Fonctionnalités de l'opérateur `[` du package data.table

par Mamadou YAUCK et Oumaima ELKHOUDAOUI

2017-04-18

Table des matières

Introduction

Le logiciel `R` offre différentes structures de données telles que les vecteurs, les listes, les matrices, les data frames. Cette dernière structure, le data frame, offre une manière compréhensible d'organiser, de visualiser et d'accéder aux données. La sélection d'observations et de variables requiert la connaissances des différents éléments de la structure. Par ailleurs, l'aggrégation de l'information statistique contenue dans un objet de la classe `data.frame` fait appel à un minimum de connaissances de fonctions d'aggrégation telles que `tapply`, `table`, `aggregate`. Idéalement, il serait beaucoup plus intéressant de disposer d'objets de structures similaires au data frame qui effectuent le travail de manipulation de jeu de données de manière beaucoup plus **simple**.

La présente fiche s'intéresse à `data.table`, 9^{e} package le plus téléchargé dans `R` en date du 18 avril 2017 (consulter https://www.rdocumentation.org/trends) . Introduit en 2006 par Matt Dowle, le package est une version avancée de la fonction `data.frame` issue des fonctionnalités de base de `R` offrant des manipulations simples de jeux de données. En particulier, la fiche décrit les différentes possibilités de l'opérateur `[`. De façon spécifique, la fiche décrira les fonctionnalités suivantes :

- Sélection d'un sous-échantillon d'une base de données (lignes, colonnes)
- Aggrégation d'information

La présentation se fera sous une approche comparative à travers le jeu de données `iris`, qui a la structure interne ainsi décrite.

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Ce jeu de données contient des informations sur trois espèces de plantes de la famille des iridacées (setosa, virginica et versicolor). Pour chacune de ces espèces, des informations sur la longueur et la largeur du pétale et du sépale ont été extraites à partir d'un échantillon de 50 plants. Un aperçu du jeu de données est ainsi présenté.

```
head(iris,5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

```
tail(iris,5)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

Possibilités de l'installation de base : `data.frame`

Les jeux de données qu'on a eu à manipuler dans le cours de `R` pour Scientifique sont de la classe `data.frame`. Le jeu de données `iris` n'est pas une exception.

```
DF<-iris
class(DF)
```

```
## [1] "data.frame"
```

La syntaxe de manipulation de l'objet `DF` de la classe `data.frame` est $DF[i, j]$, avec

- `i` : l'indexation de la ligne du jeu de données
- `j` : l'indexation de la colonne du jeu de données

Sur un objet de la classe `data.frame`, plusieurs manipulations sont possibles : sélection de lignes, de colonnes, aggrégations d'informations, ordonnancement de variables, etc.

Sélection de données

Il est ainsi possible d'effectuer les manipulations usuelles sur un jeu de données. Par exemple, pour sélectionner les deux premières observations du jeu de données `iris`, la commande suivante est exécutée.

```
DF[1:2,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
```

Si nous voulons sélectionner les deux premières observations des variables `Sepal.Length` et `Sepal.Width`, la commande suivante est exécutée.

```
DF[1:2,c(1,2)]
```

```
##   Sepal.Length Sepal.Width
```

```
## 1          5.1         3.5
## 2          4.9         3.0
```

ou

```
DF[1:2, c("Sepal.Length","Sepal.Width")]
```

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
```

Si nous voulons sélectionner les lignes (ou observations) pour lesquelles `Sepal.Width` est strictement supérieur à 4, la commande suivante est exécutée.

```
DF[DF$Sepal.Width>4,]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 16          5.7         4.4          1.5         0.4  setosa
## 33          5.2         4.1          1.5         0.1  setosa
## 34          5.5         4.2          1.4         0.2  setosa
```

Aggrégation de données

Lorsqu'on manipule un objet de classe `data.frame`, l'aggrégation des données fait appel à des fonctions telles que `apply`, `tapply`, `aggregate`, `table`... Supposons que l'on veuille obtenir la moyenne de `Sepal.Width` selon la variable `Species`.

```
tapply(DF$Sepal.Width, INDEX = DF$Species, FUN = mean)
```

```
##     setosa versicolor  virginica
##      3.428      2.770      2.974
```

Si on veut obtenir les moyennes de `Sepal.Width` et `Sepal.Length` selon la variable `Species`, la commande suivante est exécutée.

```
aggregate(DF[, c("Sepal.Width", "Sepal.Length")], by = list(Species=DF$Species), FUN = mean)
```

```
##      Species Sepal.Width Sepal.Length
## 1     setosa       3.428        5.006
## 2 versicolor       2.770        5.936
## 3  virginica       2.974        6.588
```

Supposons que l'on s'intéresse au nombre d'observations par espèces d'iridacées, la commande suivante est exécutée.

```
table(DF$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

Les manipulations importantes effectuées sur des objets de la classe `data.frame` requièrent donc une connaissance minimale de fonctions d'aggrégations. Est-il possible d'interroger les données sans connaitre les fonctions décrites plus haut ? La réponse est affirmative ! Le package `data.table` offre une alternative remarquable dans la manipulation de jeux de données.

