



Processing large-scale data efficiently: An introduction to the R package 'data.table'.

Research Data Scotland

Bayes Centre
University of Edinburgh
15th August 2024

Andreas Hoehn

University of Glasgow



andreas.hoehn@glasgow.ac.uk



[AndreasXHoehn](https://github.com/AndreasXHoehn)



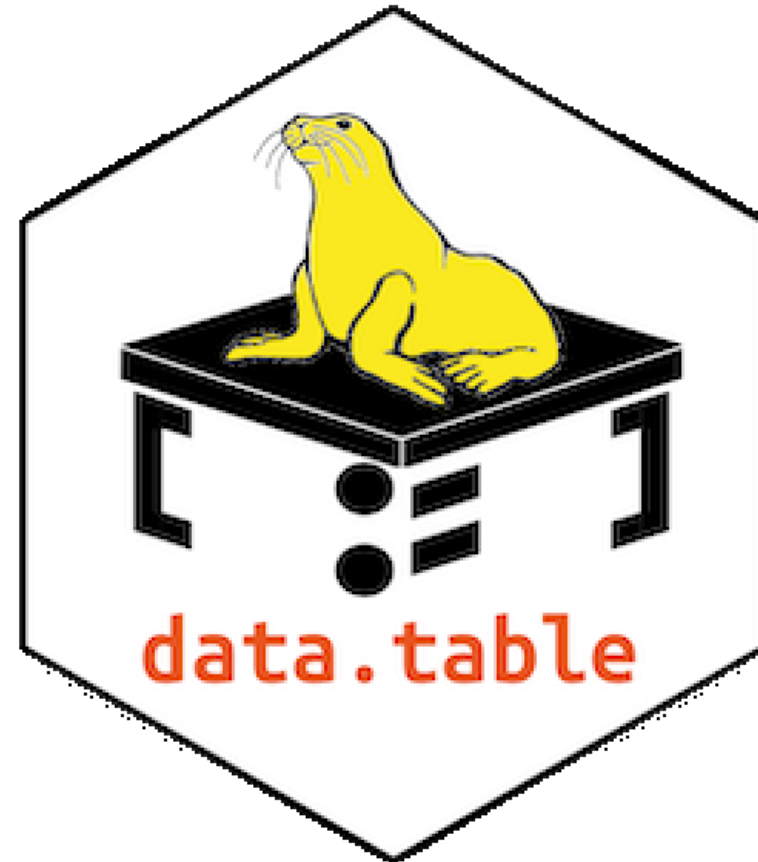
[0000-0002-7170-1205](https://orcid.org/0000-0002-7170-1205)

Outline

Introducing 'data.table'

Hands-on Session

Q&A



Source: <https://rdatatable.gitlab.io/data.table/>



Introducing 'data.table'

What will be faster: A Ferrari or a Honda?

Introducing 'data.table'

What will be faster: A Ferrari or a Honda?



Photo by [Stefano Probst](#) on [Unsplash](#)

VS.



Photo by [Brad armore](#) on [Unsplash](#)

Introducing 'data.table'

Apple-to-apple comparisons are required!

