data.table

INESSS

2025-09-03

data.table

```
library(data.table)
```

`dt[i, j, by]`

- ***dt*** : Nom du `data.table`
- ***i*** : Sous-ensemble de lignes, lignes à utiliser
- ***j*** : Colonnes à manipuler
- ***by*** : Regrouper par `by`

Créer un data.table

```
df1 <- data.frame(a = c(1, 2, 4), b = c("a", "b", "d"), c = c(111, 222, 333))
dt <- data.table(a = c(1, 2, 4), b = c("a", "b", "d"), c = c(111, 222, 333))
dt2 <- as.data.table(df1)
setDT(df1)  # Utile pour les grosses bases de données
            # Efficace au niveau de la mémoire. Ne fait pas de copie.
str(df1)
```

```
#> Classes 'data.table' and 'data.frame':    3 obs. of  3 variables:
#>  $ a: num  1 2 4
#>  $ b: chr  "a" "b" "d"
#>  $ c: num  111 222 333
#>  - attr(*, ".internal.selfref")=<externalptr>
```

Sélection de lignes avec `i`

En indiquant les numéros de lignes

```
dt[c(2, 3)]
```

```
#>         a      b     c
#>    <num> <char> <num>
#> 1:     2      b   222
#> 2:     4      d   333
```

Selon un opérateurs

```
dt[a > 2]
```

```
#>         a      b     c
#>    <num> <char> <num>
#> 1:     4      d   333
```

Extraire des colonnes

Par numéro

```
dt[, c(2)]
```

```
#>          b
#>    <char>
#> 1:      a
#> 2:      b
#> 3:      d
```

```
dt[, c(-2)]  # Un signe négatif exclu la colonne
```

```
#>         a     c
#>    <num> <num>
#> 1:     1   111
#> 2:     2   222
#> 3:     4   333
```

Extraire des colonnes

Par nom

```
dt[, .(a, c)]
```

```
#>         a     c
#>     <num> <num>
#> 1:      1   111
#> 2:      2   222
#> 3:      4   333
```

```
cols <- c("a", "c")
dt[, ..cols]
```

```
#>         a     c
#>     <num> <num>
#> 1:      1   111
#> 2:      2   222
#> 3:      4   333
```

Les fonctions *set* ou le symbole *:=*

Les fonctions avec le préfixe `set` ou les colonnes créées à partir de `:=` modifient les données sans faire de copie. En d'autres mots, pas besoin d'utiliser le `<-`.

Création de colonnes

```
dt
```

```
#>         a      b     c
#>     <num> <char> <num>
#> 1:      1      a   111
#> 2:      2      b   222
#> 3:      4      d   333
```

```
dt[, d := 1 + 2]
```

```
#>         a      b     c     d
#>     <num> <char> <num> <num>
#> 1:      1      a   111     3
#> 2:      2      b   222     3
#> 3:      4      d   333     3
```

Création de colonnes

Pour un sous-ensemble de lignes

```
dt[a == 1, d := 1 + 2]
```

```
#> Index: <a>
#>        a      b     c     d
#>    <num> <char> <num> <num>
#> 1:     1      a   111     3
#> 2:     2      b   222    NA
#> 3:     4      d   333    NA
```

Les lignes non sélectionnées auront des `NA`

Création de colonnes

Pour créer plusieurs colonnes en même temps

```
dt[, `:=` (d = 2, e = 2+3)]
```

```
#> Index: <a>
#>        a      b     c     d     e
#>    <num> <char> <num> <num> <num>
#> 1:     1      a   111     2     5
#> 2:     2      b   222     2     5
#> 3:     4      d   333     2     5
```

Supprimer une colonne

```
dt
```

```
#> Index: <a>
#>        a      b     c     d     e
#>    <num> <char> <num> <num> <num>
#> 1:     1      a   111     2     5
#> 2:     2      b   222     2     5
#> 3:     4      d   333     2     5
```

```
dt[, d := NULL]
```

```
#> Index: <a>
#>        a      b     c     e
#>    <num> <char> <num> <num>
#> 1:     1      a   111     5
#> 2:     2      b   222     5
#> 3:     4      d   333     5
```

Regrouper selon `by`

Calculer la colonne `j` en regroupant par la colonne `a`

```
dt[, j, by = .(a)]
```

