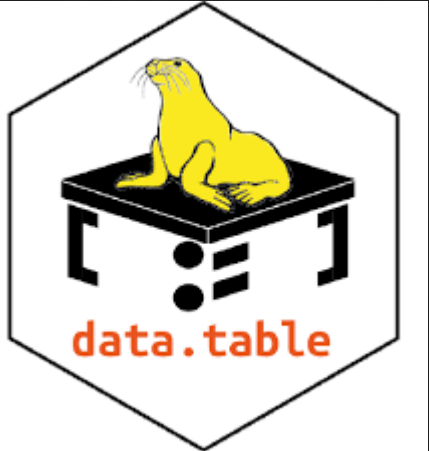
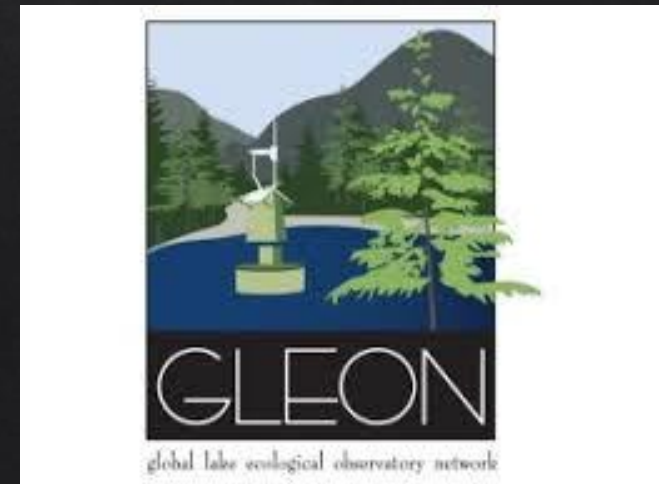


Introduction to data.table

Fast data handling and concise syntax in R



Jorrit Mesman
University of Geneva / Uppsala University
August 2021

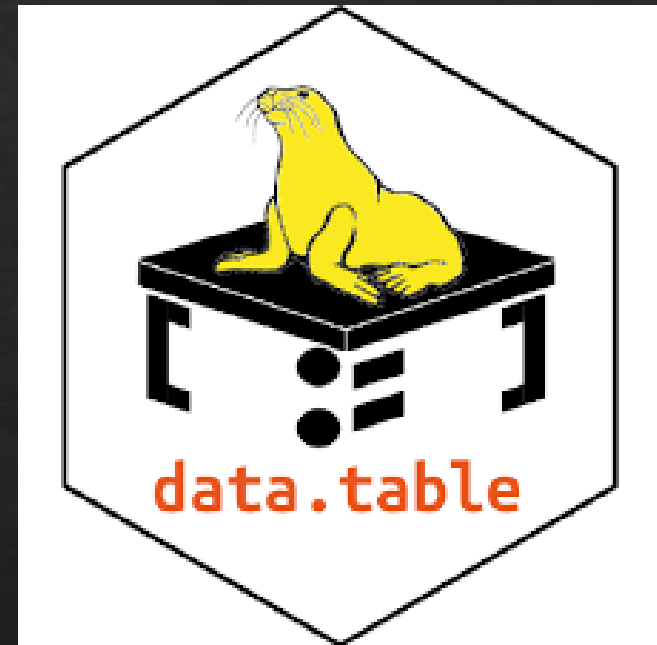


Workshop Outline

- ◆ Introductory presentation (20 min.)
 - ◆ What is data.table and why should you use it?
- ◆ Questions (5 min.)
- ◆ Walkthrough script – Part 1 (20 min.)
- ◆ Break (10 min.)
- ◆ Walkthrough script – Part 2 (20 min.)
- ◆ Questions & Try it out yourself! (15 min.)

What is data.table?

- ◇ CRAN R package
 - ◇ Main developer: Matt Dowle
 - ◇ First release: 2006
 - ◇ Active development
- ◇ Extension to data.frame
- ◇ Fast data handling
 - ◇ Main focus of the package
 - ◇ Reading, writing, selecting, joining, etc.
- ◇ Consistent and concise syntax



Website: <https://rdatatable.gitlab.io/data.table/>
Twitter: @MattDowle, #rdatatable

Why this workshop?

