Data Transformation with data.table : : CHEAT SHEET

Basics

data.table is an extremely fast and memory efficient package for transforming data in R. It works by converting R's native data frame objects into data.tables with new and enhanced functionality. The basics of working with data.tables are:

dt[i, j, by]

Take data.table **dt**,
subset rows using **i**
and manipulate columns with **j**,
grouped according to **by**.

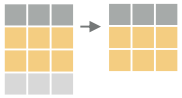
data.tables are also data frames – functions that work with data frames therefore also work with data.tables.

Create a data.table

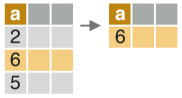
data.table(a = c(1, 2), b = c("a", "b")**)** – create a data.table from scratch. Analogous to data.frame().

setDT(df**)*** or **as.data.table(**df**)** – convert a data frame or a list to a data.table.

Subset rows using i



dt[**1:2**,] – subset rows based on row numbers.



dt[**a > 5**,] – subset rows based on values in one or more columns.

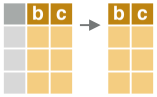
LOGICAL OPERATORS TO USE IN i

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

Manipulate columns with j

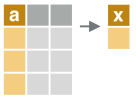
EXTRACT

dt[, **c(2)**] – extract columns by number. Prefix column numbers with "-" to drop.



dt[, **.(b, c)**] – extract columns by name.

SUMMARIZE

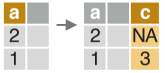


dt[, **.(x = sum(a))**] – create a data.table with new columns based on the summarized values of rows.

Summary functions like mean(), median(), min(), max(), etc. can be used to summarize rows.

COMPUTE COLUMNS*

dt[, **c := 1 + 2**] – compute a column based on an expression.



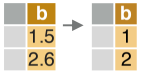
dt[**a == 1, c := 1 + 2**] – compute a column based on an expression but only for a subset of rows.

dt[, **`:=`(c = 1 , d = 2)**] – compute multiple columns based on separate expressions.

DELETE COLUMN

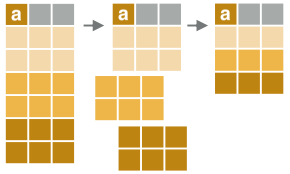
dt[, **c := NULL**] – delete a column.

CONVERT COLUMN TYPE



dt[, **b := as.integer(b)**] – convert the type of a column using as.integer(), as.numeric(), as.character(), as.Date(), etc..

Group according to by



dt[, j, **by = .(a)**] – group rows by values in specified columns.

dt[, j, **keyby = .(a)**] – group *and simultaneously sort* rows by values in specified columns.

COMMON GROUPED OPERATIONS

dt[, **.(c = sum(b))**, **by = a**] – summarize rows within groups.

dt[, **c := sum(b)**, **by = a**] – create a new column and compute rows within groups.

dt[, **.SD[1]**, **by = a**] – extract first row of groups.

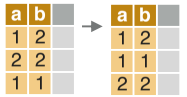
dt[, **.SD[.N]**, **by = a**] – extract last row of groups.

Chaining

dt[…**][**…**]** – perform a sequence of data.table operations by *chaining* multiple "[]".

Functions for data.tables

REORDER

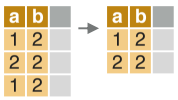


setorder(dt, a, **-**b**)** – reorder a data.table according to specified columns. Prefix column names with "**-**" for descending order.

*** SET FUNCTIONS AND :=**

data.table's functions prefixed with "set" and the operator ":=" work without "<-" to alter data without making copies in memory. E.g., the more efficient "setDT(df)" is analogous to "df <- as.data.table(df)".

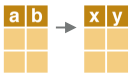
UNIQUE ROWS



unique(dt, by = c("a", "b")**)** – extract unique rows based on columns specified in "by". Leave out "by" to use all columns.

uniqueN(dt, by = c("a", "b")**)** – count the number of unique rows based on columns specified in "by".

RENAME COLUMNS



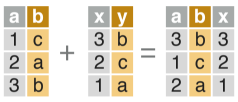
setnames(dt, c("a", "b"), c("x", "y")**)** – rename columns.

