What's new in `data.table`?

Why R?, 3rd July, 2018, Wroclaw, Poland

Jan Gorecki

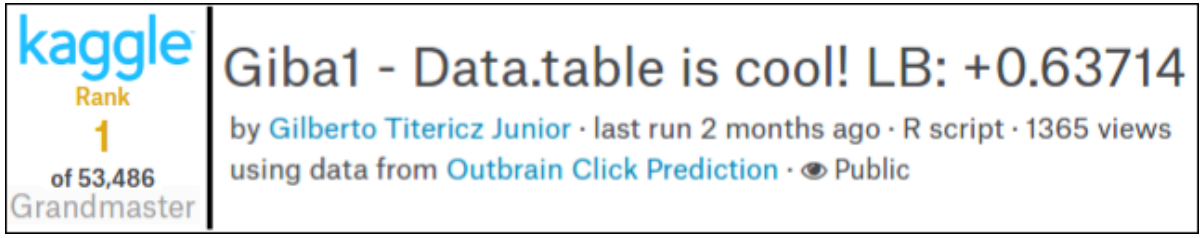
What is `data.table`?

An R package. Extension of `data.frame`.
Not strictly for machine learning, but for data cleaning, preparation, feature engineering.

What is `data.table`?

An R package. Extension of `data.frame`.
Not strictly for machine learning, but for data cleaning, preparation, feature engineering.

popular among data science community



What is `data.table`?

An R package. Extension of `data.frame`.
Not strictly for machine learning, but for data cleaning, preparation, future engineering.

how is `data.table` related to `mlr`?

`mlr` package imports `data.table` to speed up data processing:

- aggregate by group
- `melt` (transform data from columns to rows)
- *sort*
- *top N by group*

Actually it could speed up many more processes in `mlr`.

What is so special about `data.table`?

What is so special about `data.table`?

- faster speed
  - focus on implementation using efficient algorithms, some of `data.table` internals have later been incorporated into base R
  - using indexes, keys
  - using fewer in-memory copies also saves time

What is so special about `data.table`?

- faster speed
  - focus on implementation using efficient algorithms, some of `data.table` internals have later been incorporated into base R
  - using indexes, keys
  - using fewer in-memory copies also saves time
- less memory usage - not only related to *by reference* operations (`:=`, `set*`) but in general!
  - memory efficient algorithms
  - grouping allocates memory for biggest group and reuses it
  - join algorithm uses *sorting* instead of *hashing*
  - join and grouping (`by=.EACHI`) do not materialize intermediate join results
  - *by reference* operations avoid unnecessary in-memory copies (`:=`, `set`, `setkey`, `setDT`, `setcolorder`, ...)

What is so special about `data.table`?

- faster speed
  - focus on implementation using efficient algorithms, some of `data.table` internals have later been incorporated into base R
  - using indexes, keys
  - using fewer in-memory copies also saves time
- less memory usage - not only related to *by reference* operations (`:=`, `set*`) but in general!
  - memory efficient algorithms
  - grouping allocates memory for biggest group and reuses it
  - join algorithm uses *sorting* instead of *hashing*
  - join and grouping (`by=.EACHI`) do not materialize intermediate join results
  - *by reference* operations avoid unnecessary in-memory copies (`:=`, `set`, `setkey`, `setDT`, `setcolorder`, ...)
- syntax
  - concise
  - corresponding to SQL queries: `FROM[where|orderby, select, groupby]`
  - chaining using `[`, turns above query into sub-query:

