Transformer des données dans R avec **tidyr**

Fousseynou Bah

10-Sep-2020

Section 1

Introduction

Objectif de ce chapitre

Dans le présent chapitre, nous continuons avec l'exploration des outils de manipulations de données. Ici, nous introduisons **tidyr**.

Dans la sagesse commune du **tidyverse**, un jeu de données est *tidy* - au sens bien ordonné - quand il obéit à deux principes de base:

- les colonnes sont des attributs; et
- les lignes des observations.

Cette disposition n'est pas toujours celle dans laquelle les données parviennent au *data scientist*. Les organisations tendent à donner des données rectangulaires assez longues, avec beaucoup d'observations. Malheureusement, quand il s'agit de la présentation, ce schéma est souvent difficile d'accès. D'où la nécéssité d'étendre les données dans un format plus large. *tidyr* permet le passage d'un jeu de données d'un état à un autre avec une très grande facilité. Nous allons voir les deux majeurs fonctions qui permettent ces transformations d'état.

A propos de **tidyr**

Le package **tidyr** est lui aussi membre de l'univers **tidyverse**. A l'instar de **dplyr**, ses fonctions majeures sont des verbes.

Alors, que fait **tidyr**? Il a un dessein plutôt radical quand on le compare à **dplyr**. Pendant que ce dernier s'occupe à créer et à supprimer des variables et/ou des observations, **tidyr**, lui, est capable de permuter les colonnes et les lignes...bref, de "pivoter" les données. Ce faisant, il outille le *data scientist* à refondre ses jeux de données selon le schéma qui se prête le mieux à l'analyse qu'il vise. Voici le logo associé à ce package.

Installer **tidyr**

Pour travailler avec **tidyr**, l'on commence par l'installer sur son poste de travail. Etant intégré au **tidyverse**, il est possible de l'installer en même temps que les autres composantes de celui-ci.

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

Données

Nous allons illustrer ce chapitre avec une compilation de données tirées des Recensements Généraux de la Population et de l'Habitat au Mali, menés respectivement en 1998 et 2009. Il s'agit de données sur la population par commune.

Les données utilisées sont accessibles à partir d'ici et ici.

Section 2

Pivoter les données

Vers le format long: gather (1)

La fonction `gather` - qui veut dire *réunir* en anglais - fond les données en format long. En opérant dans le sens *tidy*, elle permet d'aligner dans la même colonne les valeurs relatives au même attribut et à aligner sur la même ligne les attributs relatifs à la même observation.

Pour illustrer, retournons au jeu de données du chapitre précédent, les données sur les populations des communes du Mali en 2009 (RGPH, 2009), disponibles ici.

Rappelons les données:

```
# La source
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)
adm3_pop_2009
```

```
## # A tibble: 703 x 9
##       id admin0_nom admin1_nom admin2_nom admin3_nom    annee homme femme source
##    <dbl> <chr>      <chr>      <chr>      <chr>          <dbl> <dbl> <dbl> <chr>
##  1     1 Mali       Kayes      Kayes      Bangassi        2009  6123  5974 RGPH
##  2     2 Mali       Kayes      Kayes      Colimbine       2009  6144  6353 RGPH
##  3     3 Mali       Kayes      Kayes      Diamou          2009  7115  7015 RGPH
##  4     4 Mali       Kayes      Kayes      Djelebou        2009 11466 12091 RGPH
```

Vers le format long: gather (2)

Nous relevons qu'il existe deux colonnes relatives au même attribut, le sexe. Il s'agit des variables: `homme` et `femme`. Il opérant sur la base de l'organisation `tidy`, ce jeu de données a bien besoin d'une refonte. `gather` est là pour ça.

```
adm3_pop_2009_long <-
  # Jeu de données de départ
  adm3_pop_2009 %>%
  # Réorganisation des données
  gather(
    # Nom de la variable clé: les catégories fondues ensemble
    key = sexe,
    # Nom de la variable de valeur: les valeurs correspondants aux catégories fondues ensemble
    value = population,
    # Liste des anciennes variables à reclasser
    homme, femme
  )
# Aperçu du nouveau jeu de données
adm3_pop_2009_long
```

```
## # A tibble: 1,406 x 9
##       id admin0_nom admin1_nom admin2_nom admin3_nom annee source sexe
##    <dbl> <chr>      <chr>      <chr>      <chr>      <dbl> <chr>  <chr>
##  1     1 Mali       Kayes      Kayes      Bangassi    2009 RGPH   homme
##  2     2 Mali       Kayes      Kayes      Colimbine   2009 RGPH   homme
##  3     3 Mali       Kayes      Kayes      Diamou      2009 RGPH   homme
##  4     4 Mali       Kayes      Kayes      Djelebou    2009 RGPH   homme
##  5     5 Mali       Kayes      Kayes      Faleme      2009 RGPH   homme
##  6     6 Mali       Kayes      Kayes      Fegui       2009 RGPH   homme
##  7     7 Mali       Kayes      Kayes      Gory Gope~  2009 RGPH   homme
```

