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

par Mondji Herbert Monwanou et Simon Rioux

2014-03-16

Table des matières

Introduction

L'installation de base de R offre plusieurs fonctions pour la manipulation des jeux de données. Plusieurs *packages* ont également été créés pour une réalisation plus pratique de tâches spécifiques dans la manipulation des bases de données.

Le but de cette fiche est de montrer comment modifier un jeu de données de la mise en forme large vers la mise en forme longue en utilisant deux techniques. La première utilise la fonction `reshape` de R, et la deuxième utilise des fonctions du *package* reshape2.

Pour des fins d'exemple, nous utiliserons le jeu de données des iris de Fisher ou Anderson, qui est disponible dans R en utilisant la fonction `iris`. Ce jeu de données donne les mesures en centimètres des variables *Sepal.Length* (longueur de la sépale), *Sepal.Width* (largeur de la sépale), *Petal.Length* (longeur du pétale) et *Petal.Width* (largeur du pétale) pour une cinquantaine de fleurs de chacune des 3 espèces *setosa*, *versicolor* et *virginica*. Des informations sur ce jeu de données se trouvent sur sa fiche d'aide de R (exécuter le code `?iris`) et sur sa page wikipédia.

En voici un aperçu :

```
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
```

```r
tail(iris)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 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
```

Le jeu de données comporte une ligne par observation, le but sera de le transformer afin qu'il comporte une ligne par donnée. Voici un aperçu du jeu de données que l'on désire :

```
##   UnitID Species   VariableID Value
## 1      1  setosa Sepal.Length   5.1
## 2      2  setosa Sepal.Length   4.9
## 3      3  setosa Sepal.Length   4.7
## 4      4  setosa Sepal.Length   4.6
## 5      5  setosa Sepal.Length   5.0
## 6      6  setosa Sepal.Length   5.4
```

```
##     UnitID   Species  VariableID Value
## 595    145 virginica Petal.Width   2.5
## 596    146 virginica Petal.Width   2.3
## 597    147 virginica Petal.Width   1.9
## 598    148 virginica Petal.Width   2.0
## 599    149 virginica Petal.Width   2.3
## 600    150 virginica Petal.Width   1.8
```

Méthode 1 : fonction `reshape` de R :

Description

La fonction `reshape` de R fait partie du package *stats*. Elle permet de transformer un jeu de données de la mise en forme large (*wide*) vers la mise en forme longue (*long*) et vice versa. Cette fonction comporte un grand nombre d'arguments qu'il est possible de spécifier :

Paramètre	Description
`data`	*Data frame* concerné
`varying`	Liste de variables de la mise en forme large qui seront "empilées" dans la mise en forme longue pour former une seule ou quelques variable(s). Il s'agit généralement d'une liste de noms de variables, mais ce paramètre peut être une matrice ou un vecteur de noms.
`v.names`	On nomme ici la ou les variable(s) créées dans `varying`.
`timevar`	Variable du format long permettant de différencier les observations multiples d'un même groupe ou individu. On crée donc une variable qui permettra d'identifier de quelle variable du jeu de données initial la donnée provient.
`idvar`	Variables du format long permettant d'identifier les groupes ou individus uniques. On crée donc une ou plusieurs variable(s) qui permettra d'identifier l'observation d'où provient la donnée.
`ids`	Valeurs utilisées pour la nouvelle variable nommée dans `idvar`.
`times`	Valeurs utilisées pour la nouvelle variable nommée dans `timevar`.
`drop`	Vecteur des noms de variables à exclure de la remise en forme.

Paramètre	Description
`direction`	Chaîne de caractère indiquant la direction de la mise en forme : `"wide"` pour un format large, `"long"` pour un format long.
`new.row.names`	Valeur logique indiquant s'il faut créer de nouveaux noms de lignes à partir des valeurs de `id` et de `time` lorsque l'on passe d'une mise en forme longue à une mise en forme large.
`sep`	Ce caractère précise le séparateur utilisé dans les noms de variables du format large quand la fonction `reshape` tente de déterminer les valeurs de `v.names` et `times` lorsque l'on passe d'une mise en forme large à une mise en forme longue.
`split`	Liste de trois composants : `regexp`, `include` et `fixed` qui fournissent une interface améliorée pour le découpage des noms de variables (voir la fiche d'aide pour plus de détails).

