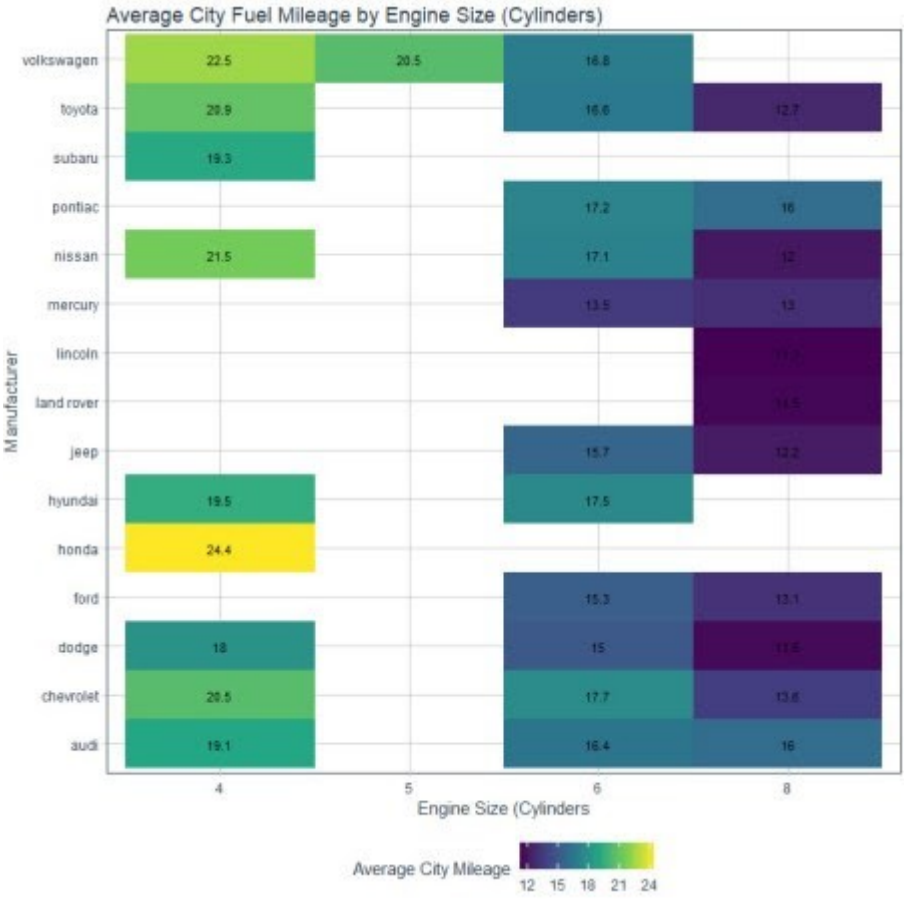
The data.table backend to dplyr

There's a new R package in town. It's called `dtplyr`. It's the `data.table` backend to `dplyr`. And, what it get's you is truly amazing:

- Enjoy the 3X to 5X `data.table` speedup with grouped summarizations
- **All from the comfort of** `dplyr`