Calculer la colonne `j` en regroupant par la colonne `a` et trier en ordre croissant la colonne `a`

```
dt[, j, keyby = .(a)]
```

Regrouper selon by

```
dt2
```

```
#>         a     b
#>     <num> <num>
#> 1:      1     4
#> 2:      1     5
#> 3:      2     6
```

Résumer les lignes au sein des groupes

```
dt2[, .(c = sum(b)), by = a]
```

```
#>         a     c
#>     <num> <num>
#> 1:      1     9
#> 2:      2     6
```

Regrouper selon by

Créer une nouvelle colonne et calculer les lignes au sein des groupes.

```
dt2[, c := sum(b), by = a]
```

```
#>         a     b     c
#>     <num> <num> <num>
#> 1:      1     4     9
#> 2:      1     5     9
#> 3:      2     6     6
```

Regrouper selon by

Extraire les premières observations d'un groupe

```
dt2[, .SD[1], by = a]  # .SD = Subset data
```

```
#>         a     b     c
#>     <num> <num> <num>
#> 1:      1     4     9
#> 2:      2     6     6
```

Extraire les dernières observations d'un groupe

```
dt2[, .SD[.N], by = a]  # .N est la dernière valeur
```

```
#>         a     b     c
#>     <num> <num> <num>
#> 1:      1     5     9
#> 2:      2     6     6
```

Enchainement d'opérations (*chaining*)

Il est possible d'enchainer les opérations avec les *pipes* `[]`

> `dt[...][...][...]`

```
dt[, e := NULL][, d := 1+3][, e := sum(c), by = a]
# 1) Supprimer colonne e
# 2) Créer colonne d = 4
# 3) Créer colonne e = somme de c groupé par colonne a
```

Exemple de fonctions `set`

```r
# Trier les données par a croissant et b décroissant
setorder(dt, a, -b)

# Order des colonnes
setcolorder(dt, c("b", "c", "a", "e", "d"))

# Renommer les colonnes a et b par col1 et col2
setnames(dt, old = c("a", "b"), new = c("col1", "col2"))
```

Jointure *left join*

```
dt_a
```

```
#>         a      b
#>     <int> <char>
#> 1:      1      c
#> 2:      2      a
#> 3:      3      b
```

```
dt_b
```

```
#>         x      y
#>     <int> <char>
#> 1:      3      b
#> 2:      2      c
#> 3:      1      a
```

```
dt_a[dt_b, on = .(b = y)]
```

```
#>         a      b     x
#>     <int> <char> <int>
#> 1:      3      b     3
#> 2:      1      c     2
#> 3:      2      a     1
```

Jointure *left join*

```
dt_a
```

```
#>         a      b     c
#>    <int> <char> <num>
#> 1:     1      c     7
#> 2:     2      a     5
#> 3:     3      b     6
```

```
dt_b
```

```
#>         x      y     z
#>    <int> <char> <num>
#> 1:     3      b     4
#> 2:     2      c     5
#> 3:     1      a     8
```

```
dt_a[dt_b, on = .(b = y, c > z)]
```

```
#>         a      b     c     x
#>    <int> <char> <num> <int>
#> 1:     3      b     4     3
#> 2:     1      c     5     2
#> 3:    NA      a     8     1
```

Rolling join

Jointure qui connserve la correspondance **la plus récente** avec la table de données de gauche. «roll = -Inf» inverse le sens.

```
  dt_a
```

```
#>         a     id       date
#>     <num> <char>     <Date>
#> 1:      1      A 2010-01-01
#> 2:      2      A 2012-01-01
#> 3:      3      A 2014-01-01
#> 4:      1      B 2010-01-01
#> 5:      2      B 2012-01-01
```

```
  dt_b
```

```
#>         b     id       date
#>     <num> <char>     <Date>
#> 1:      1      A 2013-01-01
#> 2:      1      B 2013-01-01
```

```
dt_a[dt_b, on = .(id = id, date = date), roll=TRUE]
```

```
#>         a     id       date     b
#>     <num> <char>     <Date> <num>
#> 1:      2      A 2013-01-01     1
#> 2:      2      B 2013-01-01     1
```

Dans les deux cas, les dates qui précèdent 2013-01-01 de `dt_b` sont 2012-01-01 dans `dt_a`.

