

# Reading in Larger Datasets with `read.table`

With much larger datasets, doing the following things will make your life easier and will prevent R from choking.

- Read the help page for `read.table`, which contains many hints
- Make a rough calculation of the memory required to store your dataset. If the dataset is larger than the amount of RAM on your computer, you can probably stop right here.
- Set `comment.char = ""` if there are no commented lines in your file.

# Reading in Larger Datasets with read.table

- Use the `colClasses` argument. Specifying this option instead of using the default can make 'read.table' run MUCH faster, often twice as fast. In order to use this option, you have to know the class of each column in your data frame. If all of the columns are “numeric”, for example, then you can just set `colClasses = "numeric"`. A quick and dirty way to figure out the classes of each column is the following:

```
initial <- read.table("datatable.txt", nrows = 100)
classes <- sapply(initial, class)
tabAll <- read.table("datatable.txt",
                    colClasses = classes)
```

- Set `nrows`. This doesn't make R run faster but it helps with memory usage. A mild overestimate is okay. You can use the Unix tool `wc` to calculate the number of lines in a file.

# Know Thy System

In general, when using R with larger datasets, it's useful to know a few things about your system.

- How much memory is available?
- What other applications are in use?
- Are there other users logged into the same system?
- What operating system?
- Is the OS 32 or 64 bit?

# Calculating Memory Requirements

I have a data frame with 1,500,000 rows and 120 columns, all of which are numeric data. Roughly, how much memory is required to store this data frame?

$$1,500,000 \times 120 \times 8 \text{ bytes/numeric}$$

$$= 1440000000 \text{ bytes}$$

$$= 1440000000 / 2^{20} \text{ bytes/MB}$$

$$= 1,373.29 \text{ MB}$$

$$= 1.34 \text{ GB}$$