Vers le format long: `gather` (2)

Vous remarquerez que l'on a deux fois plus d'observations. De 703 l'on est passé à 1406. C'est parce qu'au lieu d'avoir une observation par commune, l'on a une observation par commune et par chaque sexe. L'unité d'observation est mieux alignée maintenant sur les données disponibles. Là, nous avons une colonne indicative du sexe dont il s'agit et une autre affichant la population correspondante.

Vers le format long: R-base

Avant d'avancer, faisons preuve d'un peu de curiosité. Comment est-ce que ce problème peut être résolu dans R-base? Comme pour tout, dans R, un seul problème peut être résolu de manières diverses. Passer par la fonction `reshape`, composante d'un package de base, **stats**.

```
# Réorganisation
reshape(
  # Jeu de données de départ
  data = adm3_pop_2009,
  # Valeurs variantes à refondre
  varying = c("homme", "femme"),
  # Nom de la nouvelle variable de valeurs
  v.names = "population",
  # Nom de la nouvelle variable de catégories
  timevar = "sexe",
  # Nom de lignes issues de la refonte (à calculer)
  new.row.names = 1:1406,
  # Spécification du sens de la refonte: large vers long
  direction = "long")
```

```
## # A tibble: 1,406 x 9
##       id admin0_nom admin1_nom admin2_nom admin3_nom annee source  sexe
##  * <dbl> <chr>      <chr>      <chr>      <chr>      <dbl> <chr>  <int>
##  1     1 Mali       Kayes      Kayes      Bangassi    2009 RGPH       1
##  2     2 Mali       Kayes      Kayes      Colimbine   2009 RGPH       1
##  3     3 Mali       Kayes      Kayes      Diamou      2009 RGPH       1
##  4     4 Mali       Kayes      Kayes      Djelebou    2009 RGPH       1
##  5     5 Mali       Kayes      Kayes      Faleme      2009 RGPH       1
```

Vers le format large: spread (1)

Si le format long est idéal pour l'analyse de données, c'est généralement le format large qui se prête le mieux aux présentations. Imaginez si l'on partait du jeu de données suivant:

```
adm3_pop_2009_long
```

```
## # A tibble: 1,406 x 9
##       id admin0_nom admin1_nom admin2_nom admin3_nom annee source sexe
##    <dbl> <chr>      <chr>      <chr>      <chr>      <dbl> <chr>  <chr>
##  1     1 Mali       Kayes      Kayes      Bangassi    2009 RGPH   homme
##  2     2 Mali       Kayes      Kayes      Colimbine   2009 RGPH   homme
##  3     3 Mali       Kayes      Kayes      Diamou      2009 RGPH   homme
##  4     4 Mali       Kayes      Kayes      Djelebou    2009 RGPH   homme
##  5     5 Mali       Kayes      Kayes      Faleme      2009 RGPH   homme
##  6     6 Mali       Kayes      Kayes      Fegui       2009 RGPH   homme
##  7     7 Mali       Kayes      Kayes      Gory Gope~  2009 RGPH   homme
##  8     8 Mali       Kayes      Kayes      Goumera     2009 RGPH   homme
##  9     9 Mali       Kayes      Kayes      Guidimaka~  2009 RGPH   homme
## 10    10 Mali       Kayes      Kayes      Hawa Demb~  2009 RGPH   homme
## # ... with 1,396 more rows, and 1 more variable: population <dbl>
```

...et que l'on souhaitait présenter la population par région et par sexe.