- ◇ One of the main packages in R, but little advertised
- ◇ Can help you write clearer code and faster scripts
- ◇ Faster data handling is particularly useful for GLEONites!
 - ◇ High-frequency data and long-term datasets
 - ◇ Modelling
 - ◇ Global analyses

Performance of data.table

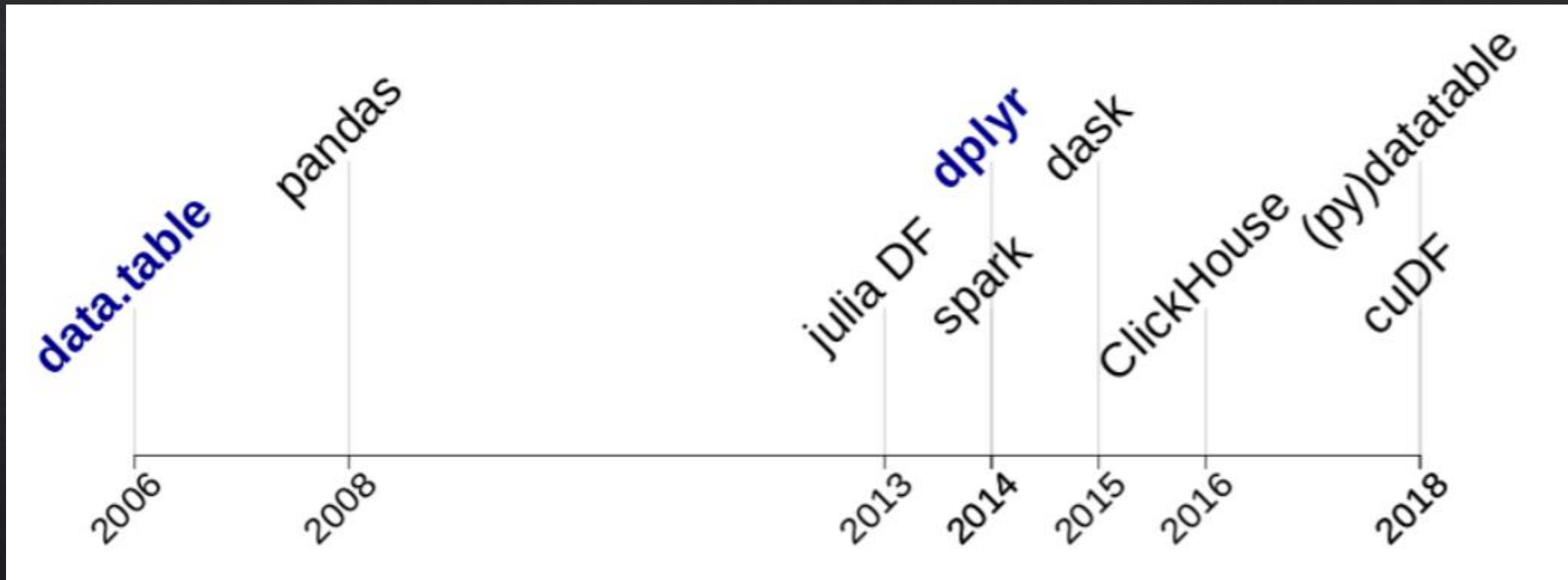
- ◇ It is often said that R is slow
- ◇ Whether speed is relevant depends on the size of your data and how often you will call your function
 - ◇ Single use:
 - ◇ Small datasets (<10MB), speed is not so important
 - ◇ Large datasets (hundreds of MB), speed matters a lot
 - ◇ Repeated analyses:
 - ◇ Even small improvements in performance can save you hours

Indication of file sizes:

