Big data

Last modified: 2026-11-14 03:52:56 (PST)

The data.table package

It may also be the case that you're working with very large datasets. Generally I would define this as 10+ million rows. As is outlined in this document, the 3 main players in the data analysis space are Base R, `tidyverse` (more specifically, `dplyr`), and `data.table`. For a majority of things, Base R is inferior to both `dplyr` and `data.table`, with concise but less clear syntax and less speed. `dplyr` is architected for medium and smaller data, and while it's very fast for everyday usage, it trades off maximum performance for ease of use and syntax compared to `data.table`. An overview of the `dplyr` vs `data.table` debate can be found in this Stack Overflow post and all 3 answers are worth a read.

You can also get much of `data.table`'s speed while keeping `dplyr` syntax, which I find to be the best of both worlds. `{dtplyr}` provides a `data.table` backend for `dplyr`: wrap any data frame with `dtplyr::lazy_dt()`, and `dplyr` verbs are translated into `data.table` code. Running `dplyr` verbs on a plain `data.table` (for example, one made with `as.data.table()`) does not do this translation. A `lazy_dt()` pipeline is lazy: nothing runs until you collect the result with `as_tibble()`, `as.data.table()`, or `collect()`.

If you want to test whether using a certain coding approach increases speed, consider the `tictoc` package. Run `tic()` before a code chunk and `toc()` after to measure the amount of system time it takes to run the chunk. For example, you might use this to decide if you *really* need to switch a code chunk from `dplyr` to `data.table`.

Using downsampled data

In our studies with very large datasets, we save "downsampled" data that usually includes a 1% random sample stratified by any important variables, such as year or household id. This allows us to efficiently write and test our code without having to load in large, slow datasets that can cause RStudio to freeze. Be very careful to be sure which dataset you are working with and to label results output accordingly.

Optimal RStudio set up

Using the following settings will help ensure a smooth experience when working with big data. In RStudio, go to the "Tools" menu, then select "Global Options". Under "General":

Workspace

- **Uncheck** Restore RData into workspace at startup
- Save workspace to RData on exit – choose **never**

History

- **Uncheck** Always save history

Unfortunately RStudio often gets slow and/or freezes after hours working with big datasets. Sometimes it is much more efficient to just use Terminal / gitbash to run code and make updates in git.