Lier - Lignes

```
dt_a
```

```
#>        id     x
#>    <num> <int>
#> 1:     1     1
#> 2:     1     2
```

```
dt_b
```

```
#>        id     x
#>    <num> <num>
#> 1:     2     5
```

```
rbind(dt_a, dt_b)
```

```
#>        id     x
#>    <num> <num>
#> 1:     1     1
#> 2:     1     2
#> 3:     2     5
```

Lier - Colonnes

```
dt_a
```

```
#>        id     x
#>    <num> <int>
#> 1:     1     1
#> 2:     1     2
```

```
dt_b
```

```
#>        y
#>    <int>
#> 1:     5
#> 2:     6
```

```
cbind(dt_a, dt_b)
```

```
#>        id     x     y
#>    <num> <int> <int>
#> 1:     1     1     5
#> 2:     1     2     6
```

Transformer au format large

```
dt
```

```
#>        id variable valeur
#>     <num>   <char>  <num>
#> 1:     1        A     10
#> 2:     1        B     20
#> 3:     2        A     15
#> 4:     2        B     25
```

```
dcast(dt, id ~ variable, value.var = "valeur")
```

```
#> Key: <id>
#>        id     A     B
#>     <num> <num> <num>
#> 1:     1    10    20
#> 2:     2    15    25
```

Transformer au format long

```
dt
```

```
#>       id  Var1  Var2
#>    <int> <num> <num>
#> 1:     1    10   100
#> 2:     2    20   200
#> 3:     3    30   300
```

```
melt(dt, id.vars = "id", measure.vars = c("Var1", "Var2"))
```

```
#>       id variable value
#>    <int>   <fctr> <num>
#> 1:     1     Var1    10
#> 2:     2     Var1    20
#> 3:     3     Var1    30
#> 4:     1     Var2   100
#> 5:     2     Var2   200
#> 6:     3     Var2   300
```

Appliquer une fonction sur plusieurs colonnes

```
#>         a     b
#>     <int> <int>
#> 1:      1     4
#> 2:      2     5
#> 3:      3     6
```

```
dt[, lapply(.SD, mean), .SDcols = c("a", "b")]
```

```
#>         a     b
#>     <num> <num>
#> 1:      2     5
```

```
cols <- c("a", "b")
dt[, paste0(cols, "_m") := lapply(.SD, mean), .SDcols = cols]
```

```
#>         a     b   a_m   b_m
#>     <int> <int> <num> <num>
#> 1:      1     4     2     5
#> 2:      2     5     2     5
#> 3:      3     6     2     5
```

Séquence - Index

Indiquer de 1 à N les lignes d'un même groupe

```
dt[, c := 1:.N, by = .(b)]
```

```
#>         a      b     c
#>    <int> <char> <int>
#> 1:     1      A     1
#> 2:     2      A     2
#> 3:     3      B     1
```

Valeur précédente ou suivante

```
# Valeur précédente de A par B
dt[, C := shift(A, 1), by = B]
```

```
#>         A      B     C
#>     <int> <char> <int>
#> 1:      1      X    NA
#> 2:      2      X     1
#> 3:      3      X     2
#> 4:      4      Y    NA
#> 5:      5      Y     4
```

```
# Valeur suivante de A par B
dt[, C := shift(A, 1, type =  "lead"), by = B]
```

```
#>         A      B     C
#>     <int> <char> <int>
#> 1:      1      X     2
#> 2:      2      X     3
#> 3:      3      X    NA
#> 4:      4      Y     5
#> 5:      5      Y    NA
```

Import / Export

Importer un fichier .csv

```
fread()
```

Exporter un fichier .csv

```
fwrite()
```

Bien que `fread()` et `fwrite()` soient très rapides, je conseille de rester avec `saveRDS()`/`readRDS()` et `save()`/`load()` qui prennent beaucoup moins d'espace (fichier compressé) et permette une meilleure intégrité des données. Par exemple, il pourrait arriver que les dates ne s'affichent pas correctement.