13 years of water temperature data, text format, 12 distinct depths

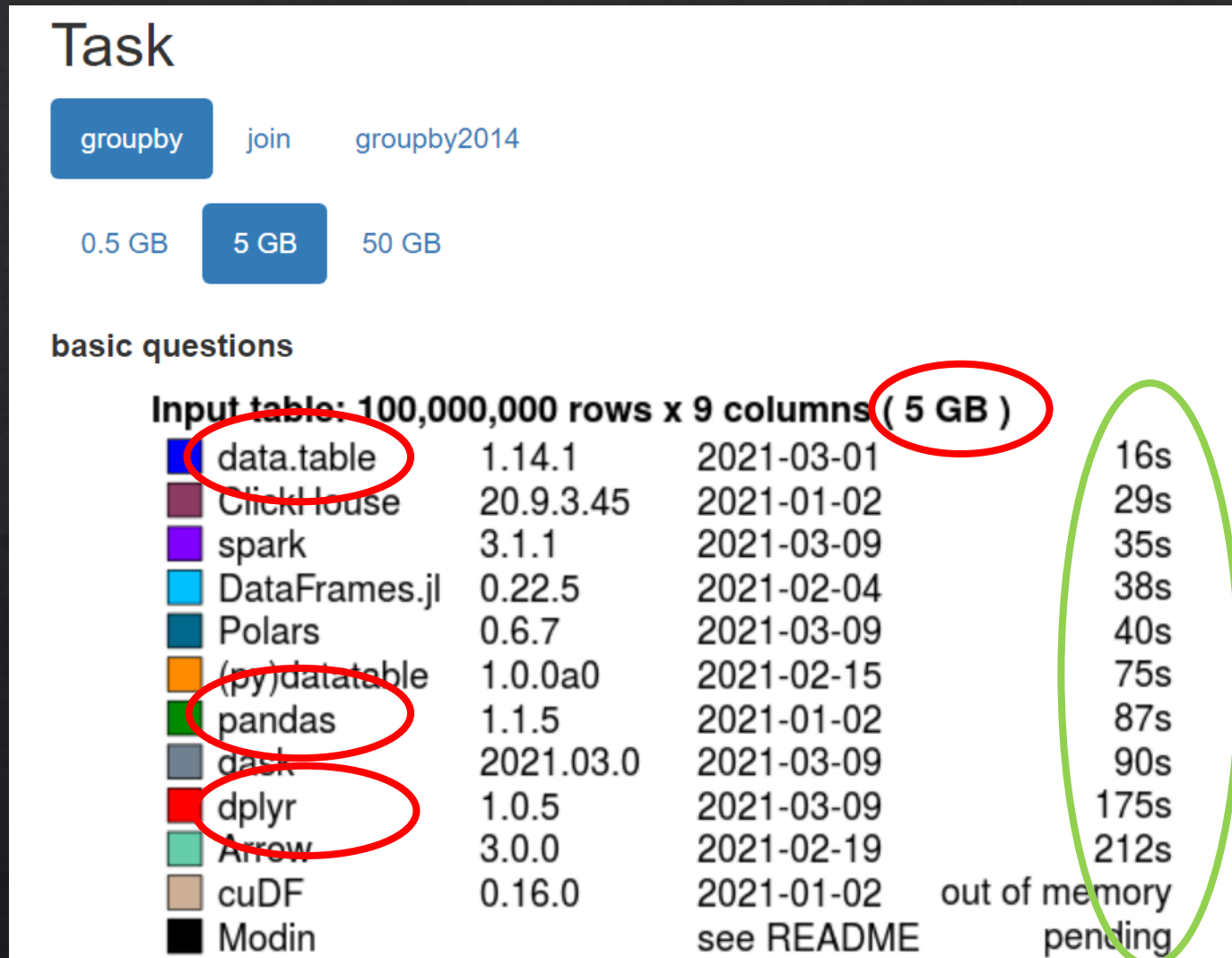
- Hourly: 11 MB
- 10 minutes: 85 MB
- 2 minutes: 387 MB

Performance of data.table



Source: Jan Gorecki, 2019

Performance of data.table



Data.table can be multiple factors faster than alternatives, even those in other languages!