SET KEYS

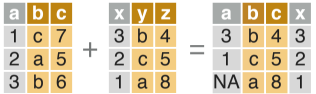
setkey(dt, a, b**)** – set keys to enable fast repeated lookup in specified columns using "dt[.(value),]" or for merging without specifying merging columns using "dt_a[dt_b]".

Combine data.tables

JOIN

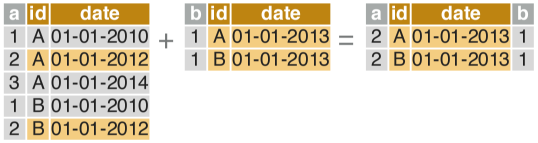


dt_a[dt_b, **on = .(b = y)**] – join data.tables on rows with equal values.



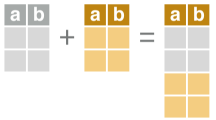
dt_a[dt_b, **on = .(b = y, c > z)**] – join data.tables on rows with equal *and unequal* values.

ROLLING JOIN

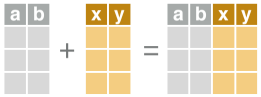


dt_a[dt_b, **on = .(id = id, date = date), roll = TRUE**] – join data.tables on matching rows in id columns but only keep the most recent preceding match with the left data.table according to date columns. "roll = -Inf" reverses direction.

BIND



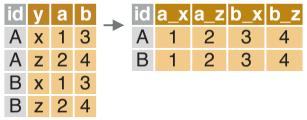
rbind(dt_a, dt_b**)** – combine rows of two data.tables.



cbind(dt_a, dt_b**)** – combine columns of two data.tables.

Reshape a data.table

RESHAPE TO WIDE FORMAT

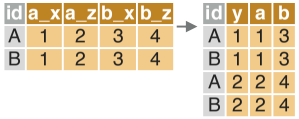


dcast(dt,
id ~ y,
value.var = c("a", "b")**)**

Reshape a data.table from long to wide format.

dt	A data.table.
id ~ y	Formula with a LHS: ID columns containing IDs for multiple entries. And a RHS: columns with values to spread in column headers.
value.var	Columns containing values to fill into cells.

RESHAPE TO LONG FORMAT



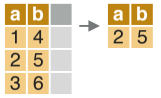
melt(dt,
id.vars = c("id"),
measure.vars = patterns("^a", "^b"),
variable.name = "y",
value.name = c("a", "b")**)**

Reshape a data.table from wide to long format.

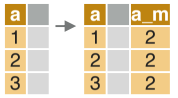
dt	A data.table.
id.vars	ID columns with IDs for multiple entries.
measure.vars	Columns containing values to fill into cells (often in pattern form).
variable.name, value.name	Names of new columns for variables and values derived from old headers.

Apply function to cols.

APPLY A FUNCTION TO MULTIPLE COLUMNS



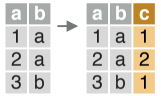
dt[, **lapply(.SD, mean)**, **.SDcols = c("a", "b")**] – apply a function – e.g. mean(), as.character(), which.max() – to columns specified in .SDcols with lapply() and the .SD symbol. Also works with groups.



cols <- c("a")
dt[, **paste0(cols, "_m") := lapply(.SD, mean)**, **.SDcols = cols**] – apply a function to specified columns and assign the result with suffixed variable names to the original data.

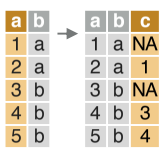
Sequential rows

ROW IDS



dt[, **c := 1:.N**, **by = b**] – within groups, compute a column with sequential row IDs.

LAG & LEAD



dt[, **c := shift(a, 1)**, **by = b**] – within groups, duplicate a column with rows *lagged* by specified amount.

dt[, **c := shift(a, 1, type = "lead")**, **by = b**] – within groups, duplicate a column with rows *leading* by specified amount.

read & write files

IMPORT

fread("file.csv"**)** – read data from a flat file such as .csv or .tsv into R.

fread("file.csv", select = c("a", "b")**)** – read specified columns from a flat file into R.

EXPORT

fwrite(dt, "file.csv"**)** – write data to a flat file from R.