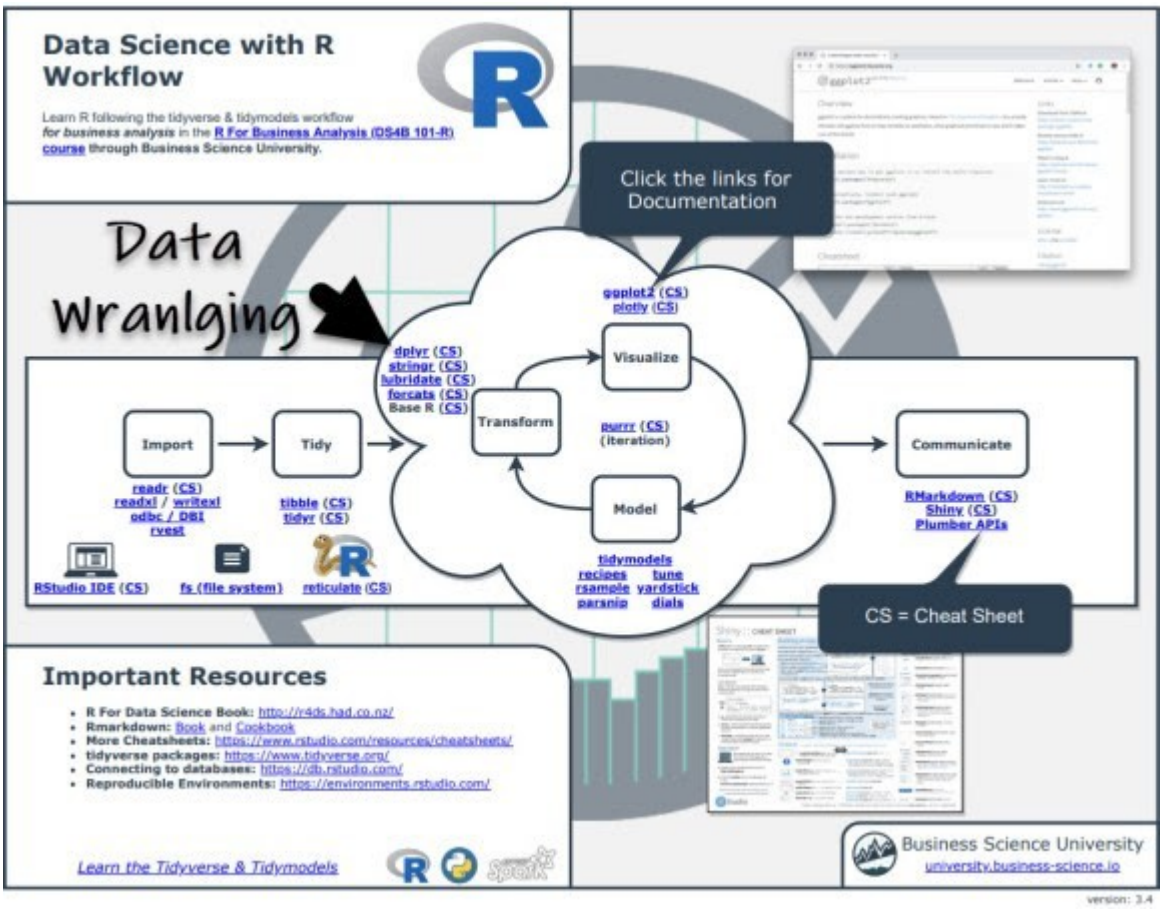


Make insanely fast grouped summaries by leveraging data.table with dtplyr then quickly visualize your summaries with ggplot2.

Before we get started, get the Cheat Sheet

The most powerful tool in my arsenal is NOT my knowledge of the key R packages, but it's **knowing where to find R packages and documentation.**

The Ultimate R Cheat Sheet consolidates the documentation on every package I use frequently (including dplyr, data.table, and dtplyr).



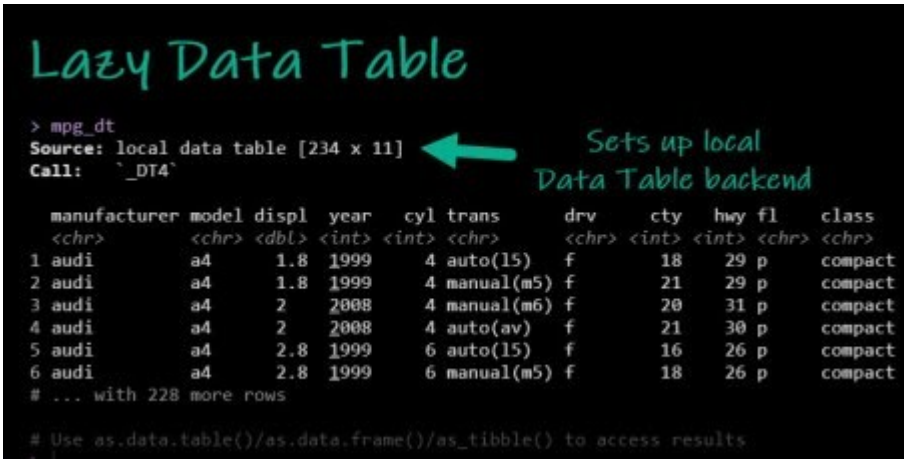
If you tab through to page 3, you'll see a section called "Speed and Scale". You can quickly see options to help including data.table, dtplyr, furrr, sparkly, and disk.frame. Enjoy.