Performance of data.table

groupby

join

groupby2014













0.5 GB

5 GB

50 GB

basic questions

Input table: 10,000,000 rows x 7 columns (0.4 GB)

	cuDF	0.16.0	2021-01-02	2s
	Polars	0.6.7	2021-03-09	6s
	data.table	1.14.1	2021-03-01	8s
	ClickHouse	20.9.3.45	2021-01-02	22s
	dplyr	1.0.5	2021-03-09	29s
	spark	3.1.1	2021-03-09	41s
	pandas	1.1.5	2021-01-02	56s
	DataFrames.jl	0.22.5	2021-02-04	129s
	(py)datatable	1.0.0a0	2021-02-15	195s
	dask	2021.03.0	2021-03-09	938s
	Arrow	3.0.0	2021-02-19	not yet implemented
	Modin		see README	pending

<https://h2oai.github.io/db-benchmark/>, 2021-03-15

Performance of data.table

groupby

join

groupby2014













0.5 GB

5 GB

50 GB

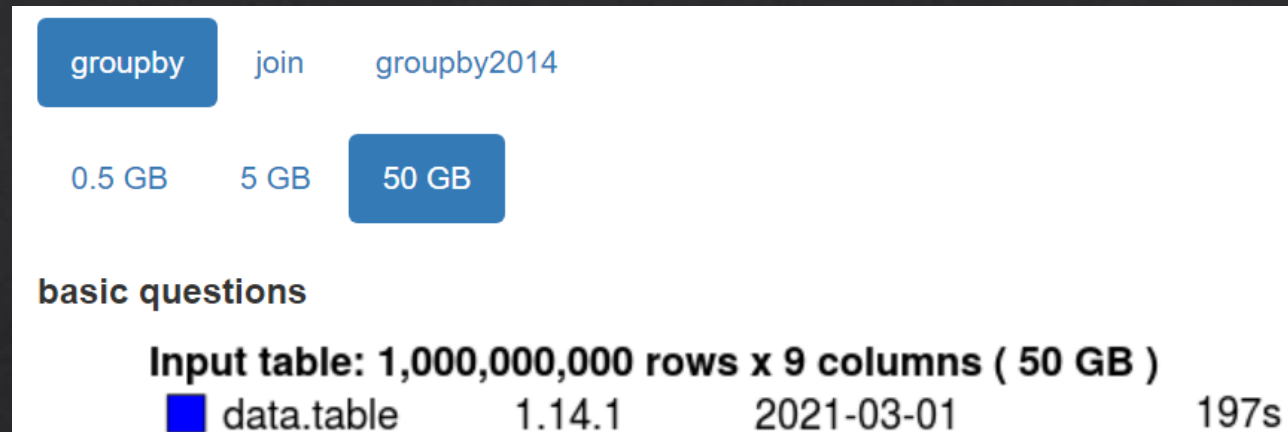
basic questions

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







	data.table	1.14.1	2021-03-01	197s
	Polars	0.6.7	2021-03-09	311s
	ClickHouse	20.9.3.45	2021-01-02	332s
	spark	3.1.1	2021-03-09	537s
	DataFrames.jl	0.22.5	2021-02-04	708s
	(py)datatable	1.0.0a0	2021-02-15	727s
	dplyr	1.0.5	2021-03-09	internal error
	pandas	1.1.5	2021-01-02	out of memory
	dask	2021.03.0	2021-03-09	out of memory
	cuDF	0.16.0	2021-01-02	out of memory
	Arrow	3.0.0	2021-02-19	internal error
	Modin		see README	pending

<https://h2oai.github.io/db-benchmark/>, 2021-03-15

Performance of data.table



for data.frame-like objects, R is not slow – if you're using data.table!

	dplyr	1.0.5	2021-03-09	internal error
	pandas	1.1.5	2021-01-02	out of memory
	dask	2021.03.0	2021-03-09	out of memory
	cuDF	0.16.0	2021-01-02	out of memory
	Arrow	3.0.0	2021-02-19	internal error
	Modin		see README	pending

<https://h2oai.github.io/db-benchmark/>, 2021-03-15

What makes data.table so fast?

- ◆ Speed has been the main focus of data.table during its whole development
 - ◆ Efficient algorithms (C code)
 - ◆ Internal structure of data.tables
 - ◆ Less in-memory copying
 - ◆ Low-level multi-threading
-
- ◆ When using data.table, most of this will happen internally, so you can code “normally” and still achieve a high efficiency

Not just fast, a clear syntax as well!

