Data Transformation with data.table : : CHEAT SHEET

Basics

data.table is an extremely fast and memory efficient package for transforming data in R with a concise syntax. 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**.

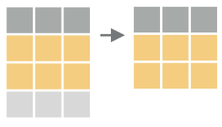
data.tables are also data frames – functions that work with data frames also work with data.tables.

Create a data.table

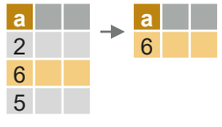
data.table(a = c(1, 2), b = c("a", "b")**)**
create a data.table from scratch. Analogous to data.frame().

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.



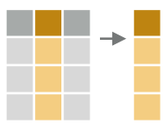
dt[**a > 5**]
subset rows based on values in one or more columns.

LOGICAL OPERATORS TO USE IN i

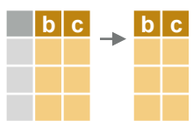
<	<=	is.na()	%in%	|	**%like%**
>	>=	!is.na()	!	&	**%between%**

Manipulate columns with j

EXTRACT



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



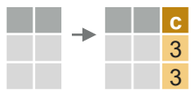
dt[, **.(b, c)**]
extract columns by name.

SUMMARIZE

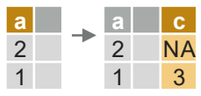
dt[, **.(x = sum(a))**]
create a data.table with new columns based on the summarized values of rows.

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

COMPUTE COLUMNS*



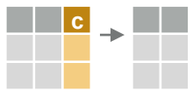
dt[, **c := 1 + 2**]
compute a column based on an expression.



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

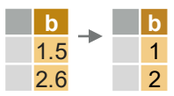
dt[, **`:=`(c = 1 , d = 2)**]
compute multiple columns based on separate expressions.

DELETE COLUMN



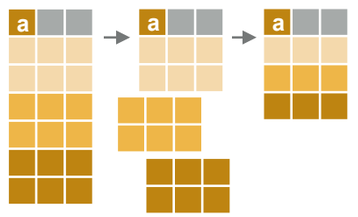
dt[, **c := NULL**]
delete a column.

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.

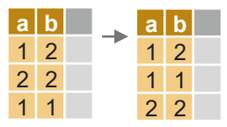
dt[, **.SD[.N]**, **by = a**] – extract last row of groups.

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

a	b	
1	2	
2	2	
1	2	

→

a	b	
1	2	
2	2	

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

a	b

→

x	y

setnames(`dt, c("a", "b"), c("x", "y")`**)**
rename columns from old names (a, b) to new names (x, y).

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

a	b
1	c
2	a
3	b

+

x	y
3	b
2	c
1	a

=

a	b	x
3	b	3
1	c	2
2	a	1

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

a	b	c
1	c	7
2	a	5
3	b	6

+

x	y	z
3	b	4
2	c	5
1	a	8

=

a	b	c	x
3	b	4	3
1	c	5	2
NA	a	8	1

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

ROLLING JOIN

a	id	date
1	A	01-01-2010
2	A	01-01-2012
3	A	01-01-2014
1	B	01-01-2010
2	B	01-01-2012

+

b	id	date
1	A	01-01-2013
1	B	01-01-2013

=

a	id	date	b
2	A	01-01-2013	1
2	B	01-01-2013	1

`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

a	b

+

a	b

=

a	b

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

a	b

+

x	y

=

a	b	x	y

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

Reshape a data.table

RESHAPE TO WIDE FORMAT

id	y	a	b
A	x	1	3
A	x	1	3
B	z	2	4
B	z	2	4

→

id	a_x	a_z	b_x	b_z
A	1	2	3	4
B	1	2	3	4

```
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

id	a_x	a_z	b_x	b_z
A	1	2	3	4
B	1	2	3	4

→

id	y	a	b
A	x	1	3
B	x	1	3
A	z	2	4
B	z	2	4

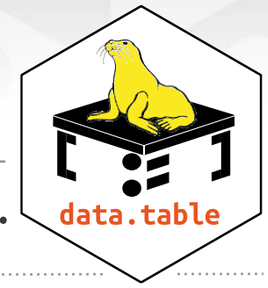
```
melt(
  dt,
  measure.vars = measure(
    value.name, y, sep="_"
  )
)
```

Reshape a data.table from wide to long format.

dt	A data.table.
measure.vars	Columns containing values to fill into cells, often using measure() or patterns ().
id.vars	Character vector of ID column names (optional).
variable.name, value.name	Names for output columns (optional).

measure(out_name1, out_name2, sep="_", pattern="([ab])_(.*)"**)**
sep(separator) or pattern (regular expression) are used to specify columns to melt, and to parse input column names.
out_name1, out_name2: names for output columns (creates single value column), or value.name (creates a value columns for each unique part of the melted column name).

Apply function to cols.



APPLY A FUNCTION TO MULTIPLE COLUMNS

a	b	
1	4	
2	5	
3	6	

→

a	b
2	5

`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.

a
1
2
3

→

a	a_m
1	2
2	2
3	2

```
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

a	b
1	a
2	a
3	b

→

a	b	c
1	a	1
2	a	2
3	b	1

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

LAG & LEAD

a	b
1	a
2	a
3	b
4	b
5	b

→

a	b	c
1	a	NA
2	a	1
3	b	NA
4	b	3
5	b	4

`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.