Get started with dtplyr

The first thing you'll want to do is **set up a Lazy Data Table** usng the `lazy_dt()` function.

```
12 # 1.0 DATA TABLE ----
13
14 # * Make a Lazy Data Table ----
15 mpg_dt <- lazy_dt(mpg)
16 mpg_dt
17
```

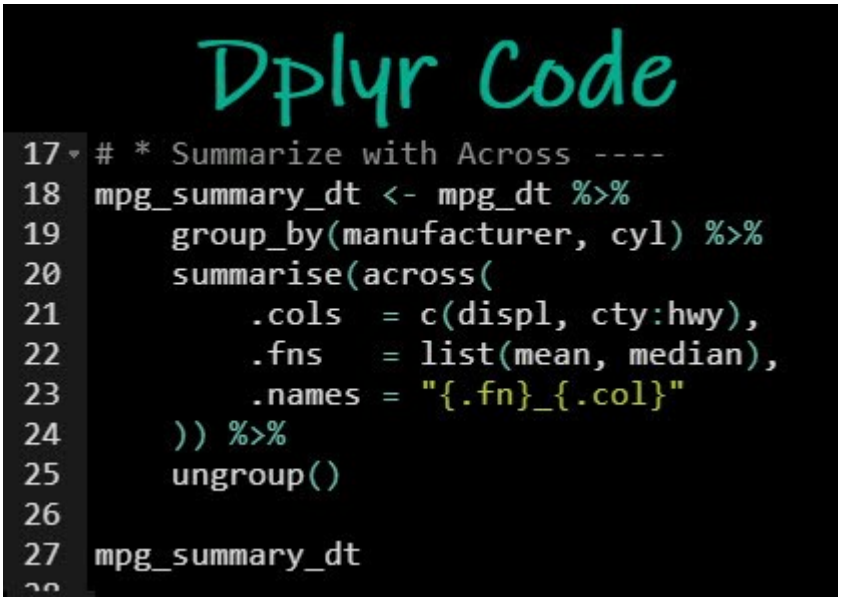
So what happened? We now have a pointer to a `data.table`. This is a special connection that we can use to write `dplyr` code that gets converted to `data.table` code.



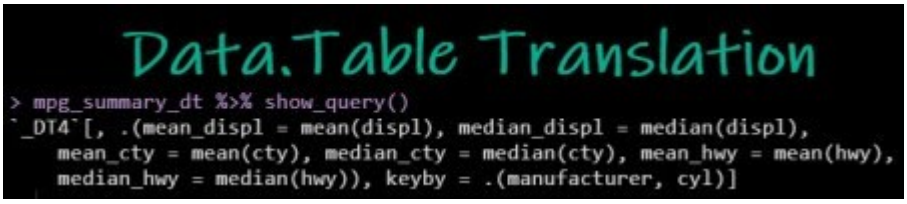
Translating dplyr to data.table

This idea of a data.table backend to dplyr is insanely powerful. Here's an example of a dplyr grouped-summarization that gets translated to data.table for a speedup.

- Start with lazy datatable connection object
- **Group by** the manufacturer and cylinder columns
- **Summarize** with the new `dplyr::across()` function
- **Ungroup** the lazy data.table