Vers le format large: spread (2)

Les outils de **dplyr** nous permettraient d'effectuer les agrégations avec une très grande commodité.

```
# Jeu de données de départ
adm3_pop_2009_long %>%
  # Spécification du niveau d'agrégation
  group_by(admin1_nom, sexe) %>%
  # Opération d'agrégation
  summarize(population = sum(population))
```

```
## # A tibble: 18 x 3
## # Groups:   admin1_nom [9]
##    admin1_nom sexe  population
##    <chr>      <chr>      <dbl>
##  1 Bamako     femme     902723
##  2 Bamako     homme     907643
##  3 Gao        femme     270685
##  4 Gao        homme     271619
##  5 Kayes      femme    1020247
##  6 Kayes      homme     992829
##  7 Kidal      femme      31404
##  8 Kidal      homme      36335
##  9 Koulikouro femme    1219803
## 10 Koulikouro homme    1199409
## 11 Mopti      femme    1030578
## 12 Mopti      homme    1008277
## 13 Segou      femme    1173313
## 14 Segou      homme    1148338
## 15 Sikasso    femme    1339803
```

Vers le format large: spread (3)

Avec quelques lignes, l'on a pu faire l'opération d'agrégation. Toutefois, cette présentation est peu commode pour un rapport ou une diapositive. Transformons en tableau croisé à l'aide de la fonction spread.

```
# Jeu de données de départ
adm3_pop_2009_long %>%
  # Spécification du niveau d'agrégation
  group_by(admin1_nom, sexe) %>%
  # Opération d'agrégation
  summarize(population = sum(population)) %>%
  # Transformation en format large
  spread(
    # Nom de la variable clé: les catégories à séparer en colonnes
    key = sexe,
    # Nom de la variable de valeur: les valeurs correspondants aux catégories à séparer
    value = population
    )
```

```
## # A tibble: 9 x 3
## # Groups:   admin1_nom [9]
##   admin1_nom   femme   homme
##   <chr>        <dbl>   <dbl>
## 1 Bamako      902723  907643
## 2 Gao         270685  271619
## 3 Kayes      1020247  992829
## 4 Kidal        31404   36335
## 5 Koulikouro 1219803 1199409
## 6 Mopti      1030578 1008277
## 7 Segou      1173313 1148338
```

Vers le format large: R-base

Reprenons ces exemples avec la fonction `reshape` du package **stats**.

```
adm3_pop_2009_rshp_wd <- reshape(
  # Jeu de données de départ
  data = adm3_pop_2009_long,
  # Nom de la variable numérique à disperser
  v.names = "population",
  # Noms des variables à maintenir
  idvar = c("id", "admin0_nom", "admin1_nom", "admin2_nom", "admin3_nom", "annee", "source"),
  # Nom de la variable catégorique à disperser en colones
  timevar = "sexe",
  # Spécification du sens de la refonte: long vers large
  direction = "wide")
# Impression
adm3_pop_2009_rshp_wd
```

```
## # A tibble: 703 x 8
##       id admin0_nom admin1_nom admin2_nom admin3_nom annee source
##    <dbl> <chr>      <chr>      <chr>      <chr>      <dbl> <chr>
##  1     1 Mali       Kayes      Kayes      Bangassi    2009 RGPH
##  2     2 Mali       Kayes      Kayes      Colimbine   2009 RGPH
##  3     3 Mali       Kayes      Kayes      Diamou      2009 RGPH
##  4     4 Mali       Kayes      Kayes      Djelebou    2009 RGPH
##  5     5 Mali       Kayes      Kayes      Faleme      2009 RGPH
##  6     6 Mali       Kayes      Kayes      Fegui       2009 RGPH
##  7     7 Mali       Kayes      Kayes      Gory Gope~  2009 RGPH
##  8     8 Mali       Kayes      Kayes      Goumera     2009 RGPH
##  9     9 Mali       Kayes      Kayes      Guidimaka~  2009 RGPH
## 10    10 Mali       Kayes      Kayes      Hawa Demb~  2009 RGPH
## # ... with 693 more rows, and 1 more variable: `population.c("homme",
```

Section 3

Séparer et unir

Séparateur: separate (1)

