STATS 769
Large Data Solutions

Paul Murrell

The University of Auckland

September 23, 2019

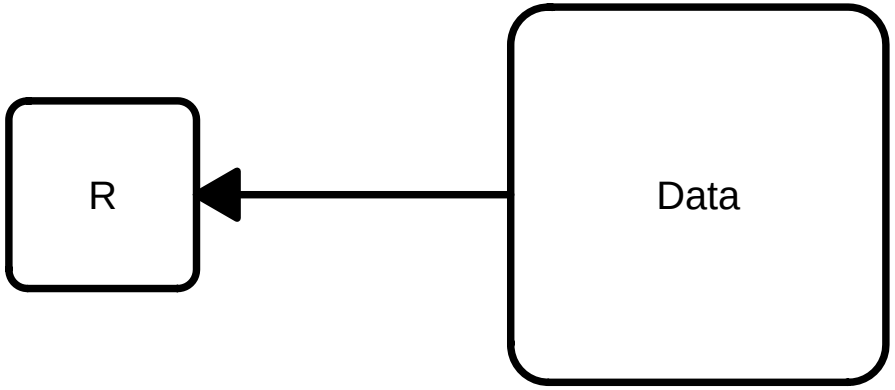
Overview

- This topic explores ways to deal with large data sets.
- "Large" means that a software tool that we know how to use cannot cope with the data set.

- We need to distinguish between data manipulation (Import, Tidy, Transform) and data analysis (Modelling).
- Often the solution to data manipulation is to use a different tool, but that is not as easily an option for data analysis.

Problem 1

- Our standard tool (R) cannot hold all of the data.



Trade Offs

Gaining greater efficiency (of storage) will usually come at the cost of at least one of the following ...

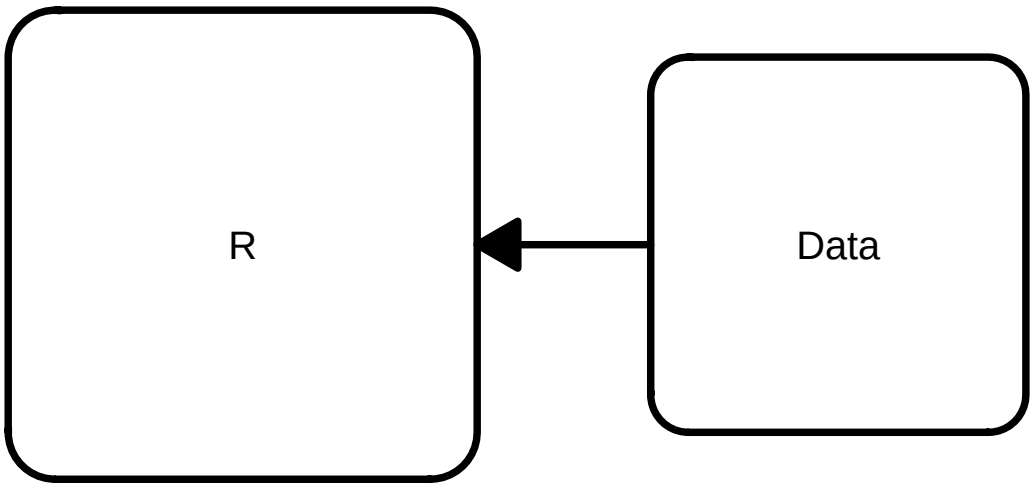
- Convenience: we will have to write more code and more complex code.
- Flexibility: we may be limited to a subset of data types and/or a subset of analysis techniques.
- Consistency: we may not be able to use the familiar R functions
  (we may have to learn a whole new set of functions or another program altogether).
- Cost: big pieces of hardware are expensive to buy
  (or even to rent).

Practical Tips

- ALWAYS develop code on a small subset of the full data.
- Test on a subset that you KNOW that answer for (e.g., generate fake data)
- Processing may take time so, if possible, print progress status, so you have some idea of where your algorithm is up to.

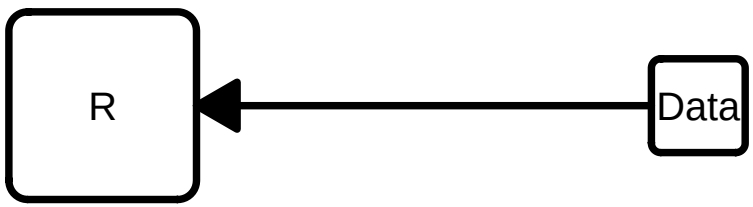
Solution 1

- Use a bigger machine (local cluster or the cloud).



Solution 2

- Make the data smaller



Solution 2

- Store the data in something else (e.g., query DBMS from R).
- Store the data in a more compact format.
- Sample the data.
- Subset the data. Break analysis into multiple analyses (e.g., one for males, one for females).
- Avoid R's copying semantics.
- "Stream" the data; only work on a bit of the data at a time.
- Use an R packages that leave most of the data on disk.

data.table

- The **data.table** package can be used to reduce the amount of copying that R does.
- **data.table** works with `data.table` objects rather than `data.frame` objects.
- `data.tables` behave a lot like `data.frames`.
- `data.table` subsetting behaves differently (and can use less memory).

- `dt[i, j, by]`
- `i` selects rows.
- `j` selects **and creates** columns.
- `by` defines grouping.

biglm

- The **biglm** package can be used to perform an analysis incrementally on chunks of data.

- `fit <- biglm(formula, someData)`
- `update(fit, moreData)`

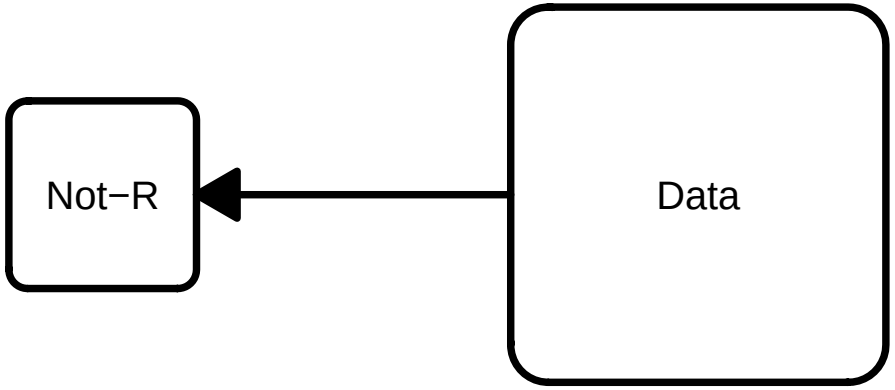
- `fit <- bigglm(formula, dataFunction)`

Leave data on disk

- The **RNetCDF** package can be used to load just part of a NetCDF file from disk.
- The **bigmemory** package can be used to create a "memory-mapped" file.
- The **bigtabulate** and **bianalytics** pacakges provide data manipulation and data analysis functions that work with **bigmemory** objects.
- There is also a package called **ff** that provides more control, but with more complexity.
- Another (complex) option is the "pbdR" project.

Solution 3

- Use a different tool (e.g., shell tools or DBMS).



Problem 2

- Our data is too big for our long term storage (not just RAM).
- This leads to distributed file systems (e.g. HDFS) and interfaces to distributed file system (e.g., Hadoop MapReduce and Apache Spark), which we will see in Week 11.

Resources

- **data.table** cheat sheet
  `https://s3.amazonaws.com/assets.datacamp.com/blog_assets/datatable_Cheat_Sheet_R.pdf`
- Introduction to **data.table**
  `http://cran.stat.auckland.ac.nz/web/packages/data.table/vignettes/datatable-intro.html`