Créer un jeu de données agrégées en calculant des statistiques descriptives pour une variable selon les niveaux d'une autre variable

par Sébastien Matte et Marc-André Thivierge

2014-03-16

Table des matières

Comparaison de la fonction `aggregate` et du package `plyr`

1. Présentation de la tâche à effectuer et du jeu de données Iris

Cette fiche montre comment calculer des statistiques descriptives pour un jeu de données agrégé en R. Une comparaison de deux méthodes permettra de déterminer laquelle est plus simple d'utilisation pour calculer le minimum, la moyenne et le maximum d'une variable pour un regroupement d'observation d'un jeu de données. La première méthode utilise des fonctions de base dans le progiciel R, alors que la deuxième méthode recourt l'installation d'un package.

Pour comparer les deux méthodes, un jeu de données très connu sur les iris de Fisher est utilisé. Ce jeu de données contient des données morphologiques sur 3 différentes espèces d'iris ; *setosa*, *virginica* et *versicolor*. Pour chacune de ces espèces, on a mesuré sur un échantillon de 50 plants la longueur et la largeur du pétale et du sépale. Pour de plus amples informations concernant ce jeu de données, on peut visiter ce lien web : http://en.wikipedia.org/wiki/Iris_flower_data_set

Pour visualiser ce jeu de données dans R, on a seulement besoin de taper `print(iris)` dans la console.

```
head(iris)
```

```
##   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
## 6          5.4         3.9          1.7         0.4  setosa
```

De plus, on peut connaître sa structure interne à l'aide de la commande suivante :

```r
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 ...
```

> On remarque que le jeu de données iris est un tableau de données de 150 observations avec 5 variables dont 4 qui sont numériques et une de type facteur (*Species*).

À l'aide de ce jeu de données, on pourra calculer le minimum, la moyenne et le maximum de la longueur du sépale (variable *Sepal.Length*) en fonction des différentes espèces (variable *Species*) avec la fonction de base de `R` `aggregate` et avec les fonctions `ddply` et `summarize` du package `plyr`.

Bref, notre objectif sera d'obtenir ce tableau agrégé des statistiques descriptives sur la longueur du sépale en fonction de l'espèce d'iris.

```
##      Species min   moy max
## 1     setosa 4.3 5.006 5.8
## 2 versicolor 4.9 5.936 7.0
## 3  virginica 4.9 6.588 7.9
```

2. Méthode de base en `R` : fonction `aggregate`

2.1 Explication générale de `aggregate` :

Rappelons qu'on veut déterminer, à partir d'un jeu de données, le minimum, le maximum et la moyenne de la longueur du sépale des différentes espèces d'iris. En premier lieu, on utilisera `aggregate` une fonction de base existant dans `R`.

Cette fonction s'écrit de façon générale : `aggregate(x, by, FUN, .)` où `x`, `by`, `FUN` sont les arguments principaux, on peut aussi ajouter des options comme `na.rm` pour enlever ou non les valeurs manquantes du calcul de la statistique descriptive.

La fonction `aggregate` est utile pour calculer une statistique descriptive d'une ou plusieurs variables provenant d'un tableau de données selon des sous-ensembles d'observation. Ce sont les combinaisons de modalités des variables incluses dans l'argument `by`, sous forme de liste, qui détermine ces sous-ensembles d'observation.

Par exemple, pour la variable *Species* dans le jeu de données *iris*, les modalités sont *virginica*, *versicolor* et *setosa*. La fonction `aggregate` fera une boucle itérative pour les variables entrées dans l'argument `x` pour les différentes combinaisons de modalités afin de déterminer la statistique descriptive entrée dans l'argument `FUN`. Par exemple, on peut entrer `mean` dans l'argument `FUN` pour déterminer la moyenne des différentes espèces de fleur pour une variable quelconque entrée dans l'argument `x`.

2.2 Utilisation de `aggregate` avec les données sur les iris de Fisher

Voici un exemple pour bien comprendre l'utilisation de la fonction `aggregate`. D'abord, on veut déterminer le minimum de la longueur du sépale selon les différentes espèces. * L'argument **`x`** est la variable *Sepal.Length* * L'argument **by** est la variable *Species* * L'argument **FUN** est la fonction **`min`**

```r
minimum <- aggregate(x = iris$Sepal.Length, by = list(iris$Species), FUN=min)
print(minimum)
```

```
##      Group.1   x
## 1     setosa 4.3
## 2 versicolor 4.9
## 3  virginica 4.9
```

On obtient donc le miminum de la longueur du sépale pour les différentes espèces.