Le changement de disposition des données peut requérir des précautions particulières en matière de traitement des noms de variables. Dans les cas que nous avons vu avec gather et spread, nous avions affaire à une seul attribut dispersés en colonnes (le sexe: homme et femme). Souvent, la situation se complique plus que ça. Regardons le cas suivant:

```
# La source
source_donnees <- paste0("https://raw.githubusercontent.com/fousseynoubah/dswr_slides/master/",
                         "6_Transformer_Donnees_dans_R_avec_tidyr/data/adm3_pop_1998_2009_wide.csv")
# Chargement du package "readr"
library(readr)
# Importation des données
adm3_pop_1998_2009_wide <- read_csv(source_donnees)
# Impression
```

```
## Rows: 703
## Columns: 10
## $ id         <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1...
## $ admin0_nom <chr> "Mali", "Mali", "Mali", "Mali", "Mali", "Mali", "Mali", ...
## $ admin1_nom <chr> "Kayes", "Kayes", "Kayes", "Kayes", "Kayes", "Kayes", "K...
## $ admin2_nom <chr> "Kayes", "Kayes", "Kayes", "Kayes", "Kayes", "Kayes", "K...
## $ admin3_nom <chr> "Bangassi", "Colimbine", "Diamou", "Djelebou", "Faleme",...
## $ homme_1998 <dbl> 3351, 4828, 6361, 7903, 3232, 1333, 2682, 1295, 6713, 25...
## $ homme_2009 <dbl> 6123, 6144, 7115, 11466, 5141, 1999, 3939, 1918, 9798, 3...
## $ femme_1998 <dbl> 3263, 4913, 6189, 8403, 3309, 1355, 2614, 1294, 7521, 26...
## $ femme_2009 <dbl> 5974, 6353, 7015, 12091, 5017, 1934, 3927, 1903, 10234, ...
## $ source     <chr> "RGPH", "RGPH", "RGPH", "RGPH", "RGPH", "RGPH", "RGPH", ...
```

Séparateur: separate (1)

Nous voyons ici quatre variables qui déclinent en réalité deux attributs: l'année et le sexe. Transformons les données en format long.

```
# Jeu de données de départ
adm3_pop_1998_2009_wide %>%
  gather(
    # Nom de la variable clé: les catégories fondues ensemble
    key = groupe,
    # Nom de la variable de valeur: les valeurs correspondants aux catégories fondues ensemble
    value = population,
    # Liste des anciennes variables à reclasser
    homme_1998, femme_1998, homme_2009, femme_2009
    ) %>%
  # Sélection de variables pour focaliser l'attention sur les variables essentielles
select(admin3_nom, groupe, population)
```

```
## # A tibble: 2,812 x 3
##    admin3_nom           groupe     population
##    <chr>                <chr>           <dbl>
##  1 Bangassi             homme_1998       3351
##  2 Colimbine            homme_1998       4828
##  3 Diamou               homme_1998       6361
##  4 Djelebou             homme_1998       7903
##  5 Faleme               homme_1998       3232
##  6 Fegui                homme_1998       1333
##  7 Gory Gopela          homme_1998       2682
##  8 Goumera              homme_1998       1295
##  9 Guidimakan Keri Kaff homme_1998       6713
## 10 Hawa Dembaya         homme_1998       2517
```

Séparateur: separate (2)

Reprenons la chaîne.

```
adm3_pop_1998_2009_tidy <-
  # Jeu de données de départ
  adm3_pop_1998_2009_wide %>%
  # Passage au format long
  gather(
    # Nom de la variable clé: les catégories fondues ensemble
    key = groupe,
    # Nom de la variable de valeur: les valeurs correspondants aux catégories fondues ensemble
    value = population,
    # Liste des anciennes variables à reclasser
    homme_1998, femme_1998, homme_2009, femme_2009
    ) %>%
  # Séparation
  separate(
    # Nom de la variable à scinder
    col = groupe,
    # Noms des nouvelles variables à créer
    into = c("sexe", "annee"),
    # Signe séparateur
    sep = "_",
    # Détecter et ajuster le type de nouvelles variables
    convert = TRUE
    )
```