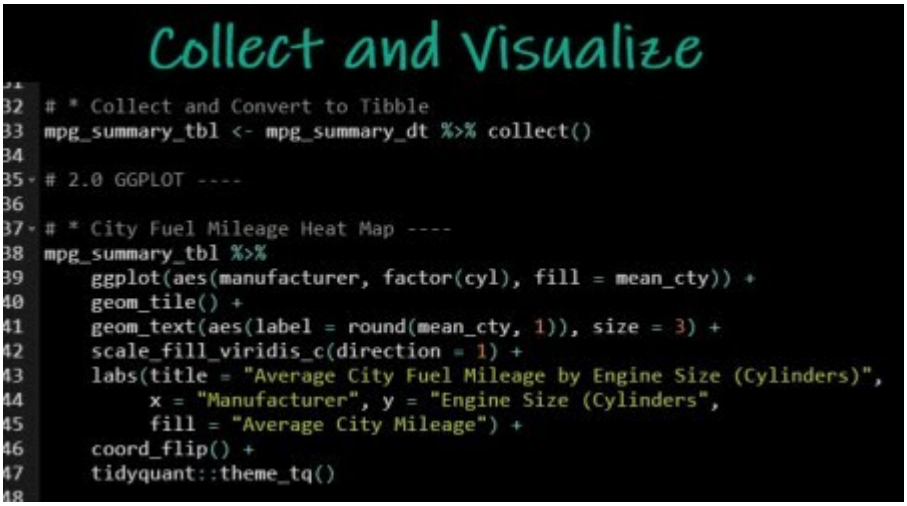


The dtplyr backend does the heavy-lifting, converting your dplyr code into data.table code.



When your done wrangling…
Just collect and visualize

Use the `collect()` function or `as_tibble()` function to apply the data.table translation to your lazy data table and extract the results.



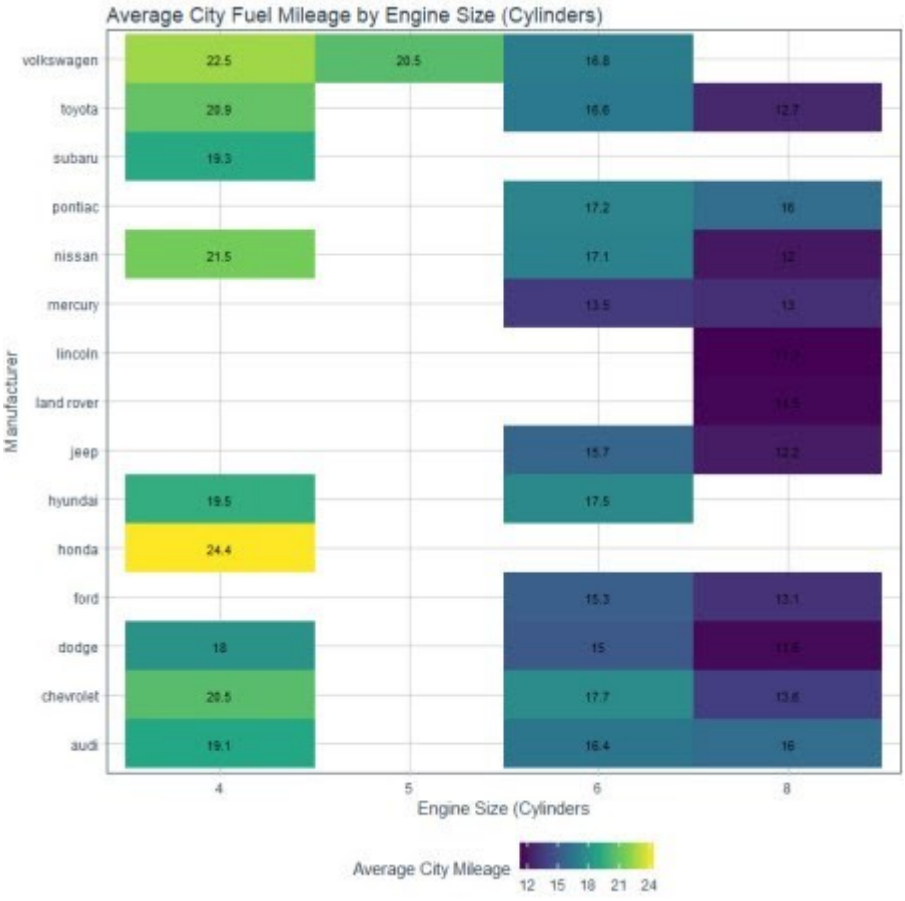

```
# * Collect and Convert to Tibble
mpg_summary_tbl <- mpg_summary_dt %>% collect()

# 2.0 GGPLOT ----

# * City Fuel Mileage Heat Map ----
mpg_summary_tbl %>%
    ggplot(aes(manufacturer, factor(cyl), fill = mean_cty)) +
    geom_tile() +
    geom_text(aes(label = round(mean_cty, 1)), size = 3) +
    scale_fill_viridis_c(direction = 1) +
    labs(title = "Average City Fuel Mileage by Engine Size (Cylinders)",
         x = "Manufacturer", y = "Engine Size (Cylinders",
         fill = "Average City Mileage") +
    coord_flip() +
    tidyquant::theme_tq()
```