Task

groupby join groupby2014

0.5 GB 5 GB 50 GB

basic questions

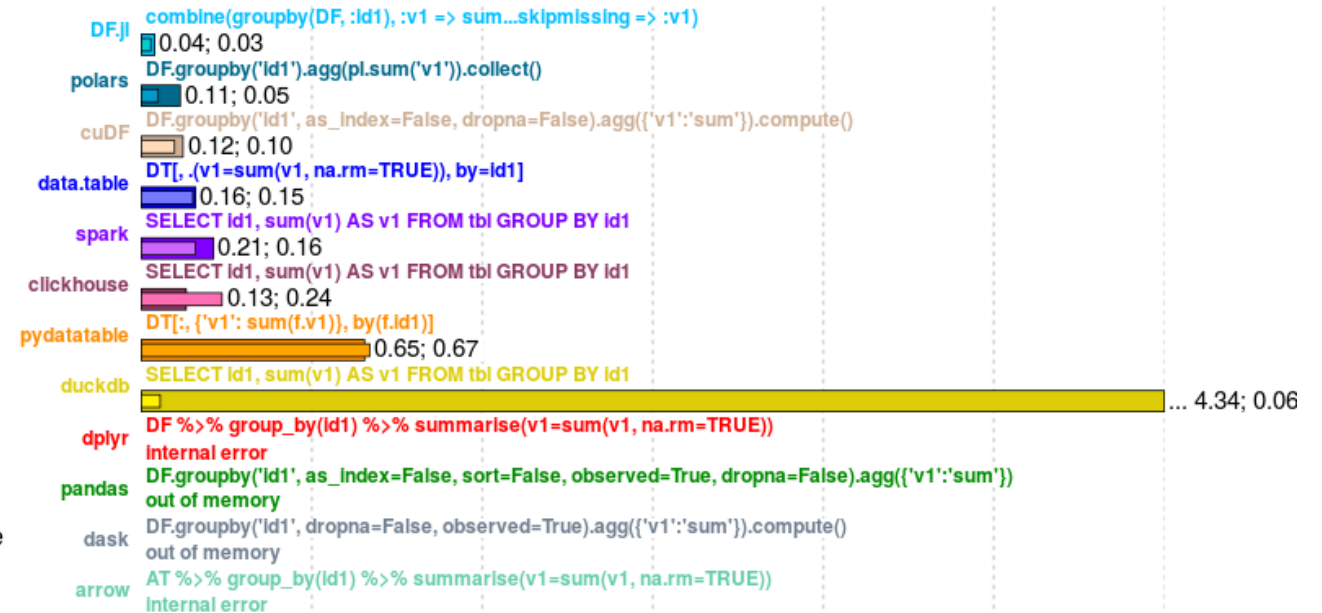
Input table: 1,000,000,000 rows x 9 columns (50 GB)

Polars	0.8.8	2021-06-30	143s
data.table	1.14.1	2021-06-30	155s
DataFrames.jl	1.1.1	2021-05-15	200s
ClickHouse	21.3.2.5	2021-05-12	256s
cuDF*	0.19.2	2021-05-31	492s
spark	3.1.2	2021-05-31	568s
(py)datatable	1.0.0a0	2021-06-30	730s
dplyr	1.0.7	2021-06-20	internal error
pandas	1.2.5	2021-06-30	out of memory
dask	2021.04.1	2021-05-09	out of memory
Arrow	4.0.1	2021-05-31	internal error
DuckDB*	0.2.7	2021-06-15	out of memory
Modin		see README	pending

First time
Second time

Minutes 0.5 1.0 1.5 2.0 2.5 3.0

Query 1: "sum v1 by Id1": 100 ad hoc groups of ~10,000,000 rows; result 100 x 2



Source: <https://h2oai.github.io/db-benchmark/>



Introducing 'data.table'

Scene Setting

1) Routinely collected data has always been large - and will only get larger (e.g. high-dimensional smart data, real-time data, omics data, CTGAN synthetic data)

2) Memory (RAM) and processing capacity (CPU) have remained limited resources

While R is a fantastic programming language, “standard” R (e.g. through base or tidyverse) is not an efficient/safe/futureproof/portable/updateable/..... way of working

Even worse! The limitations of “standard” R will fuel the problems that come from 1) and 2)

Introducing ‘data.table’

“Standard” R’s limitations

- 1) By default, R uses 1 core – but many more are available on most machines (→ CPU + time)
- 2) By default, R has a “copy-on-modify behaviour” (→ RAM + time)
- 3) Dependencies, portability, backward compatibility: Updates vs. old code (→ lots of time)
- 4) R Code can get long, especially when using long chains of pipes (→ even more time ...)

Introducing 'data.table'

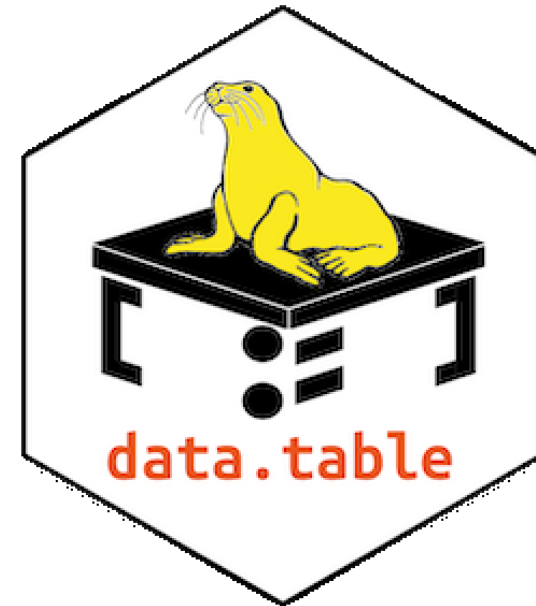
“Standard” R’s limitations **vs. data.table**

