Transformer un jeu de données d'une mise en forme longue vers une mise en forme large en R

par Laurence Desbois-Bédard et Sarah-Anne Savard

2014-03-16

Table des matières

Différents formats de jeux de données **1**

La fonction `reshape` **2**
Guide d'utilisation : 3

La fonction `dcast` **3**
Guide d'utilisation : 4

Comparaison des deux méthodes **4**

Pour plus d'informations **5**

Cette fiche présente deux fonctions pour changer le format d'un jeu de données d'une mise en forme longue vers une mise en forme large. On y décrit d'abord les particularités de chacune de ces mises en forme à l'aide d'exemples. On utilise ensuite la fonction `reshape`, qui provient de la librairie standard de `R`, afin de changer le format du jeu de données. Une option alternative est ensuite présentée pour effectuer la même tâche. Il s'agit de la fonction `dcast`, qui provient de la librairie `reshape2`, disponible au http://cran.r-project.org/web/packages/reshape2/index.html. Enfin, une comparaison des deux méthodes est présentée.

Afin d'illustrer la différence entre les mises en forme longue et large, on utilise un extrait du jeu de données des iris de Fisher à titre d'exemple. Ce jeu de données compte 150 observations et 5 variables. Chaque observation représente un iris. Quatre variables numériques continues ont été mesurées sur chacune des fleurs : la longueur et largeur d'un sépale (`Sepal.Length` et `Sepal.Width`) et la longueur et la largeur d'un pétale (`Petal.Length` et `Petal.Width`). De plus, une variable catégorique nominale à trois modalités (`Species`) donne l'espèce de l'iris, soit setosa, versicolor ou virginica. On a gardé les observations 1, 51 et 101 dans les exemples ci-dessous.

Différents formats de jeux de données

Pour qu'un jeu de données soit dit de format long, une même observation doit y revenir plusieurs fois. Une variable d'identification permet de reconnaître l'observation. Dans l'exemple ci-dessous, chaque fleur revient à quatre reprises et est identifiée par la variable `UnitID` :

```
##     UnitID    Species   VariableID Value
## 1        1     setosa Sepal.Length   5.1
## 51      51 versicolor Sepal.Length   7.0
## 101    101  virginica Sepal.Length   6.3
## 151      1     setosa  Sepal.Width   3.5
## 201     51 versicolor  Sepal.Width   3.2
## 251    101  virginica  Sepal.Width   3.3
## 301      1     setosa Petal.Length   1.4
```

```
## 351     51 versicolor Petal.Length   4.7
## 401    101  virginica Petal.Length   6.0
## 451      1     setosa  Petal.Width   0.2
## 501     51 versicolor  Petal.Width   1.4
## 551    101  virginica  Petal.Width   2.5
```

L'information contenue dans la colonne `Species` est simplement l'espèce de l'iris observé. Elle est répétée à chaque fois que l'observation revient dans le jeu de données. La colonne `VariableID` donne le nom de la variable pour laquelle on a la valeur dans la colonne `Value`. Ce format est dit long car il compte davantage de lignes, mais moins de colonnes, qu'un même jeu de données au format large. En effet, on a ici besoin de 600 lignes pour synthétiser l'information obtenue auprès de 150 fleurs, mais de seulement 4 colonnes alors que le jeu de données contient 5 variables.

Il est important de remarquer dès à présent que les colonnes d'un jeu de données de mise en forme longue ont un rôle particulier. D'une part, certaines colonnes permettent d'identifier l'observation. Il s'agit bien sûr de la colonne donnant le numéro de l'observation, mais aussi de toutes celles qui contiennent directement de l'information au sujet de cette observation. Dans l'exemple, il s'agit des colonnes `UnitID` et `Species`. On les appelera **colonnes identifiantes**. D'autre part, on a toujours une colonne qui donne un nom de variable et une autre qui donne la valeur de cette variable pour l'observation. On comprend qu'il s'agit des colonnes `VariableID` et `Value` de l'exemple. On les nommera **colonne des variables** et **colonne des valeurs** respectivement.

