Ana Lutzky,
journalist

// **Dataharvest, 30th September, 2020**

data.table library in R !

Read/write a file

```
Table = fread('myfile.csv')

fwrite(Table, file='myfile.csv')
```

- Fast (uses all the processors)
- Guesses the objects types, separators, or if there are column headers

Data.table : The syntax

dt[i, j, by]

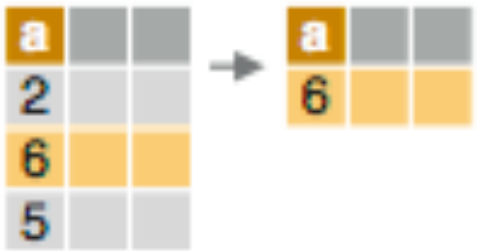
Take data.table **dt**,
subset rows using i
and manipulate columns with j,
grouped according to **by**.

Let’s start with rows

dt[i, j, by]

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

dt[a>5,]

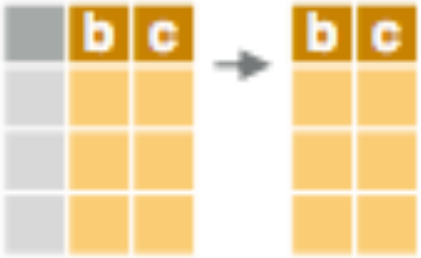


Now we go for columns

dt[i, j, by]

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

dt[, .(b,c)]



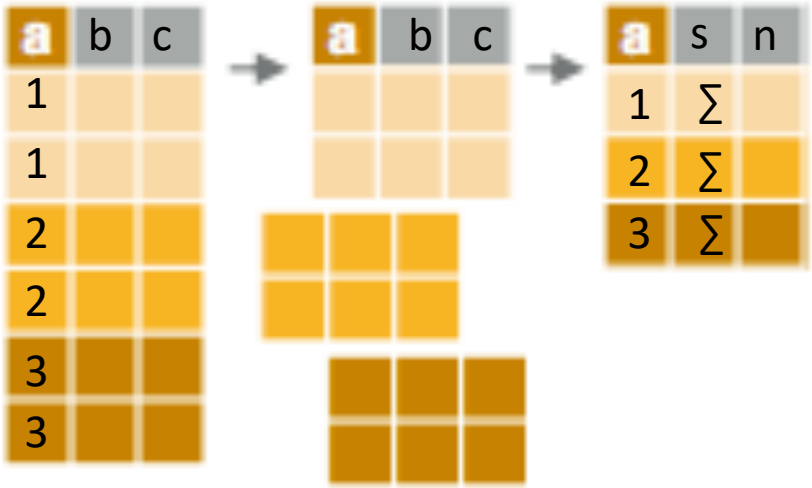
dt[, c := a + b]

Now let's group rows by

dt[, j, by = a]

dt[i, j, by]

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



dt[, .(amount = sum(b), number of purchases = length(b)), by= a]

dt[, .(amount = sum(c)), by=.(a, b)]

Reshaping a long format to wide

RESHAPE TO WIDE FORMAT

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

=> « spread » in tidyverse

Reshape a wide format to long

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

→

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

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

=> « gather » in tidyverse

// Dataharvest, 30th September, 2020

There’s a cheatsheet

- https://github.com/rstudio/cheatsheets/raw/master/datatable.pdf

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.

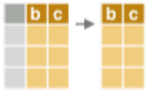
Subset rows using i

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

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.

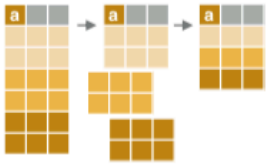
COMPUTE COLUMNS*

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.

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

THANKS !