- ◇ Data.table has a syntax that is different from base R
- ◇ “Come for the performance, stay for the syntax” – Elio Campitelli, R blogger
- ◇ Concise
- ◇ Consistent
- ◇ DT[i, j, by]
 - ◇ i: what rows
 - ◇ j: what to do
 - ◇ by: by what groups
- ◇ More on this in the walkthrough...

Not just fast, a clear syntax as well!

- ◆ Tidyverse also offers an alternative syntax to base R
 - ◆ Verbose
 - ◆ Verbs as functions
 - ◆ Pipes
- ◆ Data.table
 - ◆ More alike query languages such as SQL
 - ◆ Closer to base-R
- ◆ Which one is “better” ultimately comes down to preference, but to get optimal performance, you’ll have to use at least some of the data.table syntax
- ◆ dtplyr package (<https://cran.r-project.org/web/packages/dtplyr/index.html>)

Short comparison syntax of data.table and tidyverse

Tidyverse

```
DF %>%  
  group_by(z) %>%  
  summarise(sum(y))  
ans <- DF %>%  
  group_by(z) %>%  
  mutate(y = cumsum(y))
```

```
DF %>%  
  filter(x>2) %>%  
  group_by(z) %>%  
  summarise(sum(y))  
ans <- DF %>%  
  group_by(z) %>%  
  mutate(y = replace(y,  
                      which(x > 2),  
                      cumsum(y)))
```

data.table

```
DT[, sum(y), by = z]
```

```
DT[, y := cumsum(y), by = z]
```

```
DT[x > 2, sum(y), by = z]
```

```
DT[x > 2, y := cumsum(y), by = z]
```

Short comparison syntax of data.table and tidyverse

Tidyverse

```
diamondsDF %>%  
  filter(cut != "Fair") %>%  
  group_by(cut) %>%  
  summarize(  
    AvgPrice = mean(price),  
    MedianPrice = as.numeric(median(price)),  
    Count = n()  
  ) %>%  
  arrange(desc(Count))
```

data.table

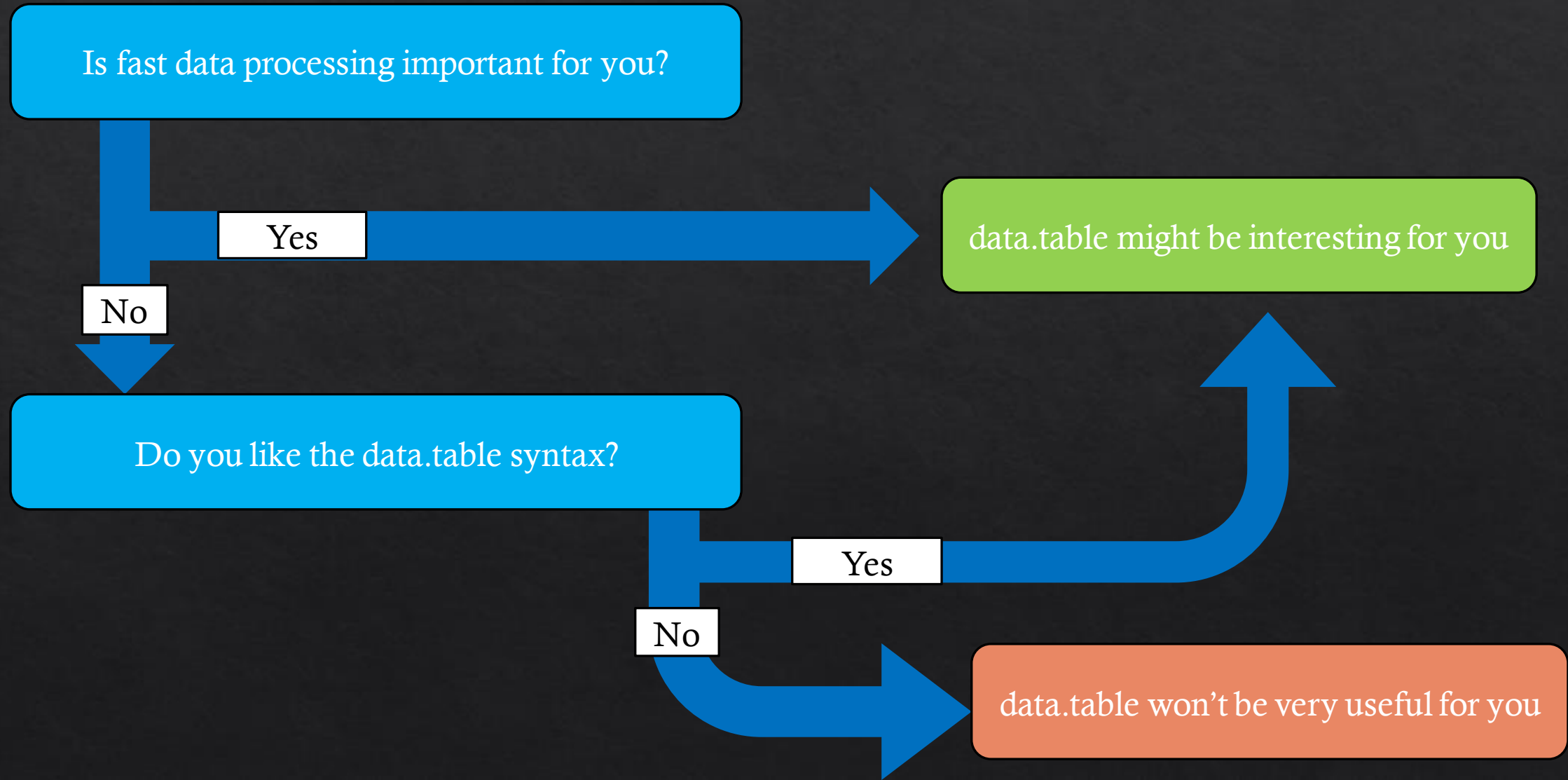
```
diamondsDT[  
  cut != "Fair",  
  .(AvgPrice = mean(price),  
    MedianPrice = as.numeric(median(price)),  
    Count = .N  
  ),  
  by = cut  
][  
  order(-Count)  
]
```