Adapté de : R l'essentiel, Joseph Adler, édition Pearson Éducation France (2011), page 217

À titre d'exemple, voici la procédure qui permet de transformer le jeu de données *iris* de la mise en forme large à la mise en forme longue :

```
iris_long <- reshape(iris, direction = "long",
                     idvar = "UnitID",
                     ids = rownames(iris),
                     varying = c("Sepal.Length", "Sepal.Width",
                                 "Petal.Length", "Petal.Width"),
                     v.names = "Value", timevar = "VariableID",
                     times = colnames(iris)[1:4])
str(iris_long)
```

```
## 'data.frame':    600 obs. of  4 variables:
##  $ Species   : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ VariableID: chr  "Sepal.Length" "Sepal.Length" "Sepal.Length" "Sepal.Length" ...
##  $ Value     : num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ UnitID    : chr  "1" "2" "3" "4" ...
##  - attr(*, "reshapeLong")=List of 4
##   ..$ varying:List of 1
##   .. ..$ Value: chr  "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
##   .. ..- attr(*, "v.names")= chr "Value"
##   .. ..- attr(*, "times")= chr  "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
##   ..$ v.names: chr "Value"
##   ..$ idvar  : chr "UnitID"
##   ..$ timevar: chr "VariableID"
```

Il est possible de constater, dans la structure interne de iris_long, que les paramètres utilisés lors de la création d'un objet avec `reshape` sont stockés dans les attributs de l'objet.

```
head(iris_long)
```

```
##                Species   VariableID Value UnitID
## 1.Sepal.Length  setosa Sepal.Length   5.1      1
## 2.Sepal.Length  setosa Sepal.Length   4.9      2
## 3.Sepal.Length  setosa Sepal.Length   4.7      3
## 4.Sepal.Length  setosa Sepal.Length   4.6      4
## 5.Sepal.Length  setosa Sepal.Length   5.0      5
## 6.Sepal.Length  setosa Sepal.Length   5.4      6
```

Ce n'est pas tout à fait le jeu de données que l'on désire. Nous allons donc changer l'ordre des colonnes et enlever les noms de lignes :

```
iris_long <- iris_long[, c(4,1,2,3)]
rownames(iris_long) <- NULL
head(iris_long)
```

```
##   UnitID Species   VariableID Value
## 1      1  setosa Sepal.Length   5.1
## 2      2  setosa Sepal.Length   4.9
## 3      3  setosa Sepal.Length   4.7
## 4      4  setosa Sepal.Length   4.6
## 5      5  setosa Sepal.Length   5.0
## 6      6  setosa Sepal.Length   5.4
```

```
tail(iris_long)
```

```
##     UnitID   Species  VariableID Value
## 595    145 virginica Petal.Width   2.5
## 596    146 virginica Petal.Width   2.3
## 597    147 virginica Petal.Width   1.9
## 598    148 virginica Petal.Width   2.0
## 599    149 virginica Petal.Width   2.3
## 600    150 virginica Petal.Width   1.8
```

Le jeu de données résultant est sous le format désiré, soit une ligne par donnée. Pour identifier l'observation associée à la donnée, la variable "UnitID" a été créée. Pour identifier la variable du jeu de données initial associée à la donnée du jeu final, la variable "VariableID" a été créée.

Pour plus de détails sur la fonction `reshape`, on peut consulter la fiche d'aide de R ou la page web correspondante ou ce lien.

Méthode 2 : utilisation du package *reshape2* :

Présentation du package *reshape2*

Le package R `reshape2` est une version plus récente du package R *reshape* développé par Hadley Wickham. Son site officiel est le suivant. Il est également possible de consulter une publication en lien avec le package `reshape`, parue en 2007 dans le *Journal of Statistical Software*. Ce package permet surtout le remodelage des données. Ses deux principales fonctions sont la fonction `melt`, qui permet le passage d'un jeu de données de la mise en forme large à la mise en forme longue, et la fonction `cast`, qui permet de réaliser l'inverse. Un truc pour retenir le nom de la fonction `melt` est de penser à un bloc de métal qui fond et qui s'étire vers le bas.

