Travaux dirigés pour l'introduction au logiciel R

Marco Pascucci

25/10/2018

préparation

Importation des données `iris`

```
data(iris)
sample_n(iris,10)

# iris est un data.frame
class(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
145	6.7	3.3	5.7	2.5	virginica
87	6.7	3.1	4.7	1.5	versicolor
49	5.3	3.7	1.5	0.2	setosa
36	5.0	3.2	1.2	0.2	setosa
149	6.2	3.4	5.4	2.3	virginica
2	4.9	3.0	1.4	0.2	setosa

```
## [1] "data.frame"
```

resumé des données avec `summarize()`

```
summarize(iris, Mean=mean(Sepal.Length),
          sd=sd(Sepal.Length))
```

Mean	sd
5.843333	0.8280661

`summarize()` génère un nouveau `data.frame`

Exercice

Extraire la moyenne de `Sepal.Length` seulement pour l'éspèce `Setosa`

Solution

```
data(iris)
t <- iris[iris$Species == "setosa",]
summarize(t, Mean=mean(Sepal.Length) )
```

```
##    Mean
## 1 5.006
```

. . . et si on voulait faire le même pour les autres éspèces?

Ce n'est pas compliqué mais un peu laborieur. . . et surtout:

*"c'est un problème tellement récourrent que **surement** quelqu'un à deja trouvé une solution"*

tidyverse

La librairie `tidyverse` contient des fonctions très pratiques pour manipuler les données.

Elle implémente aussi une evolution du data.frame:

Le `tibble`

```
iris_as_tibble <- as.tibble(iris)
```

Les fonctions de `tidyverse` comme `group()` et `nest()` retournent des tibbles.

grouper les données avec group()

```
iris_by_species <- group_by(iris, Species)
summarize(iris_by_species, Mean=mean(Sepal.Length),
          sd=sd(Sepal.Length))

# c'est un tibble!
# class(iris_by_species)
```

Species	Mean	sd
setosa	5.006	0.3524897
versicolor	5.936	0.5161711
virginica	6.588	0.6358796

`summarize` est appliqué séparement à chaque groupe. Le resultat de chaqu'un est mémorisé dans une ligne correspondante.

imbriquer avec `nest()`

`nest()` après avoir groupé, permet de séparer les données selon le groupe.

```
iris_nested <- nest(iris_by_species)
iris_nested
```

```
## # A tibble: 3 x 2
##   Species    data
##   <fct>      <list>
## 1 setosa     <tibble [50 x 4]>
## 2 versicolor <tibble [50 x 4]>
## 3 virginica  <tibble [50 x 4]>
```

les elements “data” sont des `tibbles`. On y accede avec la syntaxe usuelle:

```
iris_nested[[2,"data"]]
```

tibbles

Les tibbles sont des **structures des données**, comme les data.frames. Mieux, ils sont des data.frame

```
iris %>% as_tibble() %>% class()
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

DIFFERENCES

- les lignes (observations) n'ont pas de nom
- les "cellules" peuvent contenir des listes ou. . . des data.frames!

charger des données en format `tibble`

```
data(mtcars)
head(mtcars,3)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1

```
mtcars_t <- as_tibble(mtcars)
head(mtcars_t,3)
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
22.8	4	108	93	3.85	2.320	18.61	1	1	4	1

garder le nom des lignes

On peut garder le nom des lignes du data.frame en utulisant la fonction `rownames_to_column` du TD précedent, ou en utilsant le paramètre `rownames` de la fonction `as_tible()`:

```
mtcars_t <- as_tibble(mtcars, rownames="Model")
head(mtcars_t,3)
```

Model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1

créer des tibbles

à peu près comme un data.frame. On specifie toujours les valeurs par colomnes:

```
t <- tibble(
  x = 1:4,
  y = 1,
  z = x ^ 2 + y,
  jour = c("lundi", "mercredi","vendredi","dimanche")
)
t
```

x	y	z	jour
1	1	2	lundi
2	1	5	mercredi
3	1	10	vendredi
4	1	17	dimanche