Package intégré : `data.table`

L'utilisation des fonctionnalités des objets de la classe `data.table` passe par le chargement du package `data.table`.

```
library(data.table)
```

La création d'un objet de la classe `data.table` s'effectue exactement de la même façon qu'un objet de la classe `data.frame`. Par exemple, supposons que l'on veuille créer le jeu de données de lanceurs vu en classe. Nous allons le faire avec les deux fonctions.

```
desPlus <- data.frame(de1=1:10,de2=10:19, lanceur = rep(c("Luc", "Kim"), each = 5))
class(desPlus)
```

```
## [1] "data.frame"
```

```
desPlus <- data.table(de1=1:10,de2=10:19, lanceur = rep(c("Luc", "Kim"), each = 5))
class(desPlus)
```

```
## [1] "data.table" "data.frame"
```

Un fait intéressant est qu'un objet de la classe `data.table` est également un objet de la classe `data.frame`. Cela implique que les manipulations effectuées sur un objet de la classe `data.frame` sont aussi valables avec un objet de la classe `data.table`. **La réciproque n'est pas valable !**

Dans la suite de la présentation, nous allons travailler avec le jeu de données `iris` en prenant soin de créer un objet `data.table` avec la commande suivante.

```
DT<-data.table(DF)
str(DT)
```

```
## Classes 'data.table' and 'data.frame':   150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

La syntaxe de manipulation de l'objet `DT` de la classe `data.table` est parfaitement décrite par le schéma.

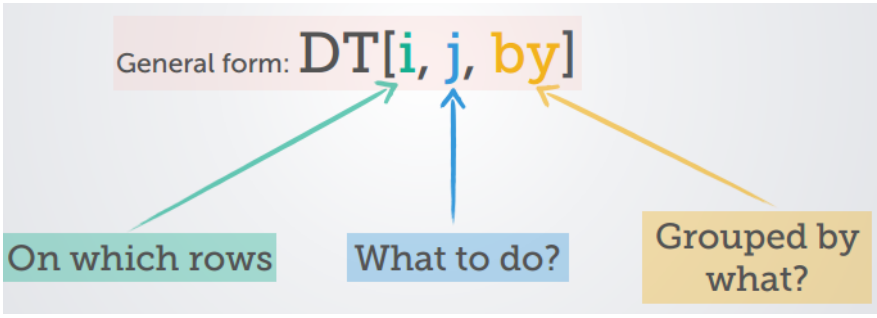

FIGURE 1 – Forme générale de la syntaxe data.table

- `i` : l'indexation de la ligne du jeu de données
- `j` : ce qu'il faut faire sur la colonne du jeu de données
- `by` : indique comment regrouper le résultat

Les manipulations sur l'objet `DF` de la classe `data.frame` sont également valables avec l'objet `DT` de la classe `data.table` pour les raisons d'identité décrites un peu plus haut.

Sélection de données

Pour sélectionner les deux premières observations du jeu de données `DT`, la commande suivante est exécutée.

```
DT[1:2,,]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1:          5.1         3.5          1.4         0.2  setosa
## 2:          4.9         3.0          1.4         0.2  setosa
```

Mieux encore, il est possible de n'indiquer que les lignes selectionnées, sans mettre les virgules.

```
DT[1:2]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1:          5.1         3.5          1.4         0.2  setosa
## 2:          4.9         3.0          1.4         0.2  setosa
```

Quand il s'agit de sélectionner une colonne du jeu de données, il suffit de placer une virgule avant d'indiquer la variable. Supposons que l'on veuille afficher les cinq premières observations de la variable `Sepal.Length`.

```
DT[1:5,c(Sepal.Length)]
```

```
## [1] 5.1 4.9 4.7 4.6 5.0
```

Nous remarquons que le résultat voulu n'est pas obtenu ! Dans un objet de classe `data.table`, le symbole `c()` offre en sortie un vecteur. Pour avoir un tableau de valeurs en sortie, il faut utiliser le symbole `.()` ou `list()`.

```
DT[1:5,.(Sepal.Length)]
```

```
##    Sepal.Length
## 1:          5.1
## 2:          4.9
## 3:          4.7
## 4:          4.6
## 5:          5.0
```

Si nous voulons sélectionner les deux premières observations des variables `Sepal.Length` et `Sepal.Width`, la commande suivante est exécutée.

```
DT[1:2,list(Sepal.Length,Sepal.Width)]
```

```
##    Sepal.Length Sepal.Width
## 1:          5.1         3.5
## 2:          4.9         3.0
```

Si nous voulons sélectionner les lignes (ou observations) pour lesquelles `Sepal.Width` est strictement supérieur à 4, la commande suivante est appropriée.