Description de la fonction `melt`

L'usage général de cette fonction est :

```
melt(data, ..., na.rm = FALSE, value.name = "value")
```

où data peut être un objet R de type *data frame*, *array* ou *list*. Les variantes de la fonctions correspondant à chacune de ces types d'objet sont respectivement `melt.data.frame`, `melt.array` et `melt.list`. L'appel à la fonction `melt` permet de réaliser automatiquement les tâches de ces variantes en précisant les arguments appropriés, donc on utilise toujours `melt`, peu importe le type d'objet que l'on désire transformer.

Le jeu de données *iris* est un *data frame*. Ainsi pour `data.frame.melt` qui nous concerne ici, on doit préciser à `melt` : - la ou les variable(s) d'identification (*id.vars*) - la ou les variable(s) de mesure (*measure.vars*)

Si on ne précise qu'un seul de ces deux paramètres, la fonction attribuera le reste des variables du jeu de données à l'autre paramètre.

Si on ne spécifie aucun de ces paramètres, melt considère les variables de type caractère ou facteur comme des variables d'identification, et les autres comme des variables de mesure.

L'usage de la fonction `melt` pour un *data frame* est donc :

```
melt(data, id.vars = "", measure.vars = "", variable.name = "variable", ...,
     na.rm = FALSE, value.name = "value")
```

Les paramètres de `melt` dans ce cas sont :

Arguments	Description
`data`	Le *data frame* à transformer
`id.vars`	Vecteur de variable(s) d'identification pouvant être de type entier (indiquant la position des variables) ou caractère (noms des variables). Si on ne spécifie rien, R prend toutes les variables non spécifiées dans measure.vars.
`measure.vars`	Vecteur de variable(s) de mesure pouvant être de type entier (indiquant la position des variables) ou caractère (noms des variables). Si on ne spécifie rien, R prend toutes les variables non spécifiées dans id.vars. *NB : si ni* ***`id.vars`*** *ni* ***`measure.vars`*** *ne sont spécifiés, R prend les variables catégoriques comme variables d'identification et les variables numériques comme variables de mesure.*
`variable.name`	Nom de la variable devant contenir les noms des variables de mesure
`value.name`	Nom de la variable devant contenir les valeurs.
`na.rm`	Valeur logique permettant de gérer les valeurs manquantes. En mentionnant `TRUE`, on indique à R de supprimer les valeurs manquantes.

Source : page d'aide du package *reshape2*, suivre ce lien. Voir la page 7.

Exemple d'utilisation de la fonction `melt`

Avant toute chose, il faut, si ce n'est déjà fait, installer le package *reshape2* puis le charger en exécutant le code :

```
install.packages("reshape2")
library(reshape2)
```

Utilisons maintenant la fonction `melt` pour changer la mise en forme du jeu de données *iris* :

```
iris_long2 <- melt(iris, id.vars = "Species",
                   measure.vars = c("Sepal.Length", "Sepal.Width",
                                    "Petal.Length", "Petal.Width"),
                   variable.name = "VariableID", value.name = "Value")
str(iris_long2)
```

```
## 'data.frame':    600 obs. of  3 variables:
##  $ Species   : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ VariableID: Factor w/ 4 levels "Sepal.Length",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Value     : num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
```

Il est intéressant de remarquer que dans ce cas ci, les paramètres utilisés lors de la création d'un objet avec `melt` ne sont pas stockés dans les attributs de l'objet.

```
head(iris_long2)
```

```
##   Species   VariableID Value
## 1  setosa Sepal.Length   5.1
## 2  setosa Sepal.Length   4.9
## 3  setosa Sepal.Length   4.7
## 4  setosa Sepal.Length   4.6
## 5  setosa Sepal.Length   5.0
## 6  setosa Sepal.Length   5.4
```

```
tail(iris_long2)
```

```
##       Species  VariableID Value
## 595 virginica Petal.Width   2.5
## 596 virginica Petal.Width   2.3
## 597 virginica Petal.Width   1.9
## 598 virginica Petal.Width   2.0
## 599 virginica Petal.Width   2.3
## 600 virginica Petal.Width   1.8
```

