Tutoriel dplyr

Tarik HAKAM

10/12/2020

Introduction

Le package **dplyr** fait parti du package *tidyverse*. (Garmendia, n.d.)

Il s'agit d'une extension facilitant le traitement et la manipulation de données contenues dans une ou plusieurs tables, qu'il s'agisse de data frame ou de tibble (tableaux de données).

Elle propose une syntaxe claire et cohérente, sous formes de verbes, pour la plupart des opérations de ce type.

Par ailleurs, les fonctions de **dplyr** sont en général plus rapides que leur équivalent sous R de base, elles permettent donc de traiter des données de grande dimension.

dplyr part du principe que les données sont tidy.

Les fonctions de l'extension peuvent s'appliquer à des tableaux de type data.frame ou tibble, et elles retournent systématiquement un tibble.

Chargement de la librarie dplyr

Au préalable, nous aurons éxécuter la fonction ***install.packages("dplyr")*** qui permet de télécharger le package **dplyr** et de l'installer sur notre machine.

Cette commande n'est à exécuter qu'une fois.

Puis exécuter la fonction **library("dplyr")** qui permet ensuite de charger le package et de rendre les fonctionnalités de celui-ci disponibles (cette fonction est à exécuter à chaque fois que l'on ouvre RStudio).

```
library("dplyr")
```

Après ces étapes indispensables, nous allons charger le dataset "iris" à l'aide de la la commande **data()**.

C'est à partir de ces données que nous réaliserons nos tests et que nous les manipulerons à l'aide des différents commandes disponibles de la librairie.

```
data("iris")
```

Sélection de colonnes à partir du dataset "iris"

A travers l'exécution de la commande précédente, nous avons chargé le dataset sur notre environnement.

Nous allons maintenant afficher le 6 premières lignes d'un data frame en fonction des variables que nous souhaitons à l'aide de la commande **select()**.

Nous allons afficher à partir de la table *iris*, les colonnes :

- Petal.Length
- Petal.Width
- Species

```
head(select(iris, Petal.Length, Petal.Width, Species))
```

```
##   Petal.Length Petal.Width Species
## 1          1.4         0.2  setosa
## 2          1.4         0.2  setosa
## 3          1.3         0.2  setosa
## 4          1.5         0.2  setosa
## 5          1.4         0.2  setosa
## 6          1.7         0.4  setosa
```

Sélection d'une rangée de colonnes

Toujours à l'aide de la commande **select()**, nous allons afficher plusieurs colonnes, mais cette fois-ci sans les nommer

Nous allons à partir de la commande suivante :

select(data, colonne3:colonne5)

afficher les colonnes 3 à 5, en excluant de fait la 1ère et la 2ème colonnes ainsi que toutes celles postérieures à la 5ème colonnes.

Ci-dessous, la même commande en utilisant les données de la base de données *iris*.

```
head(select(iris, Petal.Length:Species))
```

```
##   Petal.Length Petal.Width Species
## 1          1.4         0.2  setosa
## 2          1.4         0.2  setosa
## 3          1.3         0.2  setosa
## 4          1.5         0.2  setosa
## 5          1.4         0.2  setosa
## 6          1.7         0.4  setosa
```

Exclure des variables depuis la commande selection

De la même manière que précédemment, nous pouvons sélection l'ensemble des variables (colonnes) de la base de données *iris*, tout en excluant des colonnes par la commande suivante :

```
head(select(iris, -Sepal.Length, -Sepal.Width))
```

```
##   Petal.Length Petal.Width Species
## 1          1.4         0.2  setosa
## 2          1.4         0.2  setosa
## 3          1.3         0.2  setosa
## 4          1.5         0.2  setosa
## 5          1.4         0.2  setosa
## 6          1.7         0.4  setosa
```

Le symbole "-" suivi sans espace du nom de la variable se traduit par l'exclusion de cette même variable du tableau nouvellement affiché.

Utilisation de pattern pour selectionner des variables

Ici, nous allons voir que l'usage de pattern peut être très utile de façon à sélectionner ou à exclure des variables en fonction de nos besoins.

- La pattern **contains('text')**, permet de sélectionner l'ensemble des colonnes contenant le texte écrit entre quotes.
- La pattern **starts_with('text')**, permet de sélectionner l'ensemble des colonnes dont l'intitulé commence par le texte écrit entre quotes.
- La pattern **ends_with('text')**, permet de sélectionner l'ensemble des colonnes dont l'intitulé se termine par le texte écrit entre quotes.

```
head(select(iris, contains('Petal')))
```

```
##   Petal.Length Petal.Width
## 1          1.4         0.2
## 2          1.4         0.2
## 3          1.3         0.2
## 4          1.5         0.2
## 5          1.4         0.2
## 6          1.7         0.4
```

```
head(select(iris, starts_with('Petal')))
```

```
##   Petal.Length Petal.Width
## 1          1.4         0.2
## 2          1.4         0.2
## 3          1.3         0.2
## 4          1.5         0.2
## 5          1.4         0.2
## 6          1.7         0.4
```

```
head(select(iris, ends_with('Length')))
```

```
##   Sepal.Length Petal.Length
## 1          5.1          1.4
## 2          4.9          1.4
## 3          4.7          1.3
## 4          4.6          1.5
## 5          5.0          1.4
## 6          5.4          1.7
```