De manière plus rapide et plus simplifié, on peut utiliser une formule `y ~ x` où `y` est en fonction de `x` dans la fonction `aggregate`. Ainsi, on peut déterminer le minimum, la moyenne et le maximum avec la formule simplifiée.

```
minimum <- aggregate(Sepal.Length ~ Species, data= iris, FUN=min)
moyenne <- aggregate(Sepal.Length ~ Species, data= iris, FUN=mean)
maximum <- aggregate(Sepal.Length ~ Species, data= iris, FUN=max)
```

```
str(minimum)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 2 3
##  $ Sepal.Length: num  4.3 4.9 4.9
```

```
str(moyenne)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 2 3
##  $ Sepal.Length: num  5.01 5.94 6.59
```

```
str(maximum)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 2 3
##  $ Sepal.Length: num  5.8 7 7.9
```

On obtient bien 3 jeux de données de 3 observations avec 2 variables, soit *Species* pour les espèces et *Sepal Length* pour la statistique calculée.

On remarque que les titres des deuxièmes colonnes ne sont pas ceux désirés. On renomme les titres des colonnes à l'aide de la fonction `colnames`.

```
colnames(minimum)[2] <- "min"
colnames(moyenne)[2] <- "moy"
colnames(maximum)[2] <- "max"
```

Les jeux de données ont maintement les noms de colonne désirés :

```
# Exemple de résultat pour le calcul du maximum

print(maximum)
```

```
##      Species max
## 1     setosa 5.8
## 2 versicolor 7.0
## 3  virginica 7.9
```

Finalement, on doit fusionner les 3 tableaux de données minimum, moyenne et maximum par la variable commune *Species*. On peut simplement utiliser la fonction `merge` pour y parvenir.

```
final <- merge(merge(minimum,moyenne), maximum)
final
```

```
##      Species min   moy max
## 1     setosa 4.3 5.006 5.8
## 2 versicolor 4.9 5.936 7.0
## 3  virginica 4.9 6.588 7.9
```

Notons que la fonction `merge` s'écrit de manière à ne prendre que deux tableaux de données ou que deux objets en arguments d'où la raison de faire un `merge` dans un `merge`. En effet, l'écriture de cette fonction est `merge(x, y, .)` où x et y sont des *data frames* et que le résultat de cette fonction donne un *data frame*.

> Voir l'aide de `R` pour obtenir plus d'informations à ce sujet en tapant `help(merge)` dans la console.
>
> L'utilisation de l'option `by` n'est pas nécessaire, parce que les 3 tableaux de données (minimum, moyenne et maximum) ont la même variable pour les différentes espèces avec un nom commun qui est *Species*.

3 - Fonction `merge` versus package `plyr`

3.1 - Utilisation du package `plyr`

Une alternative à l'utilisation de la fonction `aggregate` se retrouve dans le package `plyr` de `R`. C'est Hadley Wickham qui en est l'auteur. Pour l'installer et l'utiliser, il suffit de lancer dans un éditeur de `R` les commandes suivantes :

```
install.packages("plyr")
library(plyr)
```

> La version du package `plyr` utilisée est la version 1.8.1 publiée le 26 février 2014. À noter que la version 2.11.0 de `R`, ou une plus récente, est nécessaire au bon fonctionnement du package `plyr`.

Ce package contient plusieurs fonctions très utiles pour résoudre certains problèmes de programmation. Par exemple, `plyr` offre des pistes de solutions intéressantes lorsque l'on veut appliquer des calculs ou des fonctions à des parties précises d'un jeu de données pour ensuite rassembler les résultats. Ce genre de stratégie est nommé *split-apply-combine*.

Les fonctions permettant ce type d'utilisation sont sous la forme `ddply`. La première lettre représente le type d'objet que la fonction prend en entrée, alors que la deuxième lettre représente le type d'objet que la fonction retourne. Dans ce cas-ci, la fonction prend un *data frame* en entrée et retourne un nouveau *data frame*. Un exemple d'utilisation est présenté à la section 3.2 de ce document.

Pour de l'information générale sur l'ensemble des fonctions du package `plyr`, il suffit de consulter l'aide interactive de `R` :

```
help(plyr)
```

Aussi, le manuel de référence est disponible sur http://cran.r-project.org/web/packages/plyr/plyr.pdf.

Pour la tâche présente, soit de calculer des statistiques descriptives de base d'un jeu de données en fonction d'un regroupement d'observations, les fonctions `ddply` et `summarize` seront utilisées en conjonction. À noter que la fonction `summarise` effectue exactement la même tâche. Ce n'est en fait qu'une utilisation britannique de ce terme.

3.2 - Fonction `summarize`

La fonction `summarize` est une fonction qui permet de calculer des statistiques sur l'ensemble des observations d'un jeu de données. Son écriture générale est : `summarize(.data, ...)` , où : * **`.data`** est un *data frame* * `...` correspond aux variables que l'on désire obtenir, de la forme nom_variable = valeur **(i.e. : moyenne = mean(Sepal.Length))**.

