Transformer les données avec data.table : : COMPENDIUM

Les bases

data.table est un package très rapide et performant en gestion de la mémoire pour transformer des données avec R avec une syntaxe concise. Il convertit les objets data.frame natifs de R en data.table avec des fonctionnalités nouvelles et étendues. Les bases pour utiliser data.table sont:

dt[i, j, by]

Objet data.table **dt**,
Extraction des lignes avec **i**
et manipulation des colonnes avec **j**,
avec un regroupement selon **by**.

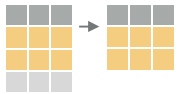
Les data.tables sont aussi des data.frames – les fonctions qui opèrent sur des data.frames sont utilisables sur les data.tables.

Créer un data.table

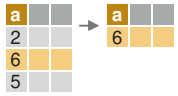
data.table(a = c(1, 2), b = c("a", "b")**)** – crée un data.table en partant de rien. Similaire à data.frame().

setDT(df**)*** ou **as.data.table(**df**)** – convertit un data.frame ou une liste en data.table.

Extraire des lignes avec i



dt[**1:2**] – extraire les lignes en fonction des numéros de lignes.



dt[**a > 5**] – extraire les lignes en fonction des valeurs contenues dans une ou plusieurs colonnes.

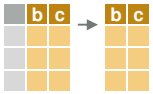
OPERATEURS LOGIQUES A UTILISER DANS i

<	<=	is.na()	%in%	|	**%like%**
>	>=	!is.na()	!	&	**%between%**

Manipuler les colonnes avec j

EXTRAIRE

dt[, **c(2)**] – extraire des colonnes par numéro. Préfixer avec "-" les numéros des colonnes à ignorer.



dt[, **.(b, c)**] – extraire les colonnes par leur nom.

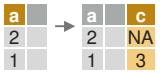
SOMMER

dt[, **.(x = sum(a))**] – créer un data.table avec de nouvelles colonnes basées sur le total des valeurs des lignes.

Les fonctions de sommation telles que mean(), median(), min(), max(), etc. peuvent être utilisées.

CALCULER DES COLONNES*

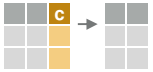
dt[, **c := 1 + 2**] – calculer une colonne sur la base d'une expression.



dt[**a == 1**, **c := 1 + 2**] – calculer une colonne sur la base d'une expression, mais seulement sur un sous-ensemble de lignes.

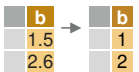
dt[, **`:=`(c = 1 , d = 2)**] – calculer plusieurs colonnes sur la base d'expressions distinctes.

SUPPRIMER UNE COLONNE



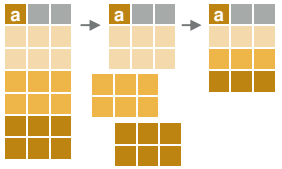
dt[, **c := NULL**] – supprimer la colonne **c**.

CONVERTIR LE TYPE D'UNE COLONNE



dt[, **b := as.integer(b)**] – convertir le type d'une colonne en utilisant as.integer(), as.numeric(), as.character(), as.Date(), etc..

Grouper avec by



dt[, j, **by = .(a)**] – grouper les lignes par valeurs des colonnes indiquées.

dt[, j, **keyby = .(a)**] – grouper *et trier simultanément* les lignes par valeur des colonnes indiquées.

OPÉRATIONS COMMUNES DE GROUPEMENT

dt[, **.(c = sum(b))**, **by = a**] – sommer les lignes par groupe.

dt[, **c := sum(b)**, **by = a**] – créer une nouvelle colonne et calculer les lignes dans chaque groupe.

dt[, **.SD[1]**, **by = a**] – extraire la première ligne de chaque groupe.

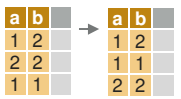
dt[, **.SD[.N]**, **by = a**] – extraire la dernière ligne de chaque groupe.

Chaînage

dt[…**][**…**]** – réaliser une séquence d'opérations sur data.table en *chaînant* plusieurs "[ ]".

Fonctions pour les data.tables

TRIER



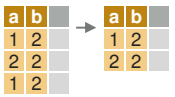
setorder(dt, a, **-**b**)** – trier un data.table en fonction des colonnes indiquées. Préfixer les noms des colonnes avec "**-**" pour trier dans l'ordre descendant.

* FONCTIONS SET ET :=

Les fonctions de data.table préfixées par "set" et l'opérateur ":=" fonctionnent sans affectation avec "<-" pour modifier les données sans faire de copies en mémoire. Par exemple la fonction "setDT(df)" est plus efficace que l'instruction analogue "df <- as.data.table(df)".

Créé par Erik Petrovsky et Mara Destefanis – maragdestefanis@gmail.com • Traduit par Christian Wiat – w9204-rs@yahoo.com • Voir la page d'accueil ou la vignette de data.table • data.table version 1.17.8 • Mise-à-jour: 2025-07 • traduction 2025-09

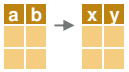
LIGNES UNIQUES



unique(dt, by = c("a", "b")**)** – extraire des lignes uniques basées sur les colonnes spécifiées dans “by”. Ne pas utiliser “by” pour avoir toutes les colonnes.

uniqueN(dt, by = c("a", "b")**)** – compter le nombre de lignes uniques basées sur les colonnes spécifiées dans “by”.

