Tyson S. Barrett, PhD | JSM 2024

Efficient Tools for Your Tidy Workflow

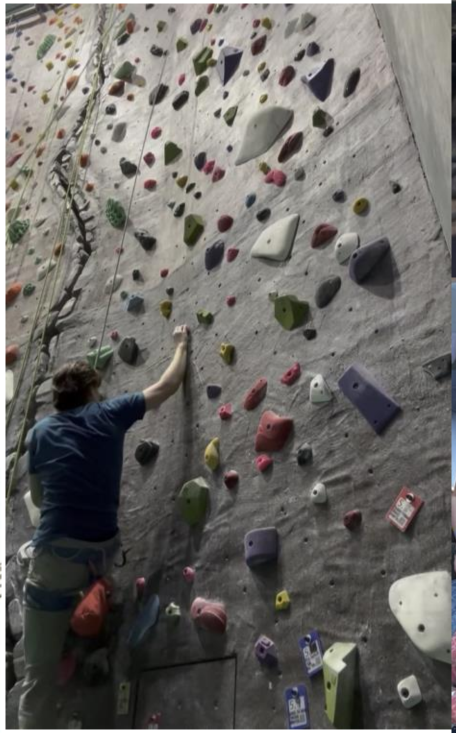
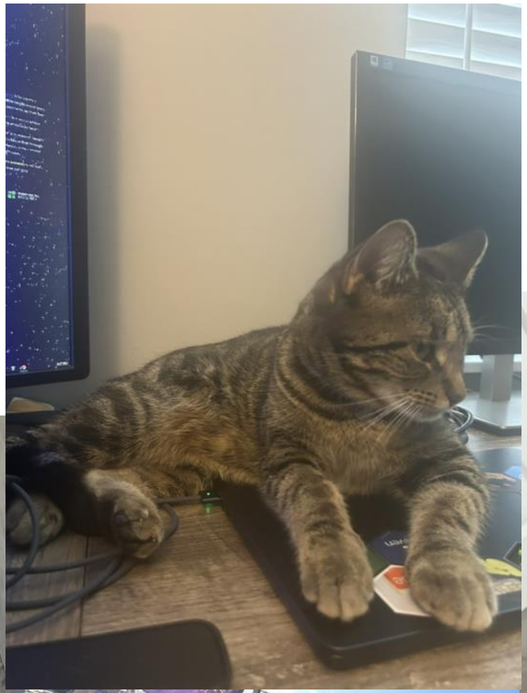
A case for incorporating `data.table`

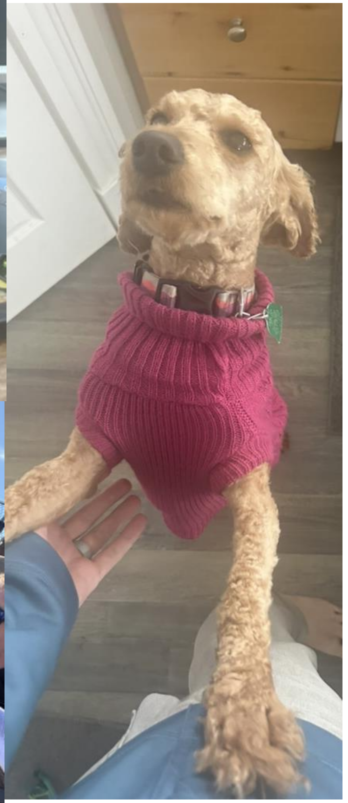
BUT FIRST, WHO AM I?

Current maintainer of
`data.table`

Research Manager at a large US Health Insurer

Human nerd



1990

BEGINNING
Data Frames have their beginning in S

2000

FIRST R RELEASE
R adopted Data Frames from S

FIRST DATA.TABLE RELEASE
Extended Data Frames with concise syntax and fast operations

2009

FIRST PANDAS RELEASE
Brought Data Frames from R to Python

2014

TIDYVERSE AND FRIENDS
dplyr, tidyr, and the rest of the tidyverse comes into being, adding a structured grammar to working with Data Frames

2020

PANDEMIC
Have to note something as big as the pandemic

2024

TODAY

https://towardsdatascience.com/preventing-the-death-of-the-dataframe-8bca1c0f83c8

WHAT IS A DATA FRAME?