Une telle mise en forme est couramment rencontrée dans la pratique statistique lorsque les données étudiées sont appariées et qu'elles proviennent d'une étude longitudinale. On peut penser, par exemple, à des données récoltées à plusieurs moments dans le temps sur un même individu. Dans un tel contexte, certaines variables seront **identifiantes**, comme le nom de la personne et son sexe, et ne varieront pas d'un moment à l'autre de l'étude. D'autres variables, comme la pression artérielle et la température corporelle, peuvent changer. Ces variables ne sont donc pas **identifiantes**. On ajoutera une ligne au jeu de données pour chaque nouvelle mesure prise.

Il arrive cependant que la mise en forme longue ne soit pas commode et qu'on souhaite changer le format du jeu de données pour le ramener sous la forme large. Ce format est celui qu'on rencontre le plus souvent : une rangée par observation et une colonne par variable.

Le jeu de données des iris est généralement présenté sous cette forme. Voici le même extrait qu'à l'exemple précédent, mais cette fois-ci sous la forme large :

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 51           7.0         3.2          4.7         1.4 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
```

Ce jeu de données compte maintenant 150 lignes et 5 colonnes. On voit que les **colonnes identifiantes** sont toujours présentes d'une certaine manière : la colonne `UnitID` correspond maintenant au numéro d'observation et la colonne `Species` est une variable du jeu de données. Chaque mot apparaissant dans la **colonne des variables** dans le format long est devenu le nom d'une variable dans le format large. Enfin, les valeurs de la **colonne des valeurs** ont tout simplement été organisées pour apparaître au bon endroit dans le format large.

La fonction `reshape`

La fonction `reshape` permet de changer le format d'un jeu de données en `R`, que ce soit d'une mise en forme longue vers une mise en forme large ou l'inverse. Pour l'utiliser dans le cas qui nous intéresse, il importe de bien identifier les **colonnes identifiantes** et la **colonne des variables** du jeu de données au format long. Cela simplifie grandement l'utilisation de la fonction.

Pour transformer un jeu de données d'une mise en forme longue vers une mise en forme large, les arguments les plus utiles sont `data`, `direction`, `v.names`, `timevar`, `idvar` et `drop`. Les arguments `new.row.names`, `sep` et `split` peuvent aussi être utilisés afin de changer les noms des lignes et des colonnes du jeu de données final, mais on recommande plutôt d'effectuer ces modifications par la suite.

Guide d'utilisation :

- `data` = le jeu de données, au format `data frame`
- `direction` = `'wide'`
- `v.names` = les variables dans le jeu de données au format long qui doivent être mises en colonnes dans le format large (optionnel)
- `timevar` = la **colonne des variables**
- `idvar` = les **colonnes identifiantes** (un vecteur)
- `drop` = la ou les variables à retirer du jeu de données avant la transformation

Par exemple, pour passer du format long au format large dans le jeu de donnée des iris, on utilise la fonction ainsi :

```
iris_large1 <- reshape(iris_long,
                       direction = "wide",
                       timevar = "VariableID",
                       idvar = c("Species","UnitID"))
```

Voici le résultat :

```
##     UnitID    Species Value.Sepal.Length Value.Sepal.Width
## 1        1     setosa                5.1               3.5
## 51      51 versicolor                7.0               3.2
## 101    101  virginica                6.3               3.3
##     Value.Petal.Length Value.Petal.Width
## 1                  1.4               0.2
## 51                 4.7               1.4
## 101                6.0               2.5
```

On remarque que la varible `UnitID` n'est plus utile car elle correspond au numéro d'observation. De plus, les noms des variables de groupes sont maintenant précédés du nom de la **colonne des valeurs** (`Value`). Pour rendre le résultat semblable à celui présenté plus haut, il suffit de faire quelques opérations simples : extraire, renommer et permuter les colonnes désirées. Par exemple,