RENOMMER DES COLONNES



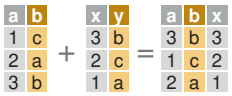
setnames(dt, c("a", "b"), c("x", "y")**)** – renommer les anciennes colonnes (a, b) en (x, y).

DÉFINIR DES CLÉS

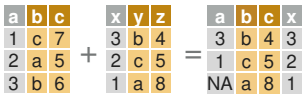
setkey(dt, a, b**)** – définir des clés pour permettre des recherches rapides et répétées dans les colonnes spécifiées en utilisant “dt[.(value),]” ou pour fusionner sans indiquer les colonnes à utiliser avec “dt_a[dt_b]”.

Combiner des data.tables

JOINTURE

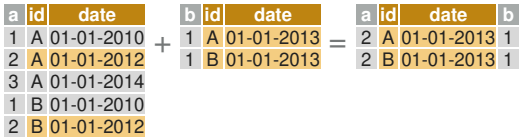


dt_a[dt_b, **on = .(b = y)**] – combiner les data.tables sur la base des lignes d’égales valeurs.



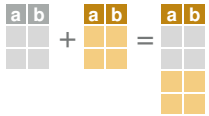
dt_a[dt_b, **on = .(b = y, c > z)**] – combiner les data.tables sur la base des lignes de valeurs égales *et différentes*.

JOINTURE GLISSANTE

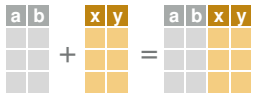


dt_a[dt_b, **on = .(id = id, date = date), roll = TRUE**] – combiner les data.tables pour les lignes qui correspondent dans les colonnes id, mais ne garder que la correspondance précédente la plus récente avec la data.table de gauche en fonction des colonnes de date. Utiliser “roll = -Inf” pour inverser la direction.

LIER



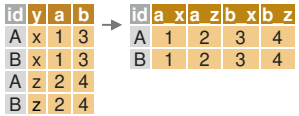
rbind(dt_a, dt_b**)** – combiner les lignes de deux data.tables.



cbind(dt_a, dt_b**)** – combiner les colonnes de deux data.tables.

Restructurer un data.table

RESTRUCTURER EN LARGEUR

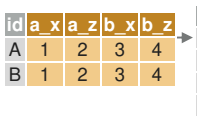


dcast(dt,
id ~ y,
value.var = c("a", "b") **)**

Restructurer une data.table d’un format long en format large.

dt	Un data.table.
id ~ y	Formule avec pour le membre de gauche : colonnes ID contenant les IDs d’entrées multiples. Et pour membre de droite : les colonnes avec les valeurs à distribuer dans l’entête des colonnes.
value.var	Colonnes des valeurs à mettre dans les cellules.

RESTRUCTURER EN LONGUEUR



melt(dt,
measure.vars = measure (
value.name, y, sep="_")
)

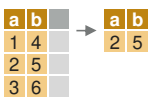
Restructurer un data.table d’un format large en format long.

dt	Un data.table.
measure.vars	Colonnes des valeurs à mettre dans les cellules, souvent en utilisant measure() ou patterns().
id.vars	Vecteur de caractères des noms des colonnes ID (optionnel).
variable.name, value.name	Noms des colonnes de sortie (optionnel).

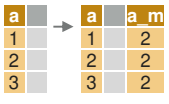
measure(out_name1, out_name2, sep="_", pattern="([ab])_(.*)"**)**
sep(séparateur) ou pattern (expression régulière) utilisés pour spécifier les colonnes à restructurer en analysant les noms des colonnes d’entrée.
out_name1, out_name2: noms des colonnes de sortie (crée une colonne à valeur unique), ou value.name (crée des colonnes de valeurs pour chaque partie unique du nom de la colonne restructurée).

Fonction appliquée aux colonnes

APPLIQUER UNE FONCTION À PLUSIEURS COLONNES



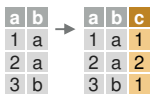
dt[, **lapply(.SD, mean)**, **.SDcols = c("a", "b")**] – appliquer une fonction – telle que mean(), as.character(), which.max() – aux colonnes indiquées dans .SDcols avec lapply() et le symbole .SD . Fonctionne aussi avec les groupes.



cols <- c("a")
dt[, **paste0(cols, "_m") := lapply(.SD, mean)**, **.SDcols = cols**] – appliquer une fonction aux colonnes indiquées et assigner le résultat avec les noms des variables suffixés aux données originales.

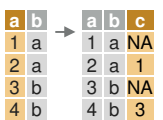
Lignes séquentielles

IDS DE LIGNES



dt[, **c := 1:.N**, **by = b**] – évaluer, au sein des groupes, une colonne avec des IDs de lignes séquentielles.

APRÈS & AVANT



dt[, **c := shift(a, 1)**, **by = b**] – dupliquer, au sein des groupes, une colonne avec les lignes *suivantes* de la valeur spécifiée.

dt[, **c := shift(a, 1, type = "lead")**, **by = b**] – dupliquer, au sein des groupes, une colonne avec les lignes *précédentes* de la valeur spécifiée.

Lire & écrire des fichiers

IMPORTER

fread("file.csv"**)** – lire les données d’un fichier de type .csv ou .tsv, dans R.

fread("file.csv", select = c("a", "b")**)** – lire des colonnes spécifiques d’un fichier dans R.

EXPORTER

fwrite(dt, "file.csv"**)** – écrire les données dans un fichier depuis R.