“A **data frame** is a **list** of variables of the same number of rows with unique row names… [it’s] a **matrix-like** structure whose columns may be of differing types (numeric, logical, factor and character and so on).“

R’s documentation for `data.frame()`

WHAT IS A DATA FRAME?

"A **data frame** is a **list** of variables of the same number of rows with unique row names… [it's] a **matrix-like** structure whose columns may be of differing types (numeric, logical, factor and character and so on)."

R's documentation for `data.frame()`

WHAT IS A DATA FRAME?

Do stuff to rows

```
df[i, j]
```

Do stuff to columns

data.table

Do stuff to rows

Group by stuff

```
dt[i, j, by]
```

And more stuff

Do stuff to columns

WHY data.table?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle management

Community

Feature rich

WHY data.table?

```
dt[i, j, by]
```

Concise syntax

[illegible]

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

Memory efficient

```
dt[dt2, on = “id”]
```

Careful API lifecycle management

```
dt[dt2, on = “id”, roll = TRUE]
```

Community

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

Feature rich

WHY data.table?

Concise syntax

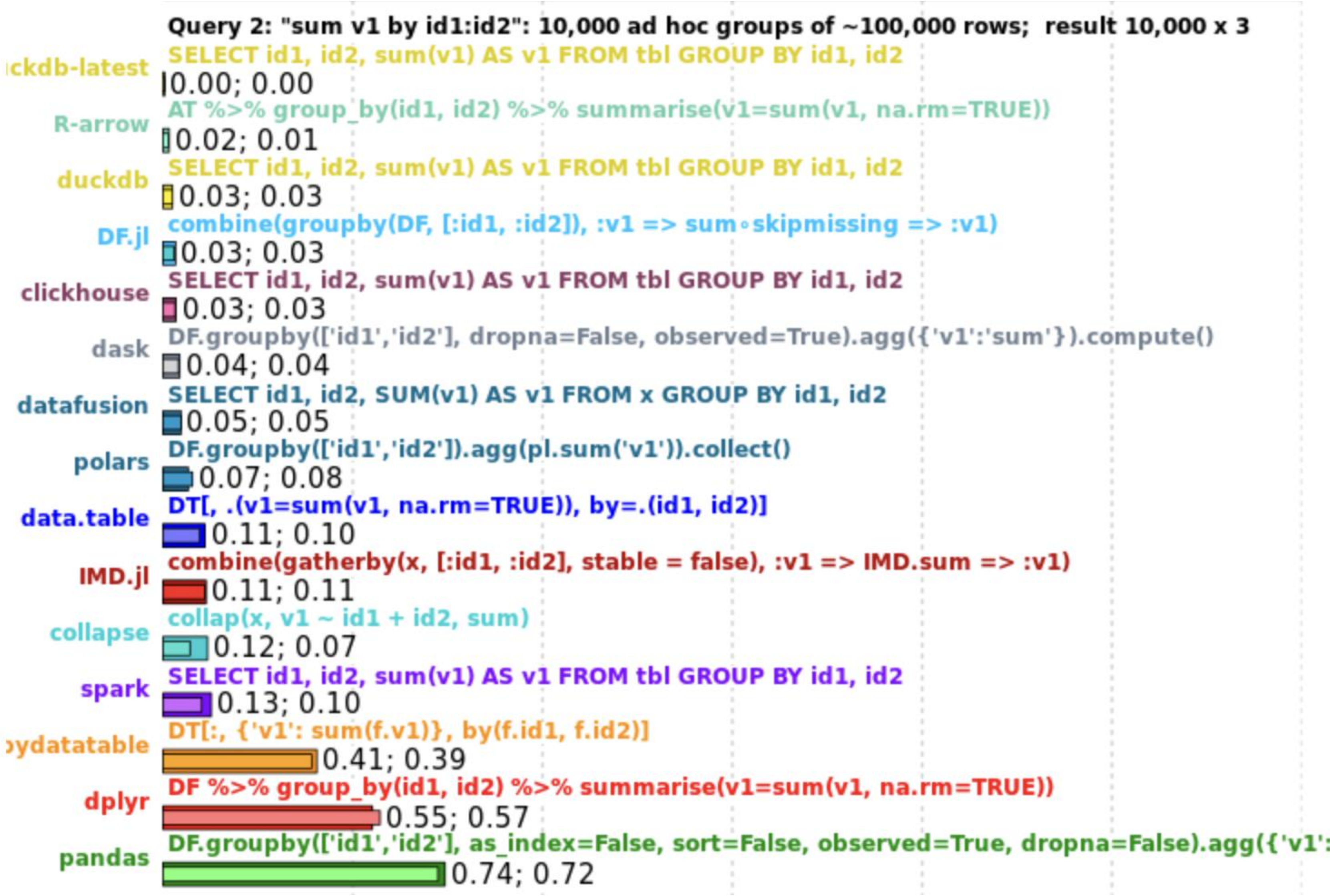
Fast speed

Memory efficient

Careful API lifecycle

Community

Feature rich



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

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

WHY `data.table`?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle management

Community

Feature rich

Still works for R version 3.3.0!

WHY

Concise syn

Fast speed

Memory effi

Careful API lifecycle management

Community

Feature rich

Still works for R version 3.3.0!

WHY SAY ALL THESE WORDS TO US STATISTICIANS?

`data.table` retains the benefits of matrices and gives you the flexibility of spreadsheets

This is ideal for statistical computing

NEW TOOLS AND `data.table`

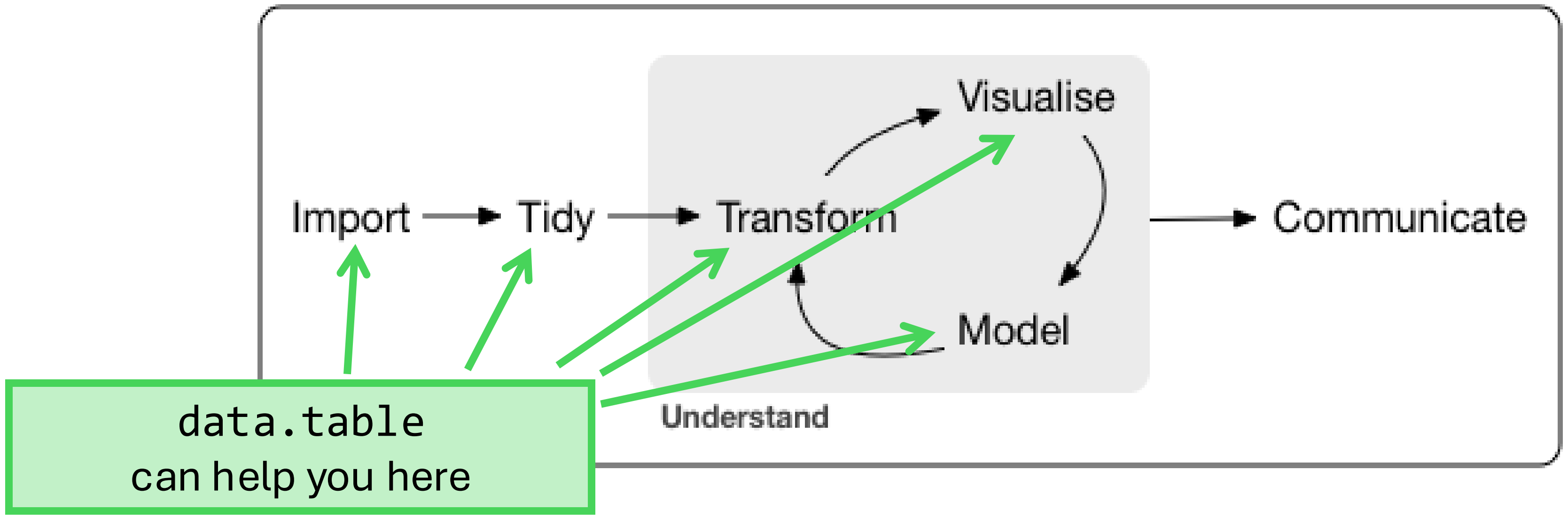
Incredible new tools (e.g., arrow, duckDB, polars) are becoming increasingly accessible, is that the death of `data.table`?