Elle prend en entrée un objet de type *data frame* et retourne un nouvel objet, aussi de type *data frame*. L'objet résultant de la fonction `summarize` contient une seule observation et chacune des variables correspond aux statistiques qui ont été demandées. Par exemple, si l'on désire obtenir le minimum, le maximum et la moyenne de la longueur du sépale pour l'ensemble des iris, il suffit de lancer la commande suivante :

```
exemple <- summarize(iris, mininimum = min(Sepal.Length),
                     maximum = max(Sepal.Length), moyenne = mean(Sepal.Length))

str(exemple)
```

```
## 'data.frame':    1 obs. of  3 variables:
##  $ mininimum: num 4.3
##  $ maximum  : num 7.9
##  $ moyenne  : num 5.84
```

> `summarize` retourne bel et bien un objet de type *data frame* avec une seule observation et trois variables, soit une variable par statistique calculée.

Contrairement à la fonction `aggregate`, il est donc possible, à l'aide de la fonction `summarize`, de calculer plus d'une statistique sur les données et de donner un nom significatif à chacune de ces variables. Cependant, on ne peut pas appliquer ces fonctions à plusieurs sous-parties d'un jeu de données. Pour subvenir à ce besoin, le package `plyr` permet d'utiliser en conjonction les fonctions `summarize` et `ddply`.

3.3 - Conjonction de `ddply` et `summarize`

L'écriture générale pour la fonction `ddply` est la suivante : `ddply(.data, .variables, .fun = NULL, ...)`, où : * **`.data`** est un jeu de données de format *data frame* * **`.variables`** est une formule ou un vecteur de caractères spécifiant les variables servant à regrouper les observations * **`.fun`** est la fonction qui doit être appliquée à chacun des sous-ensembles du jeu de données

> L'aide de `R` peut être consultée pour obtenir plus d'informations sur l'ensemble des options de la fonction `ddply`.

En combinant les fonctions `ddply` et `summarize`, on se retrouve à calculer des statistiques descriptives à un jeu de données en fonction d'un regroupement d'observations. Pour l'exemple des iris de Fisher, on peut facilement calculer les minimums, les moyennes et les maximums de la longueur du sépale des différentes espèces d'iris. Voici ce que l'on obtient :

```
resultat <- ddply(.data = iris, .variables = .(Species), .fun = summarize,
                  min = min(Sepal.Length), moy = mean(Sepal.Length),
                  max=max(Sepal.Length))

print(resultat)
```

```
##      Species min   moy max
## 1     setosa 4.3 5.006 5.8
## 2 versicolor 4.9 5.936 7.0
## 3  virginica 4.9 6.588 7.9
```

4 - Comparaison des deux méthodes

En somme, pour calculer le minimum, la moyenne et le maximum de la longueur du sépale (variable *Sepal.Length*) en fonction des différentes espèces (variable *Species*), il faut effectuer plusieurs fois la fonction `aggregate` pour chacune des statistiques descriptives désirées. En comparaison, un seul appel à la fonction `ddply`, en combinaison avec la fonction `summarize`, est nécessaire. Avec `aggregate`, on doit aussi renommer chacune des colonnes de façon manuelle et par la suite combiner autant de `merge` que l'on a de statistique à calculer. Il est donc plus simple d'utiliser les fonctions `ddply` et `summarize` lorsque l'on désire obtenir beaucoup d'information sur un grand nombre de variables.

Notons qu'il aurait été possible d'obtenir les statistiques désirées par un seul appel à la fonction `aggregate`. En effet, nous aurions pu donner en entrée à `aggregate` une fonction qui calcule toutes les statistiques désirées et les retourne dans un vecteur, comme suit :

```
aggregate(Sepal.Length ~ Species, data = iris,
          FUN = function(x) c(min = min(x), moy = mean(x), max = max(x)))
```

Cependant, le résultat obtenu aurait tout de même demandé une mise en forme afin d'avoir l'allure souhaitée au départ.

Sources d'information

Nous espérons que cette introduction au package `plyr` et aux fonctions `ddply` et `summarize` vous aura été utile pour vos futures analyses.

Merci de votre attention !

Bibliographie

- Aide interactive de `R`
- Information sur le jeu de données des iris de Fisher : http://en.wikipedia.org/wiki/Iris_flower_data_set
- Manuel de référence du package `plyr` : http://cran.r-project.org/web/packages/plyr/plyr.pdf
- Matloff, Norman. 2011. The Art of R Programming : A Tour of Statistical Software Design. San Fancisco. No Starch Press Inc., 373 p.