Le résultat obtenu est proche de celui désiré. En effet, les variables présentes ont les bons attributs mais on note l'absence de colonne "id", ce qui est normal car la seule variable d'identification specifiée est "species". Pour avoir la colonne "id", il est possible de la juxtaposer manuellement avec la fonction `cbind` :

```
iris_long2 <- cbind(UnitID = as.numeric(rep(rownames(iris),4)), iris_long2)
head(iris_long2)
```

```
##   UnitID Species   VariableID Value
## 1      1  setosa Sepal.Length   5.1
## 2      2  setosa Sepal.Length   4.9
## 3      3  setosa Sepal.Length   4.7
## 4      4  setosa Sepal.Length   4.6
## 5      5  setosa Sepal.Length   5.0
## 6      6  setosa Sepal.Length   5.4
```

Une autre option aurait été de la créer dans le jeu de données original et de la spécifier dans `id.vars` lors de la transformation :

```
iris2 <- cbind(UnitID = as.integer(rownames(iris)), iris)
head(iris2)
```

```
##   UnitID Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1      1          5.1         3.5          1.4         0.2  setosa
## 2      2          4.9         3.0          1.4         0.2  setosa
## 3      3          4.7         3.2          1.3         0.2  setosa
## 4      4          4.6         3.1          1.5         0.2  setosa
## 5      5          5.0         3.6          1.4         0.2  setosa
## 6      6          5.4         3.9          1.7         0.4  setosa
```

```
iris_long3 <- melt(iris2, id.vars = c("UnitID", "Species"),
                   measure.vars = c("Sepal.Length", "Sepal.Width",
                                    "Petal.Length", "Petal.Width"),
                   variable.name = "VariableID", value.name = "Value")
```

```r
head(iris_long3)
```

```
##   UnitID Species   VariableID Value
## 1      1  setosa Sepal.Length   5.1
## 2      2  setosa Sepal.Length   4.9
## 3      3  setosa Sepal.Length   4.7
## 4      4  setosa Sepal.Length   4.6
## 5      5  setosa Sepal.Length   5.0
## 6      6  setosa Sepal.Length   5.4
```

```r
tail(iris_long3)
```

```
##     UnitID   Species  VariableID Value
## 595    145 virginica Petal.Width   2.5
## 596    146 virginica Petal.Width   2.3
## 597    147 virginica Petal.Width   1.9
## 598    148 virginica Petal.Width   2.0
## 599    149 virginica Petal.Width   2.3
## 600    150 virginica Petal.Width   1.8
```

Le jeu de données obtenu est exactement celui désiré.

Il faut noter qu'il aurait été possible d'omettre l'argument "measure.vars" : la fonction `melt` aurait pris toutes les variables non mentionnées dans l'argument "id.vars", ce qui correspond aux quatre variables mentionnées ci dessus. L'inverse est également vrai : il aurait été possible d'omettre l'argument "id.vars". Avec le jeu de données original *iris* non modifié, il aurait même été possible de ne spécifier ni l'argument "id.vars" ni l'argument "measure.vars". Dans ce cas, `melt` prend les variables catégoriques comme variables d'identification et les variables numériques comme variables de mesure :

```r
iris_long2a <- melt(iris, variable.name = "VariableID", value.name = "Value")
```

```
## Using Species as id variables
```

```r
head(iris_long2a)
```

```
##   Species   VariableID Value
## 1  setosa Sepal.Length   5.1
## 2  setosa Sepal.Length   4.9
## 3  setosa Sepal.Length   4.7
## 4  setosa Sepal.Length   4.6
## 5  setosa Sepal.Length   5.0
## 6  setosa Sepal.Length   5.4
```

Une autre façon d'utiliser la fonction `melt` aurait été en spécifiant les numéros de colonnes au lieu des noms de variables dans "id.vars" et dans "measure.vars".

```r
iris_long4 <- melt(iris2, id.vars = c(1, 6), measure.vars = c(2:5),
                   variable.name = "VariableID", value.name = "Value")
head(iris_long4)
```

```
##   UnitID Species   VariableID Value
## 1      1  setosa Sepal.Length   5.1
## 2      2  setosa Sepal.Length   4.9
## 3      3  setosa Sepal.Length   4.7
## 4      4  setosa Sepal.Length   4.6
## 5      5  setosa Sepal.Length   5.0
```

```
## 6      6  setosa Sepal.Length   5.4
```

On se retrouve avec le même résultat.