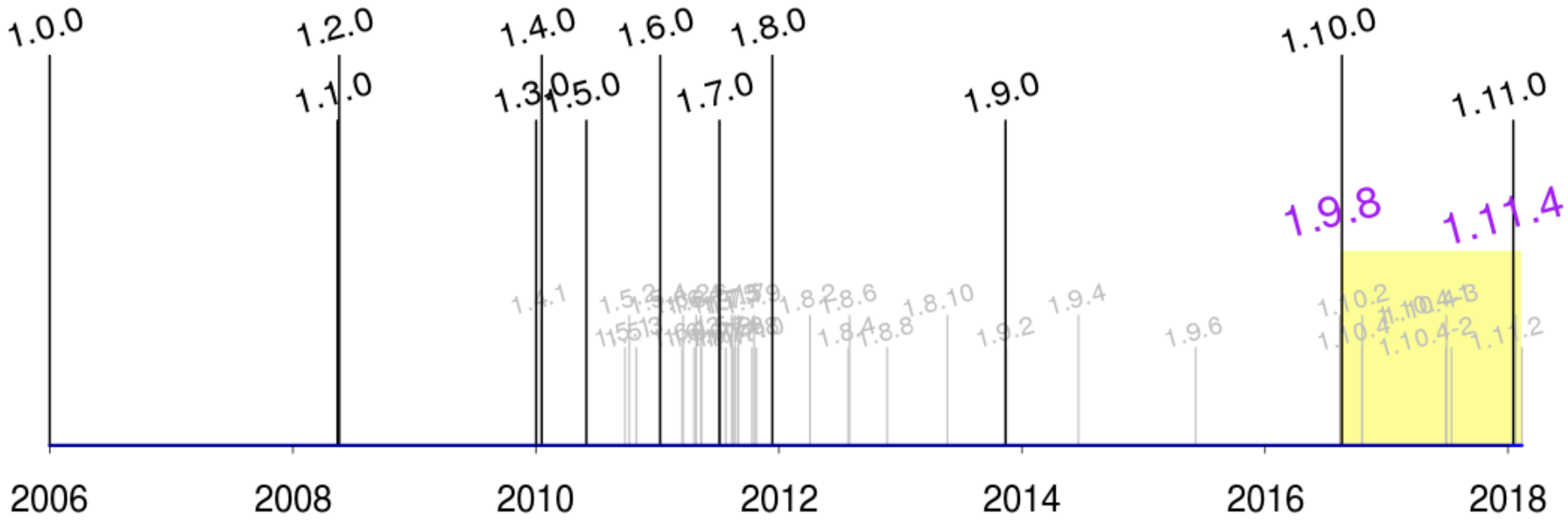
    ```
    FROM[sub-query][outer-query][...][most-outer-query]
    ```

  - for some people too concise

So what is new?

As of this writing, the `data.table` R package is 12 years old.
Focus of this presentation covers last 2 years of development, starting from 1.9.8 version.

`data.table` development timeline:



parallel processing with OpenMP

In short OpenMP is a programming interface for shared memory multiprocessing. Works on most platforms and operating systems.
First used in `data.table` in version 1.9.8 (Nov 2016) in `fwrite` function, a fast csv writer.
`data.table` does not now depend on or require OpenMP. If you don't have it then `data.table` should build, run and pass all tests just fine.
Control over cores used by `data.table` is provided by following functions:

```
setDTthreads(0) # use all available cores (default)
getDTthreads()
```

`fwrite` fast csv writer

`fwrite` was the first function to use OpenMP in `data.table`. By default it uses all available CPU cores. Together with `fread` it practically allows for csv files to be a medium for data exchange, making data in/out competitive with binary formats. `fwrite` is not only fast but also feature rich. Additionally we test that `fread(fwrite(DT))==DT` is true.
Below benchmark results are from April 2016; read more in H2O.ai blog: Fast csv writing for R blog post.

		Laptop SSD 4core/16gb 10m rows		Server 32core/256gb 100m rows		
		Time Sec	Size GB	RamDisk Time	HDD Time	Size GB
`fwrite(DT,"fwrite.csv")`	csv	2	0.8	9	61	7.5
`write_feather(DT, "feather.bin")`	bin	5	1.0	27	75	9.1
`save(DT,file="save1.Rdata",compress=F)`	bin	11	1.2	90	137	12.0
`save(DT,file="save2.Rdata",compress=T)`	bin	70	0.4	647	679	2.8
`write.csv(DT,"write.csv.csv",**)`	csv	63	0.8	749	824	7.3
`readr::write_csv(DT,"write_csv.csv")`	csv	132	0.8	1997	1571	7.3

Initial single threaded version of `fwrite` was contributed by Otto Seiskari in PR#1613: *First version of the fwrite function*.

non-equi joins

Non-equi joins have been available since version 1.9.8.

```
# equi join
dt2[dt1, on=.(ID, Date3==Date1, Date3==Date2)]
# non equi join
dt2[dt1, on=.(ID, Date3>=Date1, Date3<=Date2)]
# join and grouping also supported
dt2[dt1, .(sum = sum(Value)), on=.(ID, Date3>=Date1, Date3<=Date2), by=.EACHI]
```