No! Use these powerful tools together

Open-Source Love

INJECT data.table INTO YOUR WORKFLOW

From "R for Data Science"



```
# examples are using
dt = palmerpenguins::penguins_raw
dt = janitor::clean_names(dt)
setDT(dt)
```

IMPORT

From “R for Data Science”

data.table
can help you here

```
dt = fread("path")
```

TIDY

From “R for Data Science”

Import
Tidy

data.table
can help you here

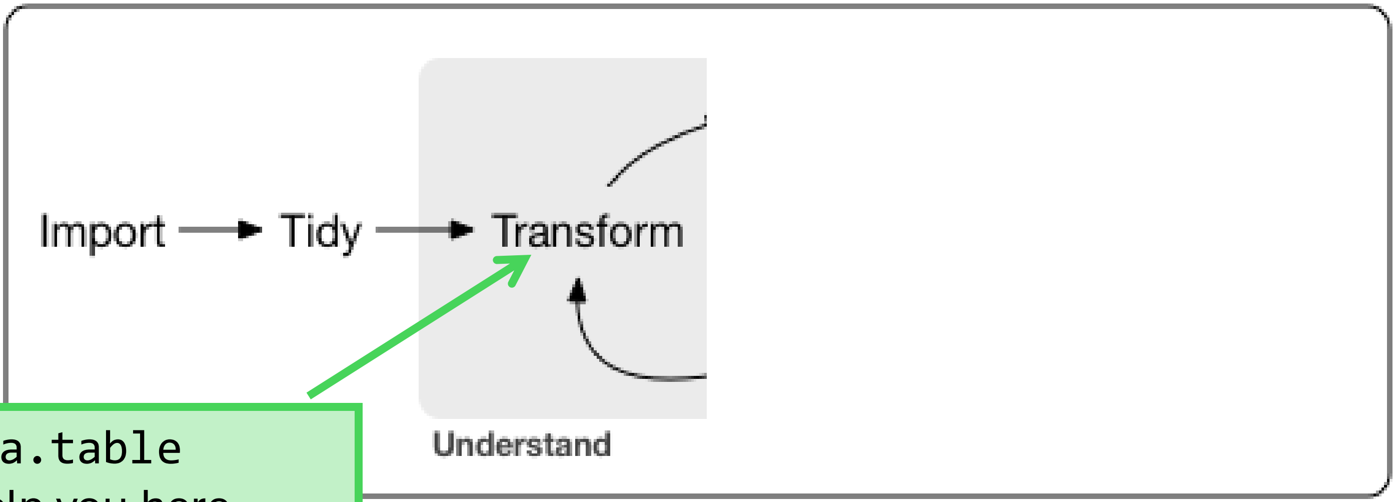
```
dt2 = melt(
  dt,
  id.vars = c("study_name", "sample_number", "species"),
  measure.vars = c("culmen_length_mm", "culmen_depth_mm")
)

dt3 = dcase(
  dt2,
  study_name + sample_number + species ~ variable,
  value.var = "value"
)
```

Pivot longer

Pivot wider

TRANSFORM

From "R for Data Science"