Symboles spéciaux

```
#>          x     v     y     a     b
#>     <char> <num> <num> <int> <int>
#> 1:      b     1     1     1     9
#> 2:      b     1     3     2     8
#> 3:      b     1     6     3     7
#> 4:      a     2     1     4     6
#> 5:      a     2     3     5     5
#> 6:      a     1     6     6     4
#> 7:      c     1     1     7     3
#> 8:      c     2     3     8     2
#> 9:      c     2     6     9     1
```

```
# Dernière ligne
dt[.N]
```

```
#>          x     v     y     a     b
#>     <char> <num> <num> <int> <int>
#> 1:      c     2     6     9     1
```

```
# Nombre de lignes
dt[, .N]
```

```
#> [1] 9
```

```
# Nombre de lignes dans chaque group
dt[, .N, by=x]
```

```
#>          x     N
#>     <char> <int>
#> 1:      b     3
#> 2:      a     3
#> 3:      c     3
```

Symboles spéciaux

```
#>          x     v     y     a     b
#>     <char> <num> <num> <int> <int>
#> 1:      b     1     1     1     9
#> 2:      b     1     3     2     8
#> 3:      b     1     6     3     7
#> 4:      a     2     1     4     6
#> 5:      a     2     3     5     5
#> 6:      a     1     6     6     4
#> 7:      c     1     1     7     3
#> 8:      c     2     3     8     2
#> 9:      c     2     6     9     1
```

```
# Sélectionne les colonnes x à y
dt[, .SD, .SDcols=x:y]
```

```
#>          x     v     y
#>     <char> <num> <num>
#> 1:      b     1     1
#> 2:      b     1     3
#> 3:      b     1     6
#> 4:      a     2     1
#> 5:      a     2     3
#> 6:      a     1     6
#> 7:      c     1     1
#> 8:      c     2     3
#> 9:      c     2     6
```

```
# Première ligne de chaque colonne
dt[, .SD[1]]
```

```
#>          x     v     y     a     b
#>     <char> <num> <num> <int> <int>
#> 1:      b     1     1     1     9
```

```
# Première ligne de chaque colonne
# de chaque groupe dans 'x'
dt[, .SD[1], by=x]
```

```
#>          x     v     y     a     b
#>     <char> <num> <num> <int> <int>
#> 1:      b     1     1     1     9
#> 2:      a     2     1     4     6
#> 3:      c     1     1     7     3
```

Symboles spéciaux

```
#>          x     v     y     a     b
#>     <char> <num> <num> <int> <int>
#> 1:      b     1     1     1     9
#> 2:      b     1     3     2     8
#> 3:      b     1     6     3     7
#> 4:      a     2     1     4     6
#> 5:      a     2     3     5     5
#> 6:      a     1     6     6     4
#> 7:      c     1     1     7     3
#> 8:      c     2     3     8     2
#> 9:      c     2     6     9     1
```

```
# Numéro de la première ligne par
# groupe de x
dt[, .I[1], by=x]
```

```
#>          x    V1
#>     <char> <int>
#> 1:      b     1
#> 2:      a     4
#> 3:      c     7
```

```
# Numéro de la dernière ligne par
# groupe de x
dt[, .I[.N], by=x]
```

```
#>          x    V1
#>     <char> <int>
#> 1:      b     3
#> 2:      a     6
#> 3:      c     9
```

Symboles spéciaux

```
#>          x      y      b
#>     <char> <num> <int>
#> 1:       b     1     9
#> 2:       b     3     8
#> 3:       b     6     7
#> 4:       a     1     6
#> 5:       a     3     5
#> 6:       a     6     4
#> 7:       c     1     3
#> 8:       c     3     2
#> 9:       c     6     1
```

```
# Itérateur de groupe
dt[, grp := .GRP, keyby=x]
```

```
#> Key: <x>
#>          x      y      b   grp
#>     <char> <num> <int> <int>
#> 1:       a     1     6     1
#> 2:       a     3     5     1
#> 3:       a     6     4     1
#> 4:       b     1     9     2
#> 5:       b     3     8     2
#> 6:       b     6     7     2
#> 7:       c     1     3     3
#> 8:       c     3     2     3
#> 9:       c     6     1     3
```

```
# Itérateur de groupe avec un compte
# de progrès
dt[
  , grp_pct := round(.GRP/.NGRP, 2),
  by=x
]
```

```
#> Key: <x>
#>          x      y      b   grp grp_pct
#>     <char> <num> <int> <int>   <num>
#> 1:       a     1     6     1    0.33
#> 2:       a     3     5     1    0.33
#> 3:       a     6     4     1    0.33
#> 4:       b     1     9     2    0.67
#> 5:       b     3     8     2    0.67
#> 6:       b     6     7     2    0.67
#> 7:       c     1     3     3    1.00
#> 8:       c     3     2     3    1.00
#> 9:       c     6     1     3    1.00
```