- 1) By default, R uses 1 core – but many more are available on most machines (→ CPU + time)
data.table parallelises whenever this is easily done, running compiled C/C++ underneath
- 2) By default, R has a “copy-on-modify behaviour” (→ RAM + time)
data.table modifies on reference (“ := ”) and uses pointers, requiring less working RAM
- 3) Dependencies, portability, backward compatibility: Updates vs. old code (→ lots of time)
data.table has no dependencies other than R >= Version 3.1 (10+ years old)
- 4) R Code can get long, especially when using long chains of pipes (→ even more time ...)
data.table has short and expressive code, similar to high-level programming languages

Introducing 'data.table'

What is data.table?

- 1) **A selection of functions**, optimised to work efficiently with large amounts of data (e.g., fread, fwrite, or for reshaping long-wide / wide-long)
- 2) **A separate dialect for R**, with some degree of similarity to SQL or C/C++ code
- 3) **A unique chance to learn about R and computing**, with the aim of improving processes and futureproofing our work



Source: <https://rdatatable.gitlab.io/data.table/>



Introducing 'data.table'

Standard 'data.table' notation

Standard format 1: `data[i , j]` → basic format when not operating on groups

Standard format 2: `data[i , j , by]` → when operating by group

"data" is our dataset

"i" subset of "data" based on row information ("subset on observations/rows")

"j" states what to execute for the columns ("execute on variables/columns")

"by" defines whether "i" and "j" should be done by groups

Introducing 'data.table'

How to read data.table?

`data[sex == "Male",]` → subset where sex is "Male", nothing to execute

`data[, V2 := V1+1]` → nothing to subset, create a new variable "V2" which is "V1+1"

`data[sex == "Male", V2 := V1+1]` → subset where sex is "Male", then create V2...

`data[sex == "Male", .(V2 = max(V1))]` → subset where sex is "Male", then return a new variable

`data[, .(V2 = max(V1)), by = c("sex")]` → no subset, return new variable by group ("sex" variable)



Introducing 'data.table'

Any guesses?

Situation: A three-variable dataset (ID, age, income) reflecting yearly income data (repeated obs.)

```
data2 <- data[age >= 16, .(income_median = median(income)), by = c("ID")]
```

Introducing 'data.table'

Any guesses?

Situation: A three-variable dataset (ID, age, income) reflecting yearly income data (repeated obs.)

```
data2 <- data[age >= 16, .(income_median = median(income)), by = c("ID")]
```

- ... assign to a new object data2 something that comes out of data, new data will have 1 row per ID
- ... subset is what comes before the comma: select those aged 16 or older
- ... the new variable "income_median" is the median income of all recorded incomes of the subset
- ... which was established separately for all "IDs" (e.g., there are multiple income records per "ID")



Hands-on Session

All course materials are available on GitHub!

No account required, a .zip bundle can be downloaded

Repository: 2024_RDS_DT



<https://github.com/AndreasxHoehn>



Hands-on Session

Objectives: Big Picture + Transferable Skills + Strong Foundation

- 1) Benchmarking time: `'microbenchmark::microbenchmark()'`
- 2) Benchmarking memory: `'object.size()'` and variable types
- 3) Tracing the location of objects within `'tracemem()'`
- 4) Introduction to `'data.table'` - basic functions, subsets, creating new variables, group by operations, reshaping data "wide to long" & "long to wide"



Further Resources

data.table on cran: <https://cran.r-project.org/web/packages/data.table/>

Benchmarking data.table operations: https://tysonbarrett.com//jekyll/update/2019/10/06/datatable_memory/

Benchmarking joins: https://tysonbarrett.com/jekyll/update/2019/10/11/speed_of_joins/

A good basic data.table intro: <https://atrebas.github.io/post/2020-06-17-datatable-introduction/>

The official data.table FAQ: <https://cran.r-project.org/web/packages/data.table/vignettes/datatable-faq.html>



Q&A

Questions – Comments – Feedback?



THANK YOU FOR LISTENING

Find out more

Website: www.sipher.ac.uk

Email: sipher@Glasgow.ac.uk

Twitter: [@SipherC](https://twitter.com/SipherC)