Manipulations de base sur un jeu de données

Pascale Aubin et Pierre Gagnon

2016-04-07

Table des matières

Présentation des tâches à effectuer et du jeu de données utilisé

Cette fiche a pour but de montrer comment faire des manipulations de base sur un jeu de données en R. Lorsqu'on parle de manipulations de base, ce sont les manipulations qui sont courramment effectuées afin de bien analyser l'information d'un jeu de données comme :

- l'ajout de variables
- la sélection d'observations
- la sélection de variables

Il existe évidemment des outils dans le package `base` de R, qui seront présentés dans un premier temps, pour effectuer ces manipulations. Cette fiche détaillera aussi dans un deuxième temps l'utilisation de certaines fonctions du package `dplyr` qui permettent d'effectuer les mêmes manipulations.

Afin d'illustrer les différentes manipulations possibles, un même jeu de données sera utilisé tout au long de cette fiche. Il s'agit d'un jeu qui contient toutes les parties de tennis en simple masculin ayant eu lieu lors du tournoi de Wimbledon en 2013. Chaque observation représente une partie. Pour chaque partie, on possède les noms des joueurs qui s'affrontent, les résultats des différents sets et plusieurs statistiques de jeu. Voici les 2 premières observations :

```
head(tennis, n=2)
```

```
##    Player1  Player2 Round Result FNL.1 FNL.2 FSP.1 FSW.1 SSP.1 SSW.1 ACE.1
## 1 B.Becker A.Murray     1      0     0     3    59    29    41    14     5
## 2   J.Ward   Y-H.Lu     1      0     1     3    62    77    38    35    18
##   DBF.1 WNR.1 UFE.1 BPC.1 BPW.1 NPA.1 NPW.1 TPW.1 ST1.1 ST2.1 ST3.1 ST4.1
## 1     1    26    18     5     1    28    19    NA     4     3     2    NA
## 2     4    60    28    13     1    27    19    NA     7     4     6     6
##   ST5.1 FSP.2 FSW.2 SSP.2 SSW.2 ACE.2 DBF.2 WNR.2 UFE.2 BPC.2 BPW.2 NPA.2
## 1    NA    57    39    43    20    11     2    38    16    10     5    23
## 2    NA    67    85    33    31    12     3    57    32    15     2    46
##   NPW.2 TPW.2 ST1.2 ST2.2 ST3.2 ST4.2 ST5.2
```

```
## 1    17    NA     6     6     6    NA    NA
## 2    39    NA     6     6     7     7    NA
```

Ce jeu de données possède 114 observations et 42 variables. Les descriptions détaillées des différentes variables du jeu de données se trouvent sur le site : Tennis Major Tournament Match Statistics Data Set

Méthodes de base en R

Ajout de variables

Pour ajouter une nouvelle variable dans un *data.frame*, on peut d'abord penser à utiliser la fonction `cbind`. Elle permet de reprendre un jeu de données existant et d'y ajouter une ou plusieurs colonnes. On peut aussi simplement réutiliser la fonction `data.frame` et combiner le *data.frame* existant et la variable désirée. Par exemple, on veut créer une variable *Winner* qui contient le nom du gagnant pour chaque partie et l'ajouter au jeu de données «tennis». On ajoute alors une colonne qui contient le nom du gagnant au jeu de données «tennis». Ici, on doit d'abord réaliser une étape intermédiaire à l'aide d'un `ifelse` pour identifier correctement le gagnant à partir du résultat de la partie contenu dans la variable *Result*. Si la valeur de *Result* est 1, le joueur 1 (*Player1*) a gagné la partie. Les variables *Player1* et *Player2* sont converties en format caractère pour éviter d'obtenir des résultats inatendus, puisque ces deux variables sont des facteurs avec des niveaux différents.

```
gagnant <- ifelse(tennis$Result==1,as.character(tennis$Player1),as.character(tennis$Player2))
```

```
tennis <- cbind(tennis,Winner= gagnant)
```

