  1552  3099
##  5 Ouro Modi              1566  1762  3328
##  6 Sokourani Missirikoro  1632  1740  3372
##  7 Diou                   1744  1760  3504
```

Triage: avec arrange (1)

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(
    # Critère 1: population > 100000
    total > 100000 |
    # Critère 2: femme > homme
    femme > homme
    ) %>%
  # Tri: ordre décroissant
  arrange(desc(total))
```

```
## # A tibble: 538 x 4
##    admin3_nom        homme  femme  total
##    <chr>             <dbl>  <dbl>  <dbl>
##  1 Commune VI       237951 231702 469653
##  2 Commune V        206749 206517 413266
##  3 Commune I        168308 166587 334895
##  4 Commune IV       152153 152373 304526
##  5 Sikasso Commune  114171 112447 226618
##  6 Kalabancoro       81018  80864 161882
##  7 Commune II        78690  80670 159360
##  8 Koutiala Commune  70905  70539 141444
##  9 Segou Commune     66819  66682 133501
## 10 Commune III       63792  64874 128666
## # ... with 528 more rows
```

Triage: avec arrange (2)

L'on peut agrémenter arrange avec une fonction qui spécifie le nombre d'observations à afficher ou à sauvegarder: top_n. Celle-ci peut opérer par le haut (les valeurs élevées)...

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(
    # Critère 1: population > 100000
    total > 100000 |
    # Critère 2: femme > homme
    femme > homme
    ) %>%
  # Tri: décroissant
  arrange(desc(total)) %>%
  # Sélection des 5 premières observations
  top_n(n = 5, wt = total)
```

```
## # A tibble: 5 x 4
##   admin3_nom       homme  femme  total
##   <chr>            <dbl>  <dbl>  <dbl>
## 1 Commune VI      237951 231702 469653
## 2 Commune V       206749 206517 413266
## 3 Commune I       168308 166587 334895
## 4 Commune IV      152153 152373 304526
## 5 Sikasso Commune 114171 112447 226618
```

Triage: avec arrange (3)

...ou par le bas (les valeurs faibles).

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Sélection des variables d'intérêt
  select(admin3_nom, homme, femme) %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Sélection d'observations d'intérêt
  filter(
    # Critère 1: population > 100000
    total > 100000 |
    # Critère 2: femme > homme
    femme > homme
    ) %>%
  # Tri: décroissant
  arrange(desc(total)) %>%
  # Sélection des 5 dernières observations
  top_n(n = -5, wt = total)
```

```
## # A tibble: 5 x 4
##   admin3_nom   homme femme total
##   <chr>        <dbl> <dbl> <dbl>
## 1 Ouro Modi     1566  1762  3328
## 2 Soignebougou  1547  1552  3099
## 3 Arham         1358  1461  2819
## 4 Bassirou       825   893  1718
## 5 Adarmalane     463   492   955
```

Vers l'agrégation: group_by et summarize

Aperçu

Jusque là, nous avons opéré en ajoutant ou enlevant des variables, en sélectionnant ou excluant des observations. Ces opérations ont toutes été intra-individuelles c'est-à-dire qu'à aucun moment il n'a été nécéssaire de mélanger les valeurs de deux ou plusieurs observations. Or, il arrive souvant que le *data scientist* ait besoin d'agréer des valeurs pour approfondir sa propre compréhension ou tout simplement synthétiser ses résultats. **dplyr** comptent deux fonctions sont pratiques pour ce faire. Il s'agit de `group_by` et `summarize` (`summarise` aussi marche). Elles viennent consolider les quatre que nous avons vues. La première `group_by` définit les groupes sur lesquels les opérations d'agrégation doivent être exécutées. Quant à `summarize`, elle explicite ces opérations. Illustrons pour mieux comprendre.