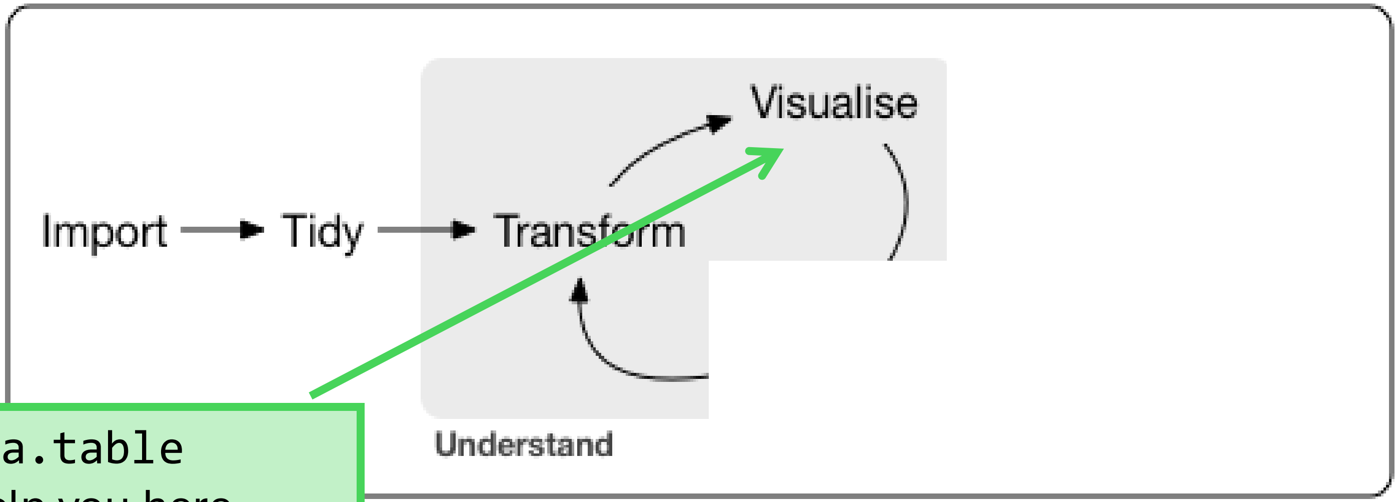


data.table
can help you here

```
# transform a single variable
dt[, species := tolower(species)]
dt[, species := stringr::str_remove(species, "penguin.*$")]

# transform multiple columns at the same time
dt[, (names(.SD)) := lapply(.SD, fun), by = species, .SDcols = culmen_length_mm:body_mass_g]

# join with another data set
dt[penguin, on = "species")]

# rolling join
dt[penguin, on = "egg_date", roll = "nearest"]
```

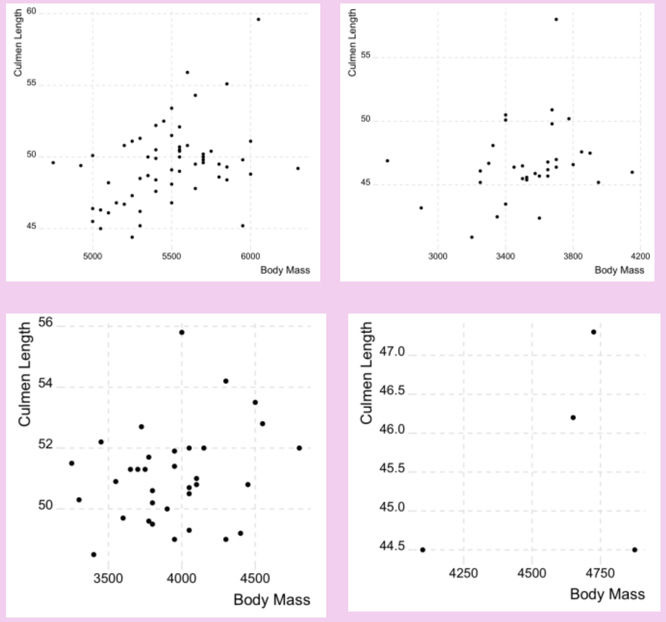
VISUALIZE

From "R for Data Science"



