Présentation du package 'dplyr' et de l'opérateur '%>%' (pipe)

Bastien Tran

.

Doctorant en Sciences de l'Information et de la Communication
à l'UVSQ

.

bastien.tran@gmail.com

16 juin 2017

Le package “dplyr”

Title A Grammar of Data Manipulation

Description A fast, consistent tool for working with data frame like objects, both in memory and out of memory.

Author Hadley Wickham [aut, cre], Romain Francois [aut], Lionel Henry [aut], Kirill Müller [aut], RStudio [cph, fnd]

URL `http://dplyr.tidyverse.org`, `https://github.com/tidyverse/dplyr`

Hadley Wickham, Romain Francois, Lionel Henry and Kirill Müller (2017). dplyr: A Grammar of Data Manipulation.

`https://github.com/rstudio/cheatsheets/raw/master/source/pdfs/data-transformation-cheatsheet.pdf`

`https://www.rstudio.com/wp-content/uploads/2016/01/data-wrangling-french.pdf`

Installation

```
devtools::install_github("tidyverse/dplyr")
```

. . . ou bien sûr:

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

Version 0.7.0

Depends R (>= 3.1.2)

Imports assertthat, *bindrcpp*, glue, *magrittr*, methods, pkgconfig, rlang (>= 0.1), R6, *Rcpp* (>= 0.12.6), *tibble* (>= 1.3.1), utils

LinkingTo Rcpp (>= 0.12.0), BH (>= 1.58.0-1), bindrcpp, plogr

??tidyverse

> *Le tidyverse est un jeu de packages partageant une même philosophie et conçus pour travailler ensemble.*

Le package *dplyr* en fait partie, tout comme *readr*, *tidyr*, *purrr* et *ggplot2*, ainsi que *tibble* et *magrittr* que nous aborderons plus loin.

Cet écosytème est complété par une douzaine d'autres packages pour notamment:

- Travailler avec certains types de vecteurs (*hms*, *stringr*, *lubridate* & *forcats*)
- Importer des données depuis diverses sources (*feather*, *haven*, *httr*, *jsonlite*, *readxl*, *rvest*, *xml2*)
- Modeliser (*modelr*, *broom*)

`http://tidyverse.org/`

Paradigme de dplyr

> *"A fast, consistent tool for working with data frame like objects. . . "*

dplyr propose un jeu de *verbes* qui constituent une "grammaire de la manipulation de données" pour notamment:

- construire de nouvelles variables à partir des variables existantes avec *mutate()*
- sélectionner des variables via leurs noms avec *select()*
- filtrer des enregistrements via leurs valeurs avec *filter()*
- résumer plusieurs valeurs sur une ligne avec *summarise()*

En fait *dplyr* est le prolongement de *plyr* (d pour data.frame ou data.table), il hérite ainsi d'une syntaxe plus explicite que *data.table* et peut offrir des performances comparables.

`https://stackoverflow.com/questions/21435339/`

Un air de SQL?

Il est également possible de:

- appliquer ces opérations à des données groupées par facteur ou variable avec *group_by()*
- réaliser des jointures entre tables avec *inner_join()*, *left_join()*, *right_join()*, *semi_join()*, *anti_join()*, *full_join()*.

"...both in memory and out of memory."

Car le package **dbplyr** permet de travailler avec des bases de données distantes en utilisant cette même grammaire (et donc le même code R).

Imports: assertthat, DBI (>= 0.5), dplyr (>= 0.5.0.9004), glue, methods, rlang (>= 0.1.0), tibble (>= 1.3.0.9007), R6, utils

`http://dplyr.tidyverse.org`

Les tibbles

Les *tibbles* (Müller & Wickham, 2017) sont des objets similaires aux data frames. Quelques différences notables:

- Ne convertit pas les types
- N'ajuste pas les noms de variables
- Pas de correspondance partielle sur les noms de variables
- Evaluation paresseuse et séquentielle
- Ne crée pas de *row.names*
- Recycle seulement les vecteurs de longueur 1
- N'affiche que les 10 premières lignes et un nombre de colonne qui tient dans l'écran
- Retourne une tibble quand échantillonnée (avec '[')

`http://r4ds.had.co.nz/tibbles.html`

`http://tibble.tidyverse.org/`

En route!

```
library(dplyr)
library(tibble)
data(iris)
```

```
glimpse(iris)
```

```
## Observations: 150
## Variables: 5
## $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.
## $ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.
## $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.
## $ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.
## $ Species      <fctr> setosa, setosa, setosa, setosa, setosa, setosa
```

```
options(tibble.print_max = 4, tibble.print_min = 4 )
```