Créer, agréger et ordonner: avec R-base

Supposons que l'on veuille agréger la population totale par région (admin1_nom) et ordonner celle-ci par ordre décroissant. Ceci reviendrait à faire la somme de la population totale en définissant admin1_nom comme variable de groupage.

```
# Sélection des variables d'intérêt
pop_df <- adm3_pop_2009[ , c("admin1_nom", "homme", "femme")]
# Création d'une nouvelle variable: numérique
pop_df$total <- pop_df$femme + pop_df$homme
# Opération d'agrégation
pop_df_adm1 <- aggregate(formula = total ~ admin1_nom, data = pop_df, FUN = sum)
# Tri: ordre décroissant
pop_decroissant <- order(pop_df_adm1$total, decreasing = TRUE)
# Mise en oeuvre du tri
pop_df_adm1[pop_decroissant, ]
```

```
##   admin1_nom   total
## 8    Sikasso 2644458
## 5 Koulikouro 2419212
## 7      Segou 2321651
## 6      Mopti 2038855
## 3      Kayes 2013076
## 1     Bamako 1810366
## 9 Tombouctou  671005
## 2        Gao  542304
## 4      Kidal   67739
```

Créer, agréger et ordonner: avec group_by et summarize

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Spécification du niveau d'agrégation
  group_by(admin1_nom) %>%
  # Opération d'agrégation
  summarize(population = sum(total)) %>%
  # Tri: décroissant
  arrange(desc(population))
```

```
## # A tibble: 9 x 2
##   admin1_nom population
##   <chr>           <dbl>
## 1 Sikasso       2644458
## 2 Koulikouro    2419212
## 3 Segou         2321651
## 4 Mopti         2038855
## 5 Kayes         2013076
## 6 Bamako        1810366
## 7 Tombouctou     671005
## 8 Gao            542304
## 9 Kidal           67739
```

L'un sans l'autre: summarize sans group_by

summarize acceptent la majorité des fonctions statistiques de R-base: sum pour la somme, mean pour la moyenne, sd pour l'écart-type, min pour le minimum, max pour le maximum, etc.

Il est utile de noter que, bien que les deux fonctions opèrent généralement en paire, elles ne sont pas toutefois obligées d'être ensemble...enfin, pas tout le temps.

Considérons par exemple que l'on veut agréger la population totale aussi bien pour les hommes que pour les femmes. Comme le groupe de référence est l'ensemble des observations (toutes les communes), l'on n'a pas besoin de group_by.

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Opérations d'agrégation
  summarize(
    # Hommes
    homme = sum(homme),
    # Femmes
    femme = sum(femme),
    # Total
    total = sum(total)
    )
```

```
## # A tibble: 1 x 3
##     homme   femme    total
##     <dbl>   <dbl>    <dbl>
## 1 7204994 7323672 14528666
```

L'un sans l'autre: group_by sans summarize

Maintenant, faisons l'inverse. Déterminons la part de chaque commune dans la population régionale. Et gardons seulement celles qui représentent plus de 5% de la population de leur région.

```
# Jeu de données de départ
adm3_pop_2009 %>%
  # Création d'une nouvelle variable
  mutate(total = homme + femme) %>%
  # Spécification du niveau d'agrégation
  group_by(admin1_nom) %>%
  # Création d'une nouvelle variable
  mutate(
    # Population au niveau de admin1_nom
    population_region = sum(total),
    # Ratio population de la commune / population de la région
    part_commune = total / population_region
    ) %>%
  # Filtre: part > 5%
  filter(part_commune > 0.05) %>%
  # Tri des résultats: ordre décroissant
  arrange(desc(part_commune)) %>%
  # Sélection des variables d'intérêt
  select(contains("nom"), part_commune)
```

```
## # A tibble: 28 x 5
## # Groups:   admin1_nom [9]
##    admin0_nom admin1_nom admin2_nom admin3_nom      part_commune
##    <chr>      <chr>      <chr>      <chr>                 <dbl>
##  1 Mali       Kidal      Kidal      Kidal                 0.383
##  2 Mali       Bamako     Bamako     Commune VI            0.259
##  3 Mali       Bamako     Bamako     Commune V             0.228
##  4 Mali       Bamako     Bamako     Commune I             0.185
```

Conclusion

Conclusion

Nous venons de voir que **dplyr** est un outil très riche. Avec un vocabulaire simple et accessible, il met à la disposition du *data scientist* une panoplie d'outil qui facilite la manipulation de données.

Merci