The ggplot2 code produces this visualization. We can easily see:

- Honda has the highest City Mileage in small engine cars (24.4 MPG)
- Audi has the highest City Mileage in large engine cars (16 MPG)



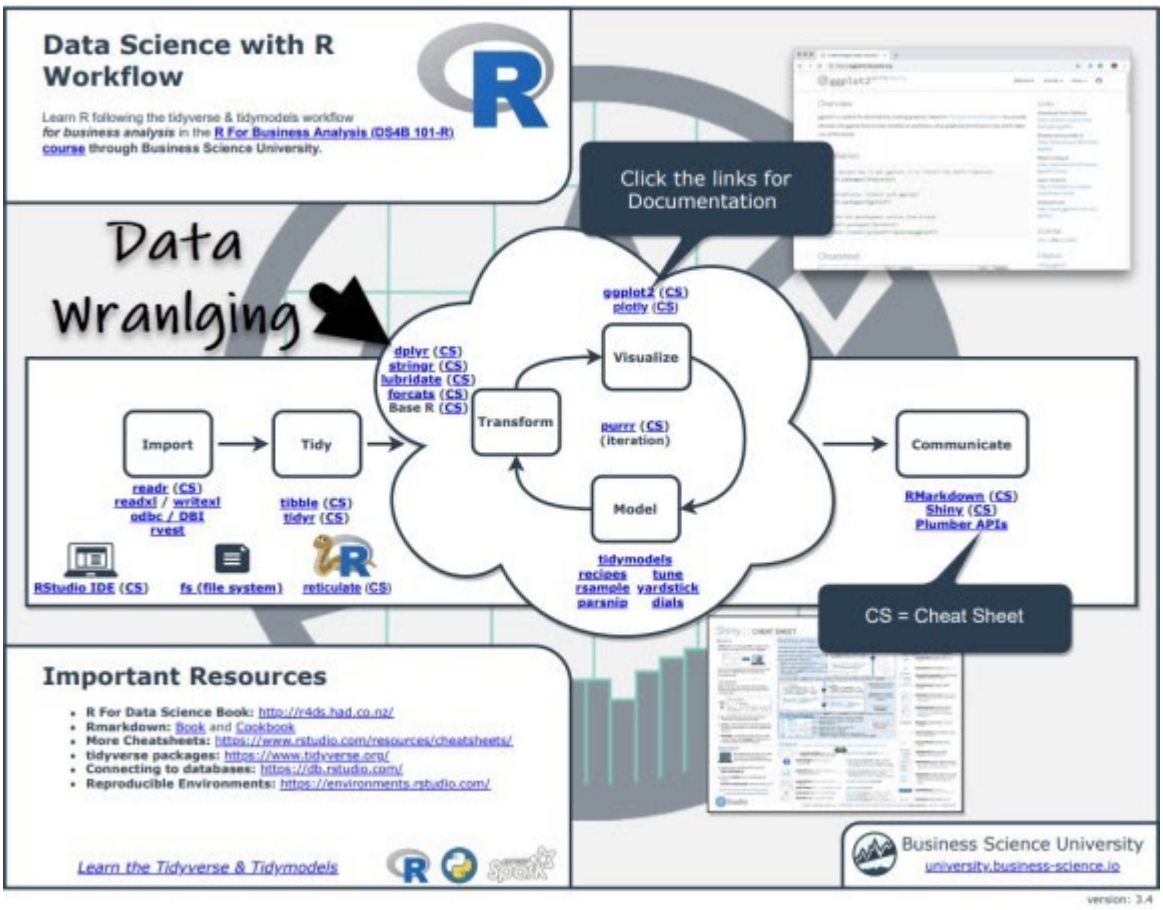
Learning Data Wrangling with Dplyr

It should be obvious now that **learning dplyr is insanely powerful**. Not only is it beginner-friendly, it unlocks data.table, the fastest in-memory data wrangling tool. Here are a few tips.