```
iris_large1 <- iris_large1[,c(3:6,2)]
colnames(iris_large1) <- c('Sepal.Length','Sepal.Width','Petal.Length',
                           'Petal.Width','Species')
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 51           7.0         3.2          4.7         1.4 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
```

La fonction `dcast`

Comme on l'a déjà mentionné, une alternative à la fonction de base existe dans la librairie `reshape2`. Il s'agit des fonctions `dcast` et `acast`. Les deux permettent de transformer un `data frame` d'une mise en forme longue vers une mise en forme large, mais la première retourne un `data frame` alors que la seconde retourne

un `array`. On présente ici la fonction `dcast`, mais l'utilisation de `acast` est similaire, dans les cas où l'objet de sortie doit avoir plus de 2 dimensions.

Guide d'utilisation :

- `data` = le jeu de données, au format `data frame`
- `formula` = **colonnes identifiantes** ~ **colonne des variables** (il existe d'autres notations pour les formules, voir `help(dcast)`)
- `fun.aggregate` = fonction à utiliser pour synthétiser l'information s'il y a plus d'une observation par cellule après la transformation (`mean`, `median`, `sum`, etc.)
- `...` = les arguments requis par la fonction précédente, s'il y a lieu
- `drop` = la ou les variables à retirer du jeu de données avant la transformation
- `value.var` = la **colonne des valeurs** (optionnel)

Il est à noter que si la **colonne des valeurs** n'est pas spécifiée, `R` peut généralement la trouver par lui-même. De plus, l'ordre des **colonnes identifiantes** est important et une attention particulière doit y être portée. Les **colonnes identifiantes** sont séparées par le symbole `+` dans une formule standard. D'autres arguments peuvent aussi être utilisés (voir `help(dcast)`).

Pour effectuer la transformation du jeu de données des iris, on procède ainsi :

```
library(reshape2) # Chargement de la librairie

iris_large2 <- dcast(data = iris_long,
                     formula = UnitID + Species ~ VariableID,
                     value.var = 'Value')
```

Voici le résultat :

```
##     UnitID    Species Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1        1     setosa          5.1         3.5          1.4         0.2
## 51      51 versicolor          7.0         3.2          4.7         1.4
## 101    101  virginica          6.3         3.3          6.0         2.5
```

Tout comme avec la fonction `reshape`, la variable `UnitID` n'est plus utile. Par contre, les noms des variables ne sont plus précédés du nom de la **colonne des valeurs**. Cela simplifie les opérations pour obtenir un jeu de données tel que celui présenté dans les sections précédentes.

Comparaison des deux méthodes

Les deux fonctions présentées permettent de transformer un jeu de données du format long vers le format large. Cependant, l'un des problèmes de la fonction `reshape` est qu'elle permet aussi d'effectuer la transformation inverse, ce qui complexifie sa syntaxe. De plus, ses arguments ont des noms peu intuitifs en dehors du cadre d'une étude longitudinale (`timevar` par exemple). Ces faiblesses ont été corrigées dans les fonctions de la librairie `reshape2`. En effet, on y trouve des fonctions différentes pour les deux opérations : `melt` pour passer du format large au format long et `dcast` ou `acast` pour passer du format long au format large. Qui plus est, les arguments sont plus simples à utiliser et la formule ajoute de nombreuses possibilités pour l'utilisateur. Enfin, l'argument `fun.aggregate` permet de considérer toutes les valeurs du jeu de données au format long, alors qu'avec la fonction `reshape`, seulement la première occurence est considérée et un avertissement est émis (voir la description de l'argument `timevar` dans la fiche d'aide de `reshape`). En somme, les fonctions de la librairie `reshape2` sont plus faciles à utiliser et ont des fonctionnalités que `reshape` ne possède pas.

Pour plus d'informations

Comme pour toutes les fonctions en `R`, on peut consulter la fiche d'aide de ces fonctions en tapant `help(reshape)` ou `help(dcast)` dans la console de `R`.

La page de Sean Anderson et cet article de Thiago G. Martins offrent une explication vulgarisée de l'utilisation des fonctions de la librairie `reshape2`.