Sélection de lignes

- Sélection à l'aide de la commande **filter(data, variable == 'text')** des lignes contenant la valeur 'text' de la colonne 'variable' à partir de la base de donnée.

```r
head(filter(iris, Species == 'virginica'))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1          6.3         3.3          6.0         2.5 virginica
## 2          5.8         2.7          5.1         1.9 virginica
## 3          7.1         3.0          5.9         2.1 virginica
## 4          6.3         2.9          5.6         1.8 virginica
## 5          6.5         3.0          5.8         2.2 virginica
## 6          7.6         3.0          6.6         2.1 virginica
```

- Sélection à l'aide de la commande **filter(data, variable != 'text')** de l'ensemble des lignes excluant la valeur 'text' de la colonne 'variable' à partir de la base de donnée.

```r
head(filter(iris, Species != 'setosa' & Species != 'versicolor'))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1          6.3         3.3          6.0         2.5 virginica
## 2          5.8         2.7          5.1         1.9 virginica
## 3          7.1         3.0          5.9         2.1 virginica
## 4          6.3         2.9          5.6         1.8 virginica
## 5          6.5         3.0          5.8         2.2 virginica
## 6          7.6         3.0          6.6         2.1 virginica
```

- Sélection à l'aide de la commande **filter(data, variable1 == 'text1', variable2 < x)** de l'ensemble des lignes incluant la valeur 'text1' de la colonne variable1 et dont la valeur variable2 est inférieur à x à partir de la base de donnée.

```r
head(filter(iris, Species == 'virginica', Petal.Length < 6))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1          5.8         2.7          5.1         1.9 virginica
## 2          7.1         3.0          5.9         2.1 virginica
## 3          6.3         2.9          5.6         1.8 virginica
## 4          6.5         3.0          5.8         2.2 virginica
## 5          4.9         2.5          4.5         1.7 virginica
## 6          6.7         2.5          5.8         1.8 virginica
```

Affichage de nouvelles variables

Ici, nous allons créer et afficher de nouvelles variables à l'aide de la commande ci-desssous.

Il sera créer la variable **"Petal.Area** à partir du produit des variables *Petal.Width* et *Petal.Length*.

Cette commande ne permettra pas d'intégrer une nouvelle colonne mais affichera uniquement le fruit de nos besoins.

```r
head(iris %>%
  mutate(Petal.Area = Petal.Width * Petal.Length,
         Sepal.Area = Sepal.Width * Sepal.Length) %>%
  select(Species, Petal.Area, Sepal.Area))
```

```
##   Species Petal.Area Sepal.Area
## 1  setosa       0.28      17.85
## 2  setosa       0.28      14.70
## 3  setosa       0.26      15.04
## 4  setosa       0.30      14.26
## 5  setosa       0.28      18.00
## 6  setosa       0.68      21.06
```

Création et ajout de nouvelles variables à la base de données iris

Contrairement à la commande précédente, nous allons ajouter les colonnes **Petal.Area** et **Sepal.Area** à la data frame *iri* à l'aide de la commande **iris$Nouvelle_Variable**.

```
iris$Petal.Area <- iris$Petal.Width * iris$Petal.Length
iris$Sepal.Area <- iris$Sepal.Width * iris$Sepal.Length
```

Utilisation des fonction group_by, summarize et arrange

Nous allons finir ce tutoriel par les fonctions **group_by()**, **summarise()** et **arrange()**.

Prenons, l'exemple ci-dessous.

- **group_by()** va nous permettre de regouper par toutes les valeurs de la base de donnée *iris* par espace.
- **summurise()** va nous permettre d'afficher par exemple la médiane par espèce des valeurs de la colonne *Petal.Length* de la base de donnée *iris*.
- **arrange()** va nous permettre d'ordonner dans l'ordre croissant les valeurs des médianes par espèce de la base de donnée *iris*. A noter que si nous voulions ordonner ces médianes dans un ordre décroissant, nous aurions ajouter un signe moins (-) devant la variable souhaitée sans espace.

```
iris %>%
  group_by(Species) %>%
  summarise(Median.Petal.Length = median(Petal.Length)) %>%
arrange(Median.Petal.Length)
```

```
## # A tibble: 3 x 2
##   Species    Median.Petal.Length
##   <fct>                    <dbl>
## 1 setosa                    1.5
## 2 versicolor                4.35
## 3 virginica                 5.55
```

Bibliographie

Garmendia, Alfonso. n.d. "R for Life Sciences. Chapter 7, Dplyr and Tidyr: Tidyverse Packages to Manage Data." http://personales.upv.es/algarsal/R-tutorials/07_Tutorial-7_R-dplyr-tidyr.html.