Qui est équivalent à :

```
tennis <- data.frame(tennis,Winner=gagnant)
```

Voici l'extraction des variables d'intérêt pour les 5 premières parties :

```
head(tennis[,c(1:4,43)], n=5)
```

```
##     Player1         Player2 Round Result    Winner
## 1  B.Becker        A.Murray     1      0  A.Murray
## 2    J.Ward          Y-H.Lu     1      0    Y-H.Lu
## 3   N.Mahut         J.Hajek     1      1   N.Mahut
## 4 T.Robredo A.Bogomolov Jr.     1      1 T.Robredo
## 5   R.Haase       M.Youzhny     1      0 M.Youzhny
```

Une autre façon d'ajouter une variable est d'utiliser l'opérateur `$`. Il suffit d'indiquer avant cet opérateur le nom du *data.frame* auquel la variable sera ajoutée et après, le nom de la nouvelle variable. Il ne reste qu'à assigner des valeurs à la nouvelle variable ainsi créée.

```
tennis$Winner <- gagnant
```

```
head(tennis[,c(1:4,43)], n=5)
```

```
##     Player1         Player2 Round Result    Winner
## 1  B.Becker        A.Murray     1      0  A.Murray
## 2    J.Ward          Y-H.Lu     1      0    Y-H.Lu
## 3   N.Mahut         J.Hajek     1      1   N.Mahut
## 4 T.Robredo A.Bogomolov Jr.     1      1 T.Robredo
```

```
## 5   R.Haase       M.Youzhny     1      0 M.Youzhny
```

Sélection d'observations

On s'intéresse souvent à la conservation des observations pour lesquelles certaines variables prennent des valeurs spécifiques. Il est possible de faire une telle sélection avec des conditions logiques en R. Dans l'exemple du tennis, on s'intéresse aux trois dernières rondes du tournoi, soit les rondes 5, 6 et 7. Cette sélection nécessite l'utilisation de la variable *Round* qui indique la ronde du tournoi. On peut d'abord extraire du *data.frame* de départ les observations qui respectent cette condition. L'opérateur `%in%` permet de vérifier si la valeur d'une variable est comprise dans un ensemble de valeurs. Autrement dit, cela est équivalent à utiliser l'opérateur `|` (ou logique) en énumérant toutes les égalités possibles désirées. Pour conserver le résultat, on va l'assigner à un nouveau *data.frame*, ici appelé tennis2.

```
tennis2 <- tennis[tennis$Round==5|tennis$Round==6|tennis$Round==7,]
```

qui est équivalent à :

```
tennis2 <- tennis[tennis$Round %in% c(5,6,7),]
```

Il est aussi possible d'effectuer cette opération avec la fonction `subset` qui permet de spéficier une expression de sélection pour un jeu de données.

```
tennis2 <- subset(tennis, subset=Round>=5 )
```

Sélection de variables

Lorsqu'on présente des données, il est préférable de n'inclure que les variables nécessaires. Pour sélectionner les variables désirées, on peut directement énumérer les numéros correspondants aux colonnes voulues. Il est aussi possible d'extraire des colonnes en les nommant dans un vecteur entre les crochets `[,]` au niveau de la dimension 2 (pour faire référence aux colonnes). Le jeu de données dit final est contenu dans tennis3 :

```
tennis3 <- tennis2[,c(1:3,20:24,38:43)]
```

équivalent à :

```
tennis3 <- tennis2[,c("Player1", "Player2", "Round", "ST1.1", "ST1.2", "ST2.1", "ST2.2", "ST3.1", "ST3.1
```

