A typical workflow for data analysis in R consists of the following steps: First load the raw data from file, then select and transform raw data into a form suitable for statistics, and then apply a statistical algorithm and visualization. However, the amount of data that can be analyzed using this process is limited by the amount of memory on the system on which R is run, which are typically desktop computers. A logical next step to mend this problem is to store the raw data in a relational database system. The standard process is now modified by not loading the raw data into R, but instead to load it into a database. Then, one can “outsource” the selection of data relevant to the analysis as well as basic calculations and aggregations to a highly optimized database system.

R’s database interface (DBI) provides a generic way of communicating with a relational database. Packages such as RPostgreSQL implement a specific driver for a particular database. However, not all relational databases are equally well suited to support statistical calculations. Transformation procedures and simple calculations make recommending a relational database optimized for “On-line analytical processing” (OLAP) rather obvious. Furthermore, R’s calculations on statistical observations are typically performed column-wise. Hence, only a fraction of columns are actually processed at a given time. These factors together suggest a column-oriented database design. MonetDB, an open-source column-oriented database system, implements this design. We have created the MonetDB.R package, which implements a native DBI driver to connect R with MonetDB.

However, in order to tell the database which data is to be transferred to R, a user still is required to write queries in the standardized Structured Query Language (SQL), which breaks work flows and increases training requirements. We went one step further and implemented a virtual data object. This monet.frame object is designed to behave like a regular R data.frame, but does not actually load data from MonetDB unless absolutely required. For example, consider the following interaction: mean(subset(mf,c1 > 42)$c2). We select a subset of the codemf object based on a filter condition on the c1 column. Then, we average of the c2 column. However, in this case the mf variable points to an instance of our virtual data object backed by a MonetDB table t1. Our implementation automatically generates and executes a SQL query: SELECT AVG(c2) FROM t1 WHERE (c1>42);. Instead of loading the potentially large table, we only transfer a single scalar value. Also, through the columnar storage layout of MonetDB, only the files that contain the data for columns c1 and c2 actually have to be accessed.

Our approach has two major advantages: Users are not exposed to SQL queries at all, and only data relevant to the analysis are loaded into R, which results in huge performance improvements. monet.frame is part of MonetDB.R, and we invite all those interested to take part in its evolution.

References