

data.table

New Developments





Who Am I?

- Current "maintainer" of **data.table** (more on this at the end!)
- PhD in Quantitative Psychology (Bachelor's in Economics)
- **data.table** user since 2016, contributor since 2019
- Author/maintainer of 6 other R packages (3 on CRAN)
- Currently managing a team of researchers at Highmark Health (lots of big data wrangling and cleaning)
- CEO of Barrett Evaluation, LLC (big talk for I consult on projects with big data)



“dplyr will be the death of data.table”

An attendee said to Matt Dowle (creator of data.table) at an R Finance Conference a decade ago



Agenda

- Why (still) use **data.table** ?
- New developments!
 - New “management”
 - New features
 - New ways to engage





Why data.table ?

But there's new tools, why not use those??

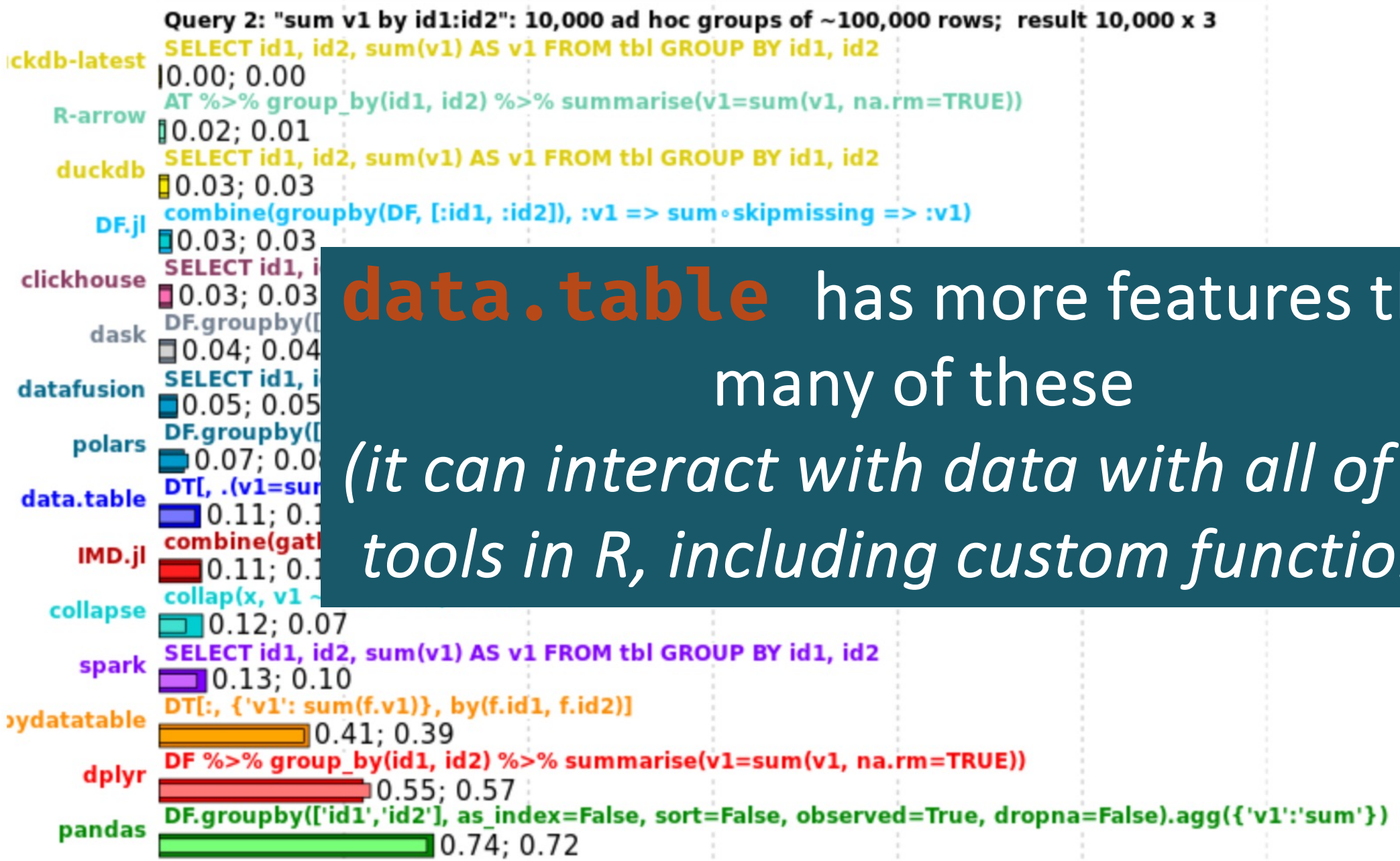
W

Query 2: "sum v1 by id1:id2": 10,000 ad hoc groups of ~100,000 rows; result 10,000 x 3