Voici les observations du jeu final :

```
tennis3
```

```
##        Player1     Player2 Round ST1.1 ST1.2 ST2.1 ST2.2 ST3.1 ST3.2 ST4.1
## 108 F.Verdasco    A.Murray     5     6     4     6     3     1     6     4
## 109    L.Kubot  J.Janowicz     5     5     7     4     6     4     6    NA
## 110   D.Ferrer J.Del Potro     5     2     6     4     6     6     7    NA
## 111 N.Djokovic   T.Berdych     5     7     6     6     4     6     3    NA
## 112 J.Janowicz    A.Murray     6     7     6     4     6     4     6     3
## 113 N.Djokovic J.Del Potro     6     7     5     4     6     7     6     6
## 114 N.Djokovic    A.Murray     7     4     6     5     7     4     6    NA
##     ST4.2 ST5.1 ST5.2      Winner
## 108     6     5     7    A.Murray
## 109    NA    NA    NA  J.Janowicz
```

```
## 110    NA    NA    NA J.Del Potro
## 111    NA    NA    NA  N.Djokovic
## 112     6    NA    NA    A.Murray
## 113     7     6     3  N.Djokovic
## 114    NA    NA    NA    A.Murray
```

Le jeu contient alors 14 des 42 variables et 7 des 114 observations de départ.

Alternative : Package `dplyr`

Les fonctions du package `dplyr` ont été conçues pour simplifier les manipulations les plus communes que l'on doit appliquer sur un jeu de données. Il s'agit d'une suite du package `plyr` pour les **data.frame**, d'où le **d**. L'auteur est Hadley Wickham. Pour l'utiliser, il suffit de lancer les commandes suivantes :

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

Fonction mutate : ajout de variables

La fonction `mutate` de `dplyr` permet d'ajouter de nouvelles colonnes qui peuvent être fonction de colonnes déjà existantes. La syntaxe est `mutate(.data, ...)`.

- `.data` est une table
- `...` sont les expressions désirées pour créer les nouvelles variables

S'il y a plusieurs expressions qui créent des variables, il est possible de réutiliser les premières variables créées dans les expressions ultérieures. Dans l'exemple, on veut simplement créer la variable *Winner* donc on peut utiliser la fonction comme ceci (*gagnant* est le vecteur contenant le nom des gagnants créé précédement) et conserver le résultat dans le jeu tennis2**d** pour **dplyr** :

```
tennis2d <- mutate(tennis,Winner=gagnant)
```

```
head(tennis2d, n=2)
```

```
##    Player1  Player2 Round Result FNL.1 FNL.2 FSP.1 FSW.1 SSP.1 SSW.1 ACE.1
## 1 B.Becker A.Murray     1      0     0     3    59    29    41    14     5
## 2   J.Ward   Y-H.Lu     1      0     1     3    62    77    38    35    18
##   DBF.1 WNR.1 UFE.1 BPC.1 BPW.1 NPA.1 NPW.1 TPW.1 ST1.1 ST2.1 ST3.1 ST4.1
## 1     1    26    18     5     1    28    19    NA     4     3     2    NA
## 2     4    60    28    13     1    27    19    NA     7     4     6     6
##   ST5.1 FSP.2 FSW.2 SSP.2 SSW.2 ACE.2 DBF.2 WNR.2 UFE.2 BPC.2 BPW.2 NPA.2
## 1    NA    57    39    43    20    11     2    38    16    10     5    23
## 2    NA    67    85    33    31    12     3    57    32    15     2    46
##   NPW.2 TPW.2 ST1.2 ST2.2 ST3.2 ST4.2 ST5.2   Winner
## 1    17    NA     6     6     6    NA    NA A.Murray
## 2    39    NA     6     6     7     7    NA   Y-H.Lu
```

L'objet R obtenu avec la fonction `mutate` est de même classe que l'argument `.data`.

```
class(tennis)==class(tennis2d)
```

```
## [1] TRUE
```

Fonction filter : sélection d'observations

La fonction `filter` du package `dplyr` permet de sélectionner des observations (lignes) d'un objet R. La synthaxe de cette fonction est `filter(.data, ...)`.

- `.data` est une table
- `...` sont les expressions logiques qui permettent de faire la sélection des observations dans la table

Il s'agit d'une fonction similaire à la fonction `subset` utilisée plus haut. Par contre, avec la fonction `filter`, on peut indiquer plusieurs expressions de sélection séparées par des virgules. Pour qu'un observation soit sélectionnée, elle doit répondre à toutes les conditions listées dans `...`.

```
tennis3d<-filter(tennis2d, Round>=5)
```

Attention : Il y a une fonction `filter` qui existe déjà dans le package `base`. Or, le chemin de recherche commence par les packages chargés avec `library`, donc la fonction utilisée est celle du package `dplyr`.