But combinations are possible!

data.table with pipes

```
library(magrittr)
library(ggplot2)
temperature %>%
  .[level == 1000] %>%
  .[, mean(air), by = .(lat, lon)] %>%
  .[lat > 0] %>%
  ggplot(aes(lon, lat)) +
  geom_raster(aes(fill = V1), interpolate = TRUE)
```

Should you use data.table?



In summary

- ◇ data.table is very fast on especially large datasets
 - ◇ Significantly faster than many alternatives
 - ◇ File size needs to be large to notice difference
- ◇ Consistent and concise syntax
 - ◇ We'll learn how to work with this during the walkthrough
- ◇ To the hands-on part!



Outline

- ◆ What is data.table (extension to data.frame) Package creator Matt Dowle. Website link. Already in R for a long time.
- ◆ Why this presentation: I like data.table a lot, helped me write better code, not advertised that much. But especially the speed argument may be very relevant for GLEON!
- ◆ First presentation (15 min.), then walkthrough R script, then questions/exercises
- ◆ Speed! Internal optimisation, update by reference. R still has the reputation of being a slow language, but actually...
- ◆ Show benchmark website (faster than base R, Python, tidyverse...) Now the speed argument is important, but realise that you need to have data >100MB to actually notice anything, or many repeated operations.
- ◆ Syntax (come for speed, stay for syntax) (quick comparison with tidyverse; concise and consistent vs. Verbose and accessible to non-coders) -> Comes down to preference

Outline

- ◆ Syntax is consistent, as it is always `i, j, by` (some additional arguments possible)
- ◆ Some like it, some prefer base R or tidyverse, and in that case, the syntax is a burden when you would like to use the fast computation of `data.table`. I hope that my walkthrough would in that case help (also consider `dtplyr` if you like the tidyverse syntax; not as fast as `data.table`, but it comes a long way)
- ◆ Additional upsides: stable, rigorous testing, large community, updates
- ◆ I like it a lot, but this is not a sales pitch; essentially, if you need the speed, check it out. If you like the syntax, also check it out.
- ◆ I hope that my walkthrough script can simultaneously act as a “cheat sheet” when using `data.table` for the first time.