More info in Efficient in-memory non-equi joins presentation.
Tons of examples can be found in Non equi joins Q on SO wiki:

Date	Link	Type
29.06.2014	http://stackoverflow.com/q/24480031/559784	
01.10.2014	http://stackoverflow.com/q/26134707/559784	

21.11.2014	https://stackoverflow.com/q/27059924/3817004	replace combn() by non-equi join
01.10.2015	http://stackoverflow.com/q/32893022/559784	
24.11.2015	http://stackoverflow.com/q/33905020/559784	
22.02.2016	http://stackoverflow.com/q/35565149/559784	range join, nomatch=0L
01.03.2016	http://stackoverflow.com/q/35713958/559784	range join, nomatch=0L
23.03.2016	http://stackoverflow.com/q/36190921/559784	
29.03.2016	http://stackoverflow.com/q/3628473/559784	
06.04.2016	http://stackoverflow.com/q/36454565/559784	

06.12.2016	http://stackoverflow.com/q/41007099/559784	multiple conditions, by=.EACHI, nomatch=0L
07.12.2016	http://stackoverflow.com/q/41024867/559784	multiple conditions
08.12.2016	http://stackoverflow.com/q/41043047/559784	multiple conditions, nomatch=0L
15.12.2016	http://stackoverflow.com/q/41164723/559784	multiple conditions, :=
22.12.2016	http://stackoverflow.com/q/41279229/559784	multiple conditions
03.01.2017	http://stackoverflow.com/q/41450543/559784	multiple conditions, by=.EACHI
08.01.2017	http://stackoverflow.com/q/41537187/559784	multiple conditions,

`fread` improvements

- parallel reading using OpenMP
- improved quote rules
- memory maps lazily; e.g. reading just the first 10 rows with `nrow=10` is 12s down to 0.01s

```
# shell
wc -l X1e9_2c.csv
#1000000001 X1e9_2c.csv
head -2 X1e9_2c.csv
#KEY,X2
#632858426,109153997
```

```
# R
getDTthreads()
#[1] 20
system.time(ans<-fread("X1e9_2c.csv"))
#   user  system elapsed
# 72.790   4.886   4.666
setDTthreads(1)
system.time(ans<-fread("X1e9_2c.csv"))
#   user  system elapsed
# 34.849   2.744  37.592
```

More info in Parallel fread presentation.

control scope of variables in `DT[...]`

`with=FALSE` no longer needed to select columns

```
DT[, 1:2, with=FALSE]                # before
DT[, c("colA","colB"), with=FALSE]
DT[, 1:2]                            # now
DT[, c("colA","colB")]
```

using `..` prefix

We added handling of double dot prefix `..myCols` which explicitly ask for `myCols` variable to be taken from parent scope. This allows to overcome to problem with overlapping names of variables in current scope and in your dataset.

```
myCols = c("colA","colB") # from 1.10.2
DT[, myCols, with=FALSE]
DT[, ..myCols]            # same

cols = "colB"             # from 1.11.0 also
DT[, c(..cols, "colC")]   # same as DT[, .(colB,colC)]
DT[, -..cols]             # all columns other than colB
```

set operators

Group of functions to perform *set theory* operations on `data.table`s.
Set operators in `data.table` can handle duplicated rows using `all=TRUE` argument.

data.table	set theory	SQL
`fintersect`	`tbl1 ∩ tbl2`	`select * from tbl1 INTERSECT select * from tbl2`
`fsetdiff`	`tbl1 \ tbl2`	`select * from tbl1 EXCEPT select * from tbl2`
`funion`	`tbl1 ∪ tbl2`	`select * from tbl1 UNION select * from tbl2`

There is also `fsetequal` to tests equality of sets.