Obtenir des résumés

```
summarise_all(iris, funs(mean))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1     5.843333    3.057333        3.758    1.199333      NA
```

```
summarise_if(iris, is.numeric, sd)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1    0.8280661   0.4358663     1.765298   0.7622377
```

```
summarise_at(iris, vars(Sepal.Length,Petal.Length),max)
```

```
##   Sepal.Length Petal.Length
## 1          7.9          6.9
```

Quelques opérations utiles

- Compter les observations

```
count(iris, Species)
```

- Dédoublonner avec *distinct()*

```
distinct(iris, Species)
```

- Réordonner ses observations avec *arrange()*

```
arrange(mtcars, desc(mpg))
```

- Ajouter des observations avec *add_row()*

```
add_row(iris, Sepal.Length = 4, Sepal.Width = 4, Petal.Length = 4, Peta
```

- Ajouter des colonnes avec *add_column()*

```
add_column(iris, new = rep(letters[1:15]))
```

Récupérer des observations aléatoirement

```
sample_frac(iris, 0.04, replace = TRUE)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 98           6.2         2.9          4.3         1.3 versicolor
## 36           5.0         3.2          1.2         0.2     setosa
## 66           6.7         3.1          4.4         1.4 versicolor
## 34           5.5         4.2          1.4         0.2     setosa
## 11           5.4         3.7          1.5         0.2     setosa
## 123          7.7         2.8          6.7         2.0  virginica
```

```
sample_n(iris, 5, replace = TRUE)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 96           5.7         3.0          4.2         1.2 versicolor
## 64           6.1         2.9          4.7         1.4 versicolor
## 142          6.9         3.1          5.1         2.3  virginica
## 107          4.9         2.5          4.5         1.7  virginica
## 56           5.7         2.8          4.5         1.3 versicolor
```

Récupérer des observations moins aléatoirement

```
filter(iris, Sepal.Length>7.5)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1          7.6         3.0          6.6         2.1 virginica
## 2          7.7         3.8          6.7         2.2 virginica
## 3          7.7         2.6          6.9         2.3 virginica
## 4          7.7         2.8          6.7         2.0 virginica
## 5          7.9         3.8          6.4         2.0 virginica
## 6          7.7         3.0          6.1         2.3 virginica
```

```
slice(iris, 10:15)
```

```
## # A tibble: 6 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl>  <fctr>
## 1          4.9         3.1          1.5         0.1  setosa
## 2          5.4         3.7          1.5         0.2  setosa
## 3          4.8         3.4          1.6         0.2  setosa
## 4          4.8         3.0          1.4         0.1  setosa
## # ... with 2 more rows
```

Sélectionner des variables

```
select(tbl_df(iris), Sepal.Length, Species)
```

```
## # A tibble: 150 x 2
##   Sepal.Length Species
##          <dbl>  <fctr>
## 1          5.1  setosa
## 2          4.9  setosa
## 3          4.7  setosa
## 4          4.6  setosa
## # ... with 146 more rows
```

```
select(tbl_df(iris), starts_with("Petal"))
```

```
## # A tibble: 150 x 2
##   Petal.Length Petal.Width
##          <dbl>       <dbl>
## 1          1.4         0.2
## 2          1.4         0.2
## 3          1.3         0.2
## 4          1.5         0.2
## # ... with 146 more rows
```

Manipuler des des variables

```
mutate(select(tbl_df(iris), starts_with("Petal")),
       Petal.Length.Width.Ratio = Petal.Length/Petal.Width)
```

```
## # A tibble: 150 x 3
##    Petal.Length Petal.Width Petal.Length.Width.Ratio
##           <dbl>       <dbl>                    <dbl>
## 1           1.4         0.2                      7.0
## 2           1.4         0.2                      7.0
## 3           1.3         0.2                      6.5
## 4           1.5         0.2                      7.5
## # ... with 146 more rows
```

On peut cibler les variables à modifier avec *mutate_all*, *mutate_at*, *mutate*.

Manipuler des variables

```
transmute(select(tbl_df(iris), starts_with("Petal")),
       Petal.Length.Width.Ratio = Petal.Length/Petal.Width)
```

```
## # A tibble: 150 x 1
##    Petal.Length.Width.Ratio
##                       <dbl>
## 1                       7.0
## 2                       7.0
## 3                       6.5
## 4                       7.5
## # ... with 146 more rows
```

mutate() et *transmute()* implémentent des fonctions vectorisées qui retournent un vecteur de même longueur que celui fournit en entrée.