Il est possible d'en apprendre plus sur le package *reshape2* et sur la fonction `melt` en consultant cette page web et cette page web.

Comparaison des deux méthodes

Pour faciliter la comparaison, voici les deux programmes permettant de transformer le jeu de données *iris* de la mise en forme large vers la mise en forme longue :

```
iris_long <- reshape(iris, direction = "long", idvar = "UnitID",
                     ids = rownames(iris),
                     varying = c("Sepal.Length", "Sepal.Width",
                                 "Petal.Length", "Petal.Width"),
                     v.names = "Value", timevar = "VariableID",
                     times = colnames(iris)[1:4])
iris_long <- iris_long[, c(4,1,2,3)]
rownames(iris_long) <- NULL
```

```
iris_long2 <- melt(iris, id.vars = "Species",
                   measure.vars = c("Sepal.Length", "Sepal.Width",
                                    "Petal.Length", "Petal.Width"),
                   variable.name = "VariableID", value.name = "Value")
iris_long2 <- cbind(UnitID = as.numeric(rep(rownames(iris),4)), iris_long2)
```

Il faut d'abord noter qu'aucune des deux fonctions ne permet d'obtenir directement le résultat désiré avec le *data frame* d'origine. La fonction `reshape` permet de créer automatiquement en dernière position une colonne d'identification constituée des numéros des lignes du jeu de données original. Il faut ensuite réorganiser l'ordre des colonnes et renommer les lignes pour obtenir le résultat désiré. La fonction `melt` s'en tient à la variable d'identification spécifiée et ne crée pas de colonne supplémentaire, il faut alors la créer manuellement avant ou après la transformation.

Le *data frame* obtenu avec `melt` est un *data frame* ordinaire alors que celui obtenu avec `reshape` a plusieurs attributs supplémentaires (paramètres utilisés pour la transformation).

Selon nous, les arguments de la fonction `melt` sont nommés de façon plus explicite, ce qui rend cette fonction plus facile à utiliser. Par exemple, les arguments pour nommer les variables que l'on crée lors de la transformation sont nommés "variable.name" et "value.name".

De plus, il y a moins d'arguments à spécifier pour la fonction `melt`. Cela est peut-être dû au fait que cette fonction sert uniquement à transformer un jeu de données de la mise en forme large vers la mise en forme longue, tandis que la fonction `reshape` permet également la transformation inverse. Le fait qu'il y ait moins d'arguments à spécifier rend également la fonction plus facile à utiliser et diminue le risque de commettre une erreur.

En conclusion, nous croyons qu'il est préférable d'utiliser la fonction `melt` pour transformer un jeu de données de la mise en forme large vers la mise en forme longue étant donné sa plus grande facilité d'utilisation et le risque d'erreur plus faible.

Références

Note : les pages web référées dans cette fiche ont été archivées avec webcite le 11 mars 2014. Si un lien fourni dans cette fiche ne fonctionne plus, il est possible de consulter les pages web en suivant les liens suivants (placés en ordre de référence) :

- Informations sur le package *reshape2* : http://www.webcitation.org/6O0BdGGUj
- Page wikipédia du jeu de données *iris* : http://www.webcitation.org/6O0Bi5TLS
- Fiche d'aide de la fonction `reshape` : http://www.webcitation.org/6O0BlfGCO
- Page d'informations sur la fonction `reshape` : http://www.webcitation.org/6O4W1UZdy
- Site web du package *reshape2* : http://www.webcitation.org/6O0Bpg0XC
- Publication sur le package *reshape* : http://www.webcitation.org/6O0Bt8R9O
- Page d'aide du package *reshape2* : http://www.webcitation.org/6O0BwYEGK
- Informations sur le package *reshape2* : http://www.webcitation.org/6O0C26eJ6 et http://www.webcitation.org/6O0C3aast