More info in `?setops`.

grouping sets

Calculate subtotals along with regular grouping.

data.table	SQL
`rollup`	`select sum(v3) from tbl1 GROUP BY ROLLUP (v1, v2)`
`cube`	`select sum(v3) from tbl1 GROUP BY CUBE (v1, v2)`
`groupingsets`	`select sum(v3) from tbl1 GROUP BY GROUPING SETS (v1, v2)`

Example on `mtcars`:

```
dt = as.data.table(mtcars)
rollup(dt, .(mean_mpg=mean(mpg)), by=c("vs","am"))
#    vs am mean_mpg
#1:  0  1 19.75000
#2:  1  1 28.37143
#3:  1  0 20.74286
#4:  0  0 15.05000
#5:  0 NA 16.61667
#6:  1 NA 24.55714
#7: NA NA 20.09062
```

More info in `?groupingsets`.

subset uses OpenMP

Useful when doing big subset of data.
Data 200M rows x 4 cols (4.5 GB), subset 100M rows.

```
prettyNum(dim(x), big.mark=",")
#[1] "200,000,000" "4"
head(x, 2)
#   a     b        c     d
#1: l 52512 0.084814 FALSE
#2: k  2198 0.119069 FALSE
prettyNum(length(ix), big.mark=",")
#[1] "100,000,000"
head(ix)
#[1]  64721375  72207325 189934101  57977103  21834433  67429500
getDTthreads()
#[1] 20
system.time(.Call("CsubsetDT", x, ix, 1:4))
#   user  system elapsed
# 12.611   0.708   4.528
setDTthreads(1)
system.time(.Call("CsubsetDT", x, ix, 1:4))
#   user  system elapsed
# 10.948   0.648  11.596
```

This is not yet hooked into `[.data.table` operator, follow #2951.

subset uses indices on multiple columns

Previously index optimization was applied only when filtering was made on single column, now indexes are utilized also in compound subset queries.

```
set.seed(108)
N = 1e8
DT = setDT(list(sample(N/10, N, TRUE), sample(letters, N, TRUE)))
setindexv(DT, c("V1","V2"))
options("datatable.use.index"=TRUE)
system.time(ans1<-DT[V1 %in% 1000:1002 & V2 %in% c("a","b","c")])
#   user  system elapsed
#  0.001   0.000   0.002
options("datatable.use.index"=FALSE)
system.time(ans2<-DT[V1 %in% 1000:1002 & V2 %in% c("a","b","c")])
#   user  system elapsed
#  4.051   0.848   4.899
```

Currently limited to & (*AND*) operator to combine fields and `==`, `%in%` as filter operator on each field.

Read more about *index* performance in Scaling data.table using index blog post.

This extension of subset using index was contributed by Markus Bonsch in PR#2494: *Better subsetting optimization for compound queries*.

new vignettes

- Keys and fast binary search based subset
  Comprehensive description on how to use `data.table`'s *key* to unleash power of binary search in place of vector scan. This vignette also explains the difference between vector scan and binary search.

- Secondary indices and auto indexing
  Explains how to use `data.table`'s *index*, a feature similar to *key* but does not need to reorder data.

- Importing `data.table`
  This document is focused on using `data.table` as a dependency in other R packages. Importing `data.table` is no different from importing other R packages. This vignette meant to answer most common questions which popups around that subject. Defining dependency presented here can be applied to other R packages.

- Benchmarking `data.table`
  Vignette is meant to guide on measuring performance of `data.table`. Single place to document best practices and traps to avoid.

stay up to date

New helper function to stay up to date with recent *stable development* version of `data.table`.
It compares git commit hash of installed `data.table` before downloading from our CRAN-like repository hosted in `gh-pages` branch of Rdatatable/data.table GitHub repository.

```
data.table::update.dev.pkg()
```

If you are on Windows without Rtools you can get binaries from alternative repo where we currently publishing 3.5 and 3.6 (devel) windows binaries.

```
data.table::update.dev.pkg(repo="https://Rdatatable.gitlab.io/data.table")
```

As always you can install recent *stable development* version of `data.table` using standard `install.packages` call.

```
# installing from source - requires Rtools on Windows
install.packages("data.table", repos="https://Rdatatable.github.io/data.table") # GitHub
# installing from binaries
install.packages("data.table", repos="https://Rdatatable.gitlab.io/data.table") # GitLab
```

More info in Installation wiki.

License change to *Mozila Public License*

Change reflects our intentions about using `data.table` in closed sourced software to be more permissive than GPL allows. Reasons should be read in full in PR#2456: *License change from GPL to MPL*.

License change from GPL to MPL #2456

Edit

Merged **mattdowle** merged 5 commits into `master` from `license` on 7 Nov 2017

Conversation 13 | Commits 5 | Checks 0 | Files changed 3 | +376 −676

mattdowle commented on 1 Nov 2017 • edited by jangorecki — Member

This pull request is a verbatim copy of an email I sent to all 24 contributors of code to the project. All 24 have now approved with either a review or a thumbs-up. Thanks everyone!