| Tool | Code | Time (min) | Time (sec) |
|---------------|-------------------------------------------------------------------------------------------------------------------|------------|------------|
| duckdb-latest | <code>SELECT id1, id2, sum(v1) AS v1 FROM tbl GROUP BY id1, id2</code> | 0.00 | 0.00 |
| R-arrow | <code>AT %>% group_by(id1, id2) %>% summarise(v1=sum(v1, na.rm=TRUE))</code> | 0.02 | 0.01 |
| duckdb | <code>SELECT id1, id2, sum(v1) AS v1 FROM tbl GROUP BY id1, id2</code> | 0.03 | 0.03 |
| DF.jl | <code>combine(groupby(DF, [:id1, :id2]), :v1 => sum(skipmissing => :v1))</code> | 0.03 | 0.03 |
| clickhouse | <code>SELECT id1, id2, sum(v1) AS v1 FROM tbl GROUP BY id1, id2</code> | 0.03 | 0.03 |
| dask | <code>DF.groupby(['id1','id2'], dropna=False, observed=True).agg({'v1':'sum'}).compute()</code> | 0.04 | 0.04 |
| datafusion | <code>SELECT id1, id2, SUM(v1) AS v1 FROM x GROUP BY id1, id2</code> | 0.05 | 0.05 |
| polars | <code>DF.groupby(['id1','id2']).agg(pl.sum('v1')).collect()</code> | 0.07 | 0.08 |
| data.table | <code>DT[, .(v1=sum(v1, na.rm=TRUE)), by=.(id1, id2)]</code> | 0.11 | 0.10 |
| IMD.jl | <code>combine(gatherby(x, [:id1, :id2], stable = false), :v1 => IMD.sum => :v1)</code> | 0.11 | 0.11 |
| collapse | <code>collap(x, v1 ~ id1 + id2, sum)</code> | 0.12 | 0.07 |
| spark | <code>SELECT id1, id2, sum(v1) AS v1 FROM tbl GROUP BY id1, id2</code> | 0.13 | 0.10 |
| pydatatable | <code>DT[:, {'v1': sum(f.v1)}, by(f.id1, f.id2)]</code> | 0.41 | 0.39 |
| dplyr | <code>DF %>% group_by(id1, id2) %>% summarise(v1=sum(v1, na.rm=TRUE))</code> | 0.55 | 0.57 |
| pandas | <code>DF.groupby(['id1','id2'], as_index=False, sort=False, observed=True, dropna=False).agg({'v1':'sum'})</code> | 0.74 | 0.72 |



W



data.table has more features than many of these
(it can interact with data with all of the tools in R, including custom functions)

ca
to



Why **data.table** ?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle management

Community

Feature rich



Why data.table?

Concise syntax

`dt[i, j, by]`

Fast speed

```
dt[grp == "treatment", new := mean(x), by = id]
```

```
dt[dt2, on = "id"]
```

```
dt[dt2, on = "id", roll = TRUE] # rolling joins!
```

```
dt[, .N, by = id]
```



Why **data.table** ?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle management

Community

Feature rich

Thoughtful and careful
so there are very few
breaking changes

Can be used in
production code safely



Why **data.table** ?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle management

Community

More on this in a moment!

Feature rich



Why data.table ?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle management

Community

Feature rich

Power of all of R + Gforce +
grouped optimization (more on
this later!)



New Developments!

- Grant from NSF (PI = Toby Hocking) to create new governance and support its development (NSF POSE program, project #2303612)
- Re-invigorated development and new features
- Ways to engage in development



New Developments!

- Grant from NSF (PI = Toby Hocking) to create new governance and support its development (NSF POSE program, project #2303612)

<https://github.com/Rdatatable/data.table/blob/master/GOVERNANCE.md>

N A semi-democratic approach to dev

- Can become any role in data.table by submitting PR and enough votes from the community
- Can help shape the development of the package
- One aspect of the governance is the “what is possible for development” which can be updated

d

New Developments!

- Re-invigorated development and new features

data.table v1.15.0 (30 Jan 2024)

BREAKING CHANGE

1. `shift` and `nafill` will now raise error `input must not be matrix or array` when `matrix` or `array` is provided on input, rather than giving useless result, [#5287](#). Thanks to @ethanbsmith for reporting.

NEW FEATURES

1. `nafill()` now applies `fill=` to the front/back of the vector when `type="locf|nocb"`, [#3594](#). Thanks to @ben519 for the feature request. It also now returns a named object based on the input names. Note that if you are considering joining and then using `nafill(..., type='locf|nocb')` afterwards, please review `roll= / rollends=` which should achieve the same result in one step more efficiently. `nafill()` is for when filling-while-joining (i.e. `roll= / rollends= / nomatch=`) cannot be applied.
2. `mean(na.rm=TRUE)` by group is now GForce optimized, [#4849](#). Thanks to the [h2oai/db-benchmark](#) project for spotting this issue. The 1 billion row example in the issue shows 48s reduced to 14s. The optimization also applies to type `integer64` resulting in a difference to the `bit64::mean.integer64` method: `data.table` returns a `double` result whereas `bit64` rounds the mean to the nearest integer.
3. `fwrite()` now writes UTF-8 or native csv files by specifying the `encoding=` argument, [#1770](#). Thanks to @shrektan for the request and the PR.
4. `data.table()` no longer fills empty vectors with `NA` with warning. Instead a 0-row `data.table` is returned, [#3727](#). Since `data.table()` is used internally by `.()`, this brings the following examples in line with expectations in most cases. Thanks to @shrektan for the suggestion and PR.