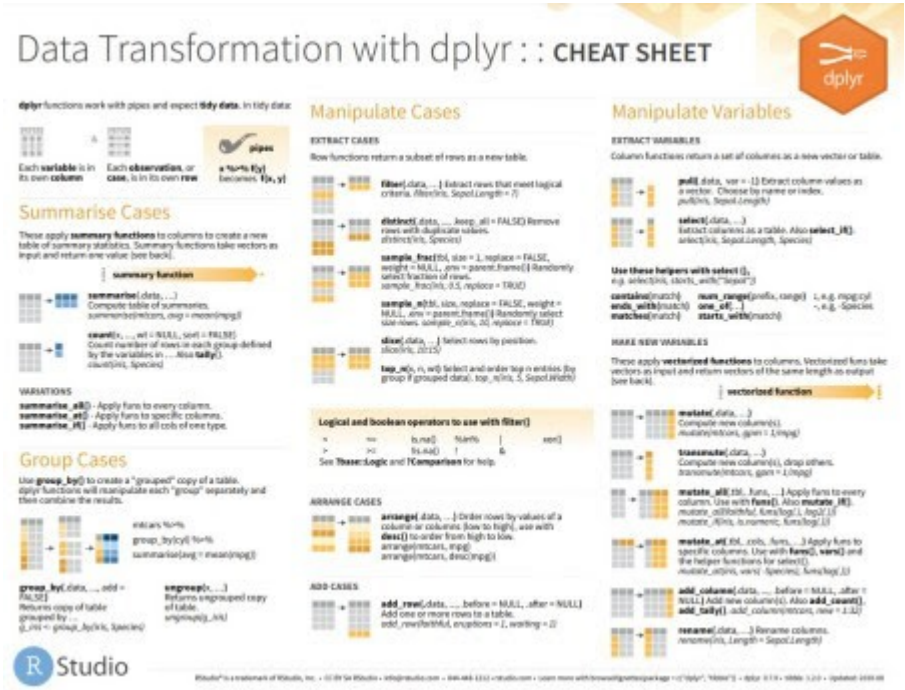
Pro Tip 1 – Use the Cheat Sheet

Dplyr is an 80/20 tool shown on the first page of my Ultimate R Cheat Sheet.

Click the "CS" next to dplyr to get the Data Wrangling with Dplyr Cheat Sheet. Woohoo!



Clicking the "CS" opens the Data Transformation with Dplyr Cheat Sheet. Now you're ready to begin learning Dplyr.



PRO TIP 2 – Learn Dplyr in my Business Analysis with R Course

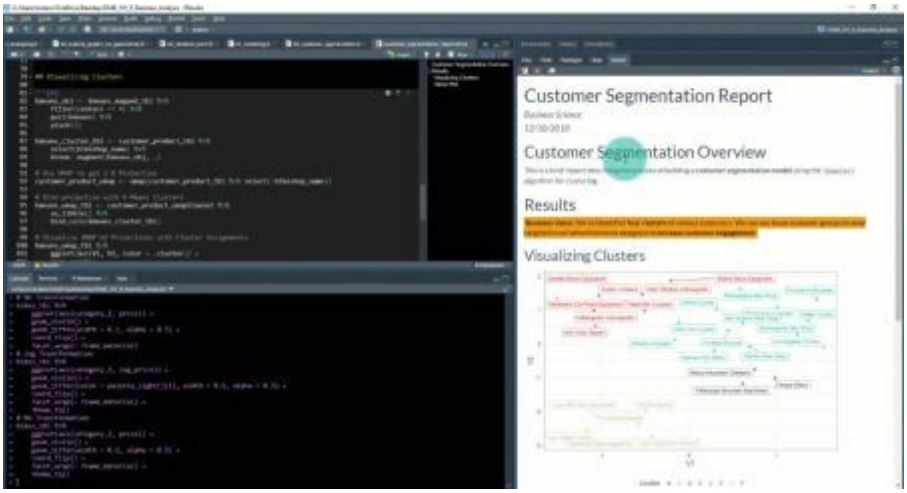
It might be difficult to learn Dplyr on your own. I have a course that walks you through the entire process from analysis to reporting.

The R for Business Analysis 101 Course is the first course in my R-Track program . You'll do a

ton of data transformations while you make a two reports:

1. Customer Segmentation Report
2. Product Pricing Estimation and Gap Analysis

Here's the **Customer Segmentation Report**.



In Summary

You just sliced and diced data with dtplyr – the data.table backend to dplyr.