Dear contributor to data.table,

Since its creation in 2006, data.table has been GPL. You are receiving this email because you have contributed either R or C code to the data.table project (even one line) and are considered a license holder. Contributors to documentation-only have not been included. Myself and Arun (because we are the biggest contributors) have tentatively agreed, subject to all your approvals and further comments, that it was never

Reviewers

- alamit
- lianos ✓
- tshort ✓
- eantonya ✓
- mllg ✓
- rsaporta ✓
- oseiskar ✓
- arunsrinivasan ✓

python `datatable`

Python `datatable` started in 2017 as a toolkit for performing big data operations on a single-node machine, at the maximum speed possible. Such requirements are dictated by modern machine-learning applications, which need to process large volumes of data and generate many features in order to achieve the best model accuracy. The first user of `datatable` was Driverless.ai.

```
import datatable as dt
from datatable import f, sum
DT = dt.fread("X1e9_2c.csv")
DT[f.X2 > 0, sum(f.X2), f.KEY]
     KEY          V0
---  ---  ----------
 0     2   103775909
 1     3   871252780
 2     4   410188763
...

[393,476,905 rows x 2 columns]
```

Project repository: github.com/h2oai/datatable

H2O Driverless AI

H2O.ai's latest enterprise offering

- Created and supported by world renowned AI experts
- Empowers companies to accomplish AI and ML with a single platform
- Performs the function on an expert data scientist and adds more power to both novice and expert teams
- Details and highlights insights and interpretability with easy to understand results and automatic visualizations

Think of it as ML with H2O + GPU + automatic tuning and feature engineering = **Kaggle grandmaster in a box**.

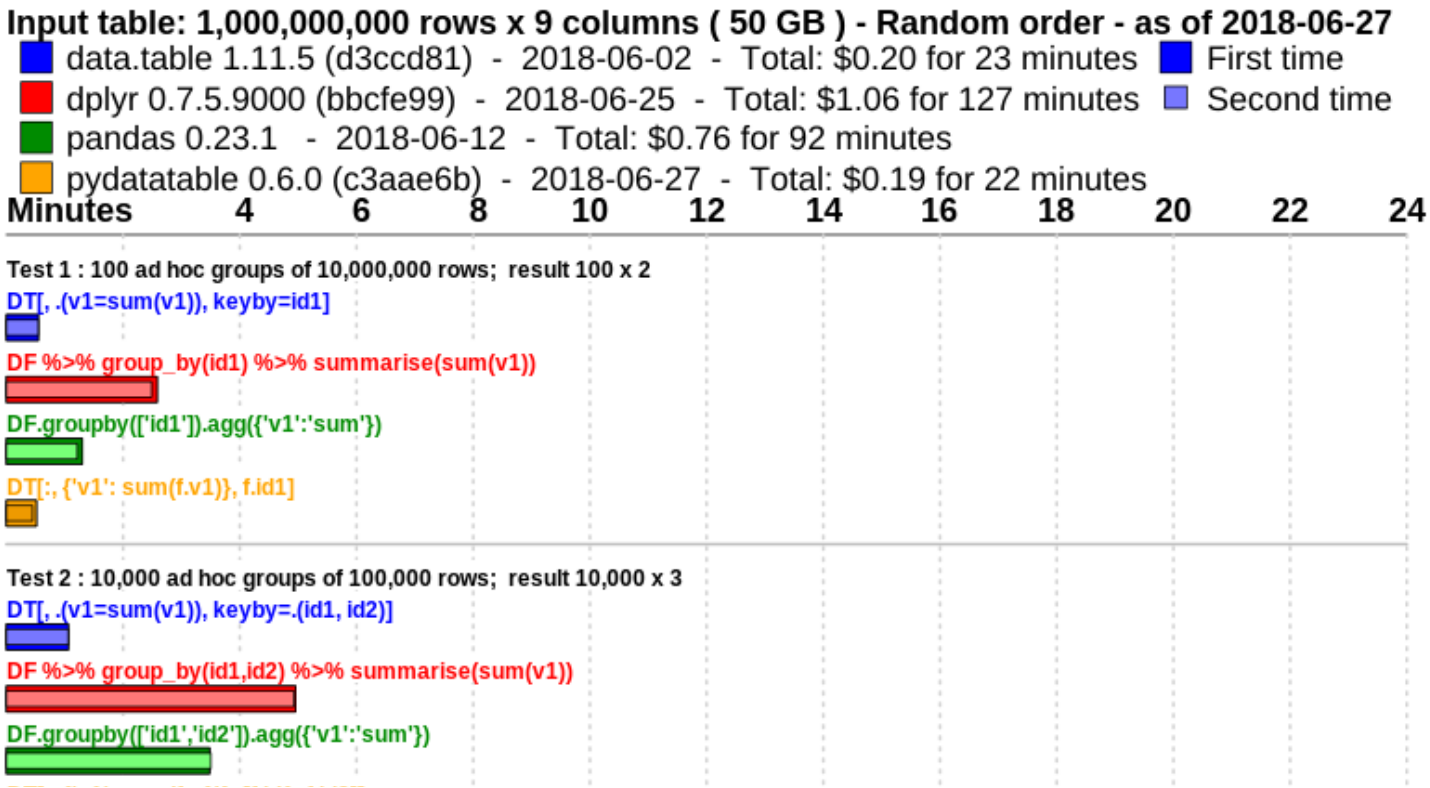
- GPU acceleration to achieve up to 40x speedups
- Automatic feature engineering to increase accuracy
- Automatic machine learning to find and tune the right ensemble of models
- Emphasize Interpretability