Grouper des observations

dplyr peut se voir comme une spécialisation du package *plyr* vis à vis des data frames (ou putôt des *tibbles*). Ainsi nous pouvons mettre en oeuvre une stratégie d'analyse de données de type Split-Apply-Combine en groupant nos données selon une variable :

```
group_by(tbl_df(iris), Species)
```

```
## # A tibble: 150 x 5
## # Groups:   Species [3]
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl>  <fctr>
## 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
## # ... with 146 more rows
```

http://www.jstatsoft.org/v40/i01/

http://had.co.nz/plyr/

http://github.com/hadley/plyr

Grouper des observations

```
summarise_if(group_by(tbl_df(iris), Species),is.numeric,mean)
```

```
## # A tibble: 3 x 5
##      Species Sepal.Length Sepal.Width Petal.Length Petal.Width
##       <fctr>        <dbl>       <dbl>        <dbl>       <dbl>
## 1     setosa        5.006       3.428        1.462       0.246
## 2 versicolor        5.936       2.770        4.260       1.326
## 3  virginica        6.588       2.974        5.552       2.026
```

On peut bien sûr 'degrouper' la table

```
ungroup(g_iris)
```

L'opérateur '%>%'

Un pipe, fréquemment représenté par une barre verticale '|', renvoie la sortie d'une commande vers l'entrée d'une autre.

La librairie magrittr fournit cet opérateur (et quelques autres) pour R et permet d'écrire du code différement:

- la séquence d'opération se lit de gauche à droite
- on peut éviter l'appel de fonctions à l'intérieur d'autres fonction
- on peut diminuer l'usage de variables locales ou de fonction
- la séquence d'opération apparait très modulaire

Problème: dans R, '|' équivaut déjà à 'OR'. La librairie *magrittr* (Bache & Wickham, 2014) fournit un pipe qui 'n'en est pas vraiment un (*"This is not a pipe"*) sous la forme'%>%'.

`https://github.com/tidyverse/magrittr`

En pratique

Si nous voulons appliquer un filtre (Sepal.Length>5) sur quelques observations (10 à 15 par exemple) nous pouvons bien sûr écrire:

```
mysubset <- filter(slice(iris, 10:15), Sepal.Length>5)
```

Mais avec l'opérateur 'pipe' on peut également noter cette transformation ainsi:

```
mysubset <- iris %>% slice(10:15) %>% filter(Sepal.Length>5)
```

Et en faire autant avec des fonctions moins récentes:

```
x<-iris$Sepal.Length
log(sum(exp(x)), exp(1))
```

```
## [1] 11.21043
```

```
x %>% exp %>% sum %>% log(exp(1))
```

```
## [1] 11.21043
```

Une popularité grandissante

- forum, fils de discussion, blogs
- packages récents
  - *dplyr*
  - *leaflet* (cartes interactives dans *shiny*)
  - *tidytext* (text mining)
  - ...

```
m <- leaflet() %>%
  addTiles() %>%  # Add default OpenStreetMap map tiles
  addMarkers(lng=174.768, lat=-36.852, popup="The birthplace of R")
```

```
text_df %>% unnest_tokens(word, text)
```

Sources et ressources

- **dplyr**: Hadley Wickham, Romain Francois, Lionel Henry and Kirill Müller (2017). dplyr: A Grammar of Data Manipulation.
  `http://dplyr.tidyverse.org`
  `https://github.com/tidyverse/dplyr`
  `https://github.com/rstudio/cheatsheets/raw/master/source/pdfs/data-transformation-cheatsheet.pdf`
- **tibble**: Kirill Müller and Hadley Wickham (2017). tibble: Simple Data Frames. R package version 1.3.3.
  `https://github.com/tidyverse/tibble`
  `http://r4ds.had.co.nz/tibbles.html`
- **magrittr**: Stefan Milton Bache and Hadley Wickham (2014). magrittr: A Forward-Pipe Operator for R. R package version 1.5.
  `https://github.com/tidyverse/magrittr`
  `http://r4ds.had.co.nz/pipes.html`
- Silge J and Robinson D (2016). "tidytext: Text Mining and Analysis Using Tidy Data Principles in R." *JOSS*, *1*(3). doi: 10.21105/joss.00037.
  `http://tidytextmining.com/`
- Joe Cheng, Bhaskar Karambelkar and Yihui Xie (2017). leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library. R package version 1.1.0.9000. `https://rstudio.github.io/leaflet/`