Séparateur: separate (3)

```
# Sélection de variables pour focaliser l'attention sur les variables essentielles
adm3_pop_1998_2009_tidy %>%
  select(admin3_nom, sexe, annee, population)
```

```
## # A tibble: 2,812 x 4
##    admin3_nom           sexe  annee population
##    <chr>                <chr> <int>      <dbl>
##  1 Bangassi             homme  1998       3351
##  2 Colimbine            homme  1998       4828
##  3 Diamou               homme  1998       6361
##  4 Djelebou             homme  1998       7903
##  5 Faleme               homme  1998       3232
##  6 Fegui                homme  1998       1333
##  7 Gory Gopela          homme  1998       2682
##  8 Goumera              homme  1998       1295
##  9 Guidimakan Keri Kaff homme  1998       6713
## 10 Hawa Dembaya         homme  1998       2517
## # ... with 2,802 more rows
```

Nous obtenons là un jeu de données *tidy* car toutes les colonnes sont relatives à **1** attribut et **1** seulement.

Unificateur: unite (1)

Imaginez maintenant que l'on veuille utiliser le nouveau jeu de données pour agréger les populations par région (au lieu de commune), par année et par sexe.

```
# Jeu de données de départ
adm3_pop_1998_2009_tidy %>%
  # Spécification du niveau d'agrégation
  group_by(admin1_nom, sexe, annee) %>%
  # Opération d'agrégation
  summarize(population = sum(population, na.rm = TRUE))
```

```
## # A tibble: 36 x 4
## # Groups:   admin1_nom, sexe [18]
##    admin1_nom sexe  annee population
##    <chr>      <chr> <int>      <dbl>
##  1 Bamako     femme  1998     501329
##  2 Bamako     femme  2009     902723
##  3 Bamako     homme  1998     514967
##  4 Bamako     homme  2009     907643
##  5 Gao        femme  1998     174330
##  6 Gao        femme  2009     270685
##  7 Gao        homme  1998     167212
##  8 Gao        homme  2009     271619
##  9 Kayes      femme  1998     708735
## 10 Kayes      femme  2009    1020247
## # ... with 26 more rows
```

Unificateur: `unite` (2)

Considérons que l'on veuille revenir à la forme initiale avec l'année et le sexe croisés et répartis entre les colonnes. La fonction `unite` intervient pour ce genre de cas.

```
adm3_pop_1998_2009_tidy_reg <-
  # Jeu de données de départ
  adm3_pop_1998_2009_tidy %>%
  # Spécification du niveau d'agrégation
  group_by(admin1_nom, sexe, annee) %>%
  # Opération d'agrégation
  summarize(population = sum(population, na.rm = TRUE)) %>%
  # Unification
  unite(
    # Nom de la nouvelle colonne
    col = "groupe",
    # Liste des variables (catégorielles) à combiner
    sexe, annee,
    # Signe séparateur
    sep = "_"
  ) %>%
  # Transformation en format large
  spread(
    # Nom de la variable clé: les catégories à séparer en colonnes
    key = groupe,
    # Nom de la variable de valeur: les valeurs correspondants aux catégories à séparer
    value = population
  )
knitr::asis_output("\\normalsize")
```

Unificateur: unite (3)

Rappelons que l'esthétique aussi compte quand il s'agit des tableaux à afficher dans les rapports et diapositives. Renommons donc nos colonnes.

```
# Jeu de données de départ
adm3_pop_1998_2009_tidy_reg %>%
  # Renommer les variables
  rename(`Région` = admin1_nom,
         `Femme (1998)` = femme_1998,
         `Femme (2009)` = femme_2009,
         `Homme (1998)` = homme_1998,
         `Homme (2009)` = homme_2009
  )
```

```
## # A tibble: 9 x 5
## # Groups:   Région [9]
##   Région     `Femme (1998)` `Femme (2009)` `Homme (1998)` `Homme (2009)`
##   <chr>               <dbl>          <dbl>          <dbl>          <dbl>
## 1 Bamako             501329         902723         514967         907643
## 2 Gao                174330         270685         167212         271619
## 3 Kayes              708735        1020247         678262         992829
## 4 Kidal               18175          31404          19932          36335
## 5 Koulikouro         785314        1219803         771384        1199409
## 6 Mopti              732467        1030578         714972        1008277
## 7 Segou              846990        1173313         834308        1148338
## 8 Sikasso            911673        1339803         889923        1304655
## 9 Tombouctou         226594         335116         216025         335889
```

Section 4

Conclusion

Conclusion

Nous venons de voir que **tidyr**, tout comme **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.