Exemple avancé avec dt[, .I[ ]]

Prescription à un patient

```
#>        ID     DateRx nJours
#>     <num>     <Date>  <num>
#> 1:      1 2020-01-01     30
#> 2:      1 2020-01-20      5
#> 3:      1 2020-02-01     30
#> 4:      2 2020-06-01     30
```

```
dt[, DateFin := DateRx + nJours - 1]
dt[, diff := as.integer(DateRx - shift(DateFin)), .(ID)]
```

```
#>        ID     DateRx nJours    DateFin  diff
#>     <num>     <Date>  <num>     <Date> <int>
#> 1:      1 2020-01-01     30 2020-01-30    NA
#> 2:      1 2020-01-20      5 2020-01-24   -10
#> 3:      1 2020-02-01     30 2020-03-01     8
#> 4:      2 2020-06-01     30 2020-06-30    NA
```

```
#>        ID     DateRx nJours    DateFin  diff
#>     <num>     <Date>  <num>     <Date> <int>
#> 1:     1 2020-01-01     30 2020-01-30    NA
#> 2:     1 2020-01-20      5 2020-01-24   -10
#> 3:     1 2020-02-01     30 2020-03-01     8
#> 4:     2 2020-06-01     30 2020-06-30    NA
```

```
dt[, .I[diff < 1], .(ID)]
```

```
#>        ID    V1
#>     <num> <int>
#> 1:     1    NA
#> 2:     1     2
#> 3:     2    NA
```

```
idx <- c(
  dt[, .I[diff < 1], .(ID)]$V1-1,
  dt[, .I[diff < 1], .(ID)]$V1
)
idx <- sort(idx[!is.na(idx)])
idx
```

```
#> [1] 1 2
```

```
dt[idx, ajout := shift(nJours, -1), .(ID)]
```

```
#>       ID     DateRx nJours    DateFin  diff ajout
#>    <num>     <Date>  <num>     <Date> <int> <num>
#> 1:     1 2020-01-01     30 2020-01-30    NA     5
#> 2:     1 2020-01-20      5 2020-01-24   -10    NA
#> 3:     1 2020-02-01     30 2020-03-01     8    NA
#> 4:     2 2020-06-01     30 2020-06-30    NA    NA
```

```
dt[!is.na(ajout), nJours := nJours + ajout]
```

```
#>       ID     DateRx nJours    DateFin  diff ajout
#>    <num>     <Date>  <num>     <Date> <int> <num>
#> 1:     1 2020-01-01     35 2020-01-30    NA     5
#> 2:     1 2020-01-20      5 2020-01-24   -10    NA
#> 3:     1 2020-02-01     30 2020-03-01     8    NA
#> 4:     2 2020-06-01     30 2020-06-30    NA    NA
```

```
#>       ID     DateRx nJours    DateFin  diff ajout
#>    <num>     <Date>  <num>     <Date> <int> <num>
#> 1:     1 2020-01-01     35 2020-01-30    NA     5
#> 2:     1 2020-01-20      5 2020-01-24   -10    NA
#> 3:     1 2020-02-01     30 2020-03-01     8    NA
#> 4:     2 2020-06-01     30 2020-06-30    NA    NA
```

```
dt <- dt[!dt[, .I[diff < 1], .(ID)]$V1]
```

```
#>       ID     DateRx nJours    DateFin  diff ajout
#>    <num>     <Date>  <num>     <Date> <int> <num>
#> 1:     1 2020-01-01     35 2020-01-30    NA     5
#> 2:     1 2020-02-01     30 2020-03-01     8    NA
#> 3:     2 2020-06-01     30 2020-06-30    NA    NA
```

Attention si plusieurs périodes doivent être fusionnées

```
#>       ID     DateRx nJours    DateFin  diff
#>    <num>     <Date>  <num>     <Date> <int>
#> 1:     1 2020-01-01     30 2020-01-30     0
#> 2:     1 2020-01-20      5 2020-01-24   -10
#> 3:     1 2020-01-22     30 2020-02-20    -2
#> 4:     2 2020-06-01     30 2020-06-30     0
```

On doit s'assurer de ne pas avoir trois nombres qui se suivent pour un même ID et répéter les étapes dans un `while loop`.
À faire en devoir! :P