```
DT[Sepal.Width>4]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1:          5.7         4.4          1.5         0.4  setosa
## 2:          5.2         4.1          1.5         0.1  setosa
```

```
## 3:          5.5         4.2          1.4         0.2  setosa
```

Aggrégation de données

L'une des différences les plus remarquables dans la manipulation des deux types d'objets se trouve dans l'aggrégation des données. En effet, pour les objets de la classe `data.frame`, il faut faire appel à des fonctions telles que `tapply` ou `table` afin de résumer l'information. Cependant, pour les objets de la classe `data.table`, tout le travail se fait entre les `[` ; il n'est donc pas nécessaire de se préoccuper de l'utilisation des fonctions `tapply` ou `table` ! Supposons que l'on veuille calculer la moyenne de `Sepal.Width`.

```
DT[,mean(Sepal.Width)]
```

```
## [1] 3.057333
```

Maintenant, supposons que l'on veuille obtenir la moyenne de `Sepal.Width` selon la variable `Species`.

```
DT[,mean(Sepal.Width),by=Species]
```

```
##       Species    V1
## 1:     setosa 3.428
## 2: versicolor 2.770
## 3:  virginica 2.974
```

Il est possible de labelliser la variable de sortie (V1) par le label `Moyenne`

```
DT[,.(Moyenne=mean(Sepal.Width)),by=Species]
```

```
##       Species Moyenne
## 1:     setosa   3.428
## 2: versicolor   2.770
## 3:  virginica   2.974
```

Il est possible d'obtenir les moyennes `Sepal.Width` et `Sepal.Length` selon la variable `Species` de façon très simple.

```
DT[,.(Moy_Width=mean(Sepal.Width),Moy_Length=mean(Sepal.Length)),by=Species]
```

```
##       Species Moy_Width Moy_Length
## 1:     setosa     3.428      5.006
## 2: versicolor     2.770      5.936
## 3:  virginica     2.974      6.588
```

Pour aller un peu plus loin dans le résumé de l'information, nous allons mettre le focus sur la tabulation des données. Pour les objets de la classe `data.frame`, la fonction `table` compte les fréquences. Dans le cas des objets de la classe `data.table`, il est inutile de se préoccuper de la fonction précédente. Dans un `data.table`, il y'a une variable spéciale `.N` qui s'occupe de compter le nombre de lignes. Si l'argument `by=` est utilisé, la variable représente alors le nombre de lignes correspondant à la variable de regroupement.

D'abord, supposons que l'on veuille faire le décompte du nombre d'observations dans le jeu de données. La commande suivante est exécutée.

```
DT[,.N]
```

```
## [1] 150
```

Supposons que l'on s'intéresse au nombre d'observations par espèces d'iridacées, la commande suivante est exécutée.

```
DT[,.N,by=Species]
```

```
##       Species  N
## 1:     setosa 50
## 2: versicolor 50
## 3:  virginica 50
```

Maintenant on s'intéresse, pour chaque espèce, au nombre d'observations pour lesquelles `Sepal.Width` est strictement supérieur à 3. Le travail est effectué de façon simple.

```
DT[Sepal.Width>3,.N,by=Species]
```

```
##       Species  N
## 1:     setosa 42
## 2: versicolor  8
## 3:  virginica 17
```

La variable `.N` fait partie d'un ensemble communément appelé symboles spéciaux. Cette possibilité permet une rapidité dans l'exécution des commandes et une flexibilité dans la manipulation des jeux de données.

Analyse comparative `data.frame` et `data.table`

Après avoir parcouru les principales options offertes par les deux types d'objets, il convient de relever les points suivants :

- Dans un objet de classe `data.table`, on se passe de syntaxes de la forme `DF$nom_de_la_variable` très souvent utilisées lorsqu'on travaille avec un objet de classe `data.frame` ; cela nous évite des codes longs, illisibles et facilement entachés d'erreurs ;
- La manipulation d'un objet de classe `data.table` est beaucoup plus intuitive dans l'utilisation des arguments. De manière caricaturale, il s'agit de **prendre le jeu de données DT, choisir les observations** i**, ensuite calculer** j **suivant le groupe** by .

Par ailleurs, l'un des avantages majeurs du package `data.table` est la **rapidité d'exécution**. Bien que ce point ne soit pas illustré dans la présente fiche, il convient de souligner que l'utilisation d'un objet de classe `data.table` - lorsque l'on manipule de gros jeux de données - , offre un avantage considérable du point de vue du temps d'exécution. Ces remarques sont parfaitement résumées dans ce qu'il convient d'appeler la philosphie de `data.table` : **des commandes concises, directes, rapides et efficientes en mémoire**.

Certains auteurs ont relevé des faiblesses du package `data.table` au regard des fonctions d'aggrégation. Toutefois, ils soulèvent que ces manquements sont bien balancés par la puissance du package. Il est possible d'en apprendre un peu plus en suivant le lien https://www.r-bloggers.com/efficient-aggregation-and-more-using-data-table/ .

Références bibliographiques

https://cran.r-project.org/web/packages/data.table/data.table.pdf

http://datatable.r-forge.r-project.org/datatable-intro.pdf

https://github.com/Rdatatable/data.table/wiki

https://s3.amazonaws.com/assets.datacamp.com/img/blog/data+table+cheat+sheet.pdf

http://user2014.stat.ucla.edu/files/tutorial_Matt.pdf

https://www.r-bloggers.com/intro-to-the-data-table-package/