Fonction select : sélection de variables

La fonction `select` de `dplyr` permet de sélectionner les variables d'intérêt. Lorsqu'une variable n'est pas explicitement écrite, la fonction `select` ne la conserve pas. La synthaxe est `select(.data, ...)` :

- `.data` est une table
- `...` noms des variables à conserver séparées par des virgules

De plus, pour conserver toutes les variables excepté une en particulier, on peut utiliser l'opérateur de soustraction `-` devant la variable à laisser tomber.

```
tennis4d<-select(tennis3d, Player1, Player2, Round, ST1.1, ST1.2, ST2.1, ST2.2, ST3.1, ST3.2, ST4.1, ST
```

Le jeu contient alors 14 variables et 7 observations .

Il existe plusieurs options afin d'alléger l'écriture de la sélection des variables. Ces quelques options supplémentaires sont des arguments compris dans les `...`

- `:` pour sélectionner toutes les variables entre 2 variables nommées de part et d'autre du `:`
- La fonction starts_with() pour sélectionner les variables qui débutent avec la chaîne de caractère mentionnée dans la fonction
- contains() pour sélectionner les colonnes qui contiennent une chaîne de caractère particulière dans le nom de la variable
- matches() pour sélectionner les colonnes qui correspondent à une expression
- one_of() pour sélectionner des colonnes qui sont dans un groupe de noms.

Voici une illustration de certaines de ces options pour alléger ce qui a été écrit plus haut :

```
tennis4d<-select(tennis3d, Player1, Player2, Round,starts_with("ST"), Winner)
```

```
tennis4d<-select(tennis3d, Player1, Player2, Round,contains("ST"), Winner)
```

```
tennis4d<-select(tennis3d, Player1, Player2, Round,ST1.1:ST5.1, ST1.2:ST5.2, Winner)
```

Ces trois jeux contiennent tous 14 variables et 7 observations.

Comparaison entre les méthodes de base et les fonctions de `dplyr`

Lorsque les manipulations ne concernent que très peu de conditions ou expressions pour sélectionner les observations ou variables, les deux méthodes sont comparables. Lorsque l'on veut sélectionner des observations selon plusieurs expressions, les fonctions du package `dplyr` permettent d'alléger l'écriture. Par exemple, l'énumération des conditions de sélection d'observations avec la fonction `filter` par son utilisation des virgules est moins lourde que l'enchaînement des conditions logiques dans les opérateurs `[,]` ou dans la fonction subset.

Lorsqu'on crée plus d'une variable, la fonction `mutate` nous permet d'écrire toutes les expressions les unes à la suite des autres dans un seul appel. Cette fonction permet aussi d'utiliser les variables créées en premier pour définir les dernières, contrairement à `$` et `cbind`. En effet, l'opérateur `$` nécéssite une assignation pour chacune des variables à créer et `cbind` ne permet pas d'utiliser les premières variables créées pour définir celles qui sont ultérieures.

De plus, lors de la sélection des variables, les nombreuses options de la fonction `select` ne nécessitent pas de spécifier parfaitement les noms de variables à conserver comme lors de l'utilisation des crochets `[,]`. Bref, les fonctions du package `dplyr` sont plus légères à écrire et à réviser, car leurs syntaxes ressemblent à une phrase et cela peut permettre à l'utilisateur d'être plus efficace qu'en utilisant les outils du package `base`. Il faut simplement penser à l'utiliser !

Bibliographie

- Référence pour information générale package `dplyr` (site web de l'auteur) : http://hadley.nz/index.html
- Référence fonction du package `dplyr` : https://cran.r-project.org/web/packages/dplyr/dplyr.pdf
- Références pour les options supplémentaires de la fonction `select` : http://genomicsclass.github.io/book/pages/dplyr_tutorial.html