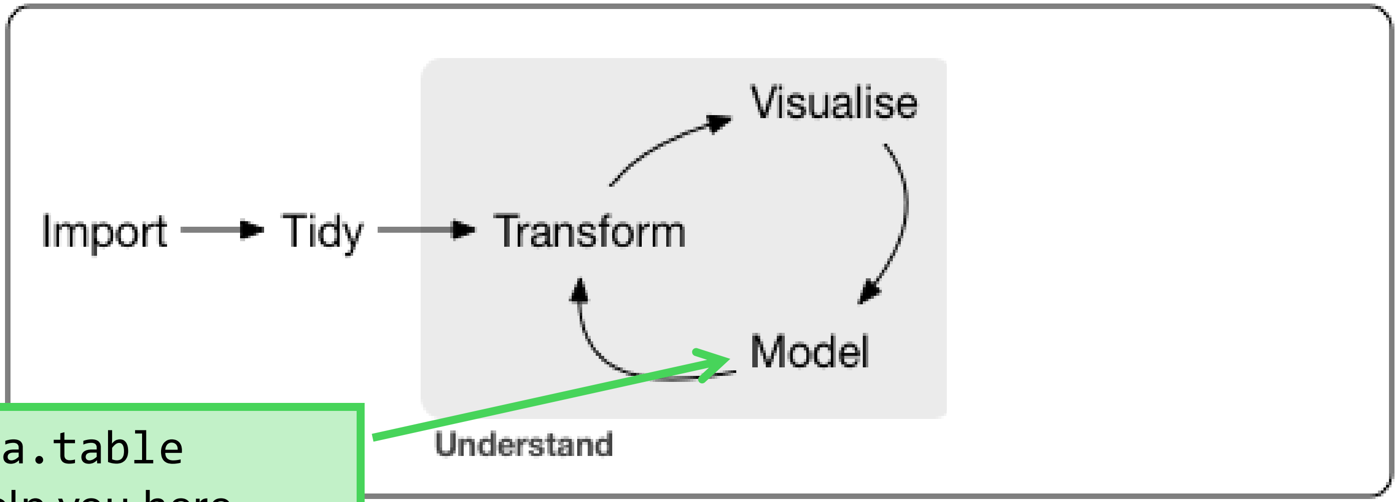
data.table
can help you here

```
# make lots of ggplots by grouping variable(s)
dt[,
  print(
    ggplot(.SD, aes(x = body_mass_g, y = culmen_length_mm)) +
    geom_point() +
    ggridges::theme_ridges() +
    theme(panel.grid = element_line(linetype = "dashed")) +
    labs(x = "Body Mass", y = "Culmen Length")
  ),
  by = .(species, sex)
]
```



MODEL

From "R for Data Science"



data.table
can help you here

```
# run many models by a grouping variable
dt[, .(mods = list(
    lm(culmen_length_mm ~ body_mass_g, data = .SD)
  )),
  by = species
]
#        species    mods
#         <char>  <list>
# 1:     adelie  <lm[13]>
# 2:     gentoo  <lm[13]>
# 3: chinstrap  <lm[12]>
```

IT TAKES A VILLAGE

I often use parquet files and turn set the data frame as a DT

```
dt = arrow::read_parquet("path")    # palmerpenguins::penguins_raw
setDT(dt)
```

Several helper packages make a number of other facets of a tidy workflow easier

```
janitor::clean_names(dt)  # clean those horrible names
janitor::remove_empty(dt) # drop empty columns and rows (happens in excel files a lot)
tidyfast::dt_fill()       # data.table implementation of tidyr::fill()
stringr::str_replace(dt, ...)   # replace strings (and a bunch of other nice string functions)
forcats::fct_inorder(fct_var)   # get factors into good form for visuals
...
```

Quarto and friends for easy output

Make your cleaned data beautiful

```
library(gtsummary)  # and officer and friends
library(ggplot2) # and friends
```

Tyson S. Barrett

Thanks to Matt Dowle, Arun Srinivasan, Michael Cherico, Jan Gorecki, Toby Hocking and the data.table team!

t.barrett88@gmail.com

tysonbarrett.com

@healthandstats

github.com/tysonstanley

Slides will be on my website to download