Read more at: h2o.ai/driverless-ai

21 day free trial for Driverless AI

Benchmarking

In H2O.ai we care about performance, we are continously benchmarking our tools.
`data.table` Grouping Benchmark from 2014 has been updated to reflect recent versions of tools, also python `datatable` has been added.
More info in github.com/h2oai/db-benchmark.



Thank You!

You are very welcome to visit our GitHub repository, provide feedback, upvote existing feature requests, create new FR, or a pull request: github.com/Rdatatable/data.table | r-datatable.com

Also thank to users who were filling issues, providing reproducible examples, debugging, and submitting pull requests:

```
franknarf1, MarkusBonsch, DavidArenburg, jsams, etienne-s, tdhock, rsaporta, dselivanov, mgahan,
sritchie73, ebs238, Henrik-P, caneff, shrektan, heavywatal, chris, mplatzer, ladida771, mllg, cguill95,
scottstanfield, skanskan, javrucebo, yaakovfeldman, cnoelke, qinjs, memoryfull, brandenkmurray,
dracodoc, rcapell, TMOTTM, mrdwab, scarrascoso, gnguy, lbilli, aaronmcdaid, sergeganakou, skranz,
royalts, renkun-ken, ProfFancyPants, MoebiusAV, SimonCoulombe, Mihael, pannnda, patrickhowerter, osofr,
kmillar, hatal175, ecoRoland, jan-glx, peterlittlejohn, hughparsonage, asenabouth, fupangpangpang,
jmosser, demydd, neomantic, ambils, manimal, Pascal, DirkJonker, dbetebenner, kendonB, MattWeller,
enfascination, maverickg, bthieurmel, STATWORX, richierocks, StephenMcInerney, geneorama, JoshuaUlrich,
sebastian-c, jaapwalhout, marc-outins, DexGroves, LyssBucks, AmyMikhail, StefanFritsch, Arthur,
alexdeng, huashan, Max, rBatt, damienchallet, alexkowa, VasilyA, AnandaMahto, kimiylilammi,
AmitaiPerlstein, richardtessier, talexand, bryan4887, sergiizaskaleta, tdeenes, slowteetoe, hshipper,
vlsi, nigmastar, DouglasClark, fabiangehring, ChristK, nachti, wligtenberg, restonslacker,
daniellemccool, rodonn, m-dz, fc9.30, fruce-ki, ywhuofu, dlithio, abielr, dougedmunds, ems, Zus,
sz-cgt, GRandom, DavidArenberg, aushev, chenghlee, pstoyanov, Roland, rajkrpan, smcinerney,
SymbolixAU, DCEmilberg, rrichmond
```

And many more, listed without leading @ in NEWS.md file.

Contact me at: `j.gorecki _in_ wit.edu.pl` | github.com/jangorecki | gitlab.com/jangorecki

You can find this presentation in data.table/wiki/Presentations and my Git**Lab** `r-talks` page jangorecki.gitlab.io/r-talks