41 new features!

And several fixes and speed ups

New Developments!

- Re-invigorated development and new features

33. `DT[, let(...)]` is a new alias for the functional form of `:=`; i.e. `DT[, ':='](...)`, [#3795](#). Thanks to Elio Campitelli for requesting, and Benjamin Schwendinger for the PR.

```
DT = data.table(A=1:2)
DT[, let(B=3:4, C=letters[1:2])]
DT
#      A      B      C
# <int> <int> <char>
# 1:    1     3     a
# 2:    2     4     b
```



10. A new interface for *programming on data.table* has been added, closing [#2655](#) and many other linked issues. It is built using base R's `substitute`-like interface via a new `env` argument to `[.data.table]`. For details see the new vignette *programming on data.table*, and the new `?substitute2` manual page. Thanks to numerous users for filing requests, and Jan Gorecki for implementing.

```
DT = data.table(x = 1:5, y = 5:1)

# parameters
in_col_name = "x"
fun = "sum"
fun_arg1 = "na.rm"
fun_arg1val = TRUE
out_col_name = "sum_x"

# parameterized query
#DT[, .(out_col_name = fun(in_col_name, fun_arg1=fun_arg1val))]

# desired query
DT[, .(sum_x = sum(x, na.rm=TRUE))]

# new interface
DT[, .(out_col_name = fun(in_col_name, fun_arg1=fun_arg1val)),
  env = list(
    in_col_name = "x",
    fun = "sum",
    fun_arg1 = "na.rm",
    fun_arg1val = TRUE,
    out_col_name = "sum_x"
  )]
```

10. A new interface for *programming on data.table* has been added, closing [#2655](#) and many other linked issues. It is built using base R's `substitute`-like interface via a new `env` argument to `[.data.table]`. For details see the new vignette *programming on data.table*, and the new `?substitute2` manual page. Thanks to numerous users for filing requests, and Jan Gorecki for implementing.

```
DT = data.table(x = 1:5, y = 5:1)
```



```
# parameters
in_col_name = "x"
```

This is important because of non-standard evaluation

R needs to be told if this is an object or something it should look for inside of another object

```
# desired query
DT[, .(sum_x = sum(x, na.rm=TRUE))]

# new interface
DT[, .(out_col_name = fun(in_col_name, fun_arg1=fun_arg1val)),
  env = list(
    in_col_name = "x",
    fun = "sum",
    fun_arg1 = "na.rm",
    fun_arg1val = TRUE,
    out_col_name = "sum_x"
  )]
```

New Developments!

- Re-invigorated development and new features

```
library(data.table)
DT = data.table(x = 1:5, y = 5:1)

custom = function(dt, var_name, mutate){
  dt[, (var_name) := mutate]
  dt
}
custom(DT, "thing", x+y) # ERROR

custom2 = function(dt, var_name, mutate){
  dt[, (var_name) := mutate,
      env = list(mutate = substitute(mutate))]
  dt
}
custom2(DT, "thing", x+y) # works!
```

```
> DT
   x   y thing
<int> <int> <int>
1:   1   5     6
2:   2   4     6
3:   3   3     6
4:   4   2     6
5:   5   1     6
```



New Developments!

- Re-invigorated development and new features

17. `data.table` printing now supports customizable methods for both columns and list column row items, part of [#1523](#). `format_col` is S3-generic for customizing how to print whole columns and by default defers to the S3 `format` method for the column's class if one exists; e.g. `format_sfc` for geometry columns from the `sf` package, [#2273](#). Similarly, `format_list_item` is S3-generic for customizing how to print each row of list columns (which lack a `format` method at a column level) and also by default defers to the S3 `format` method for that item's class if one exists. Thanks to [@mlg](#) who initially filed [#3338](#) with the seed of the idea, [@franknarf1](#) who earlier suggested the idea of providing custom formatters, [@fparages](#) who submitted a patch to improve the printing of timezones for [#2842](#), [@RichardRedding](#) for pointing out an error relating to printing wide `expression` columns in [#3011](#), [@JoshOBrien](#) for improving the output for geometry columns, and [@MichaelChirico](#) for implementing. See `?print.data.table` for examples.



New Developments!

- Ways to engage in development

1 GitHub Issue Tracker



New Developments!

- Ways to engage in development

2

“Seal of Approval”

<https://github.com/Rdatatable/data.table/issues/5723>



New Developments!

- Ways to engage in development

3 Vote on GitHub Pull Requests

New Developments!

- Ways to engage in development

4 Talk, publish about it



Tyson S. Barrett

Thanks to **Matt Dowle** and **Arun Srinivasan** and **the data.table team!**



t.barrett88@gmail.com



tysonbarrett.com



@healthandstats



github.com/tysonstanley



Slides at tysonbarrett.com/teaching