Adding large data support to R

Luke Tierney

Department of Statistics & Actuarial Science
University of Iowa

January 4, 2013

Introduction

- R is a language for data analysis and graphics.
- R is based on the S language developed by John Chambers and others at Bell Labs.
- R is widely used in the field of statistics and beyond, especially in university environments.
- R has become the primary framework for developing and making available new statistical methodology.
- Many (over 3,000) extension packages are available through CRAN or similar repositories.

Introduction

- R is an Open Source project.
- Originally developed by Robert Gentleman and Ross Ihaka in the early 1990's for a Macintosh computer lab at U. of Auckland, NZ.
- Since 1997 R is developed and maintained by the R-core group:

Douglas Bates	John Chambers	Peter Dalgaard
Robert Gentleman	Seth Falcon	Kurt Hornik
Stefano Iacus	Ross Ihaka	Friedrich Leisch
Uwe Ligges	Thomas Lumley	Martin Maechler
Duncan Murdoch	Paul Murrell	Martyn Plummer
Brian Ripley	Deepayan Sarkar	Duncan Temple Lang
Luke Tierney	Simon Urbanek	

- The 20 members are located in 11 different countires.

Introduction

- R has strong community support through
  - contributed extension packages
  - mailing lists and blogs
  - contributed documentation and task views
- There are also several commercial entities supporting R and R extensions.

R Language and Implementation

- R is designed for interactive data exploration.
- Interaction is through a read-eval-print loop.
- All computations are specified in the R language.
  - Even for simple tasks you need to know a little of the languege.
  - After learning to do simple tasks you know some of the language.
- The language is used to
  - prepare data for analysis
  - specify individual analyses
  - program repeated or similar analyses
  - program new methods of analysis

R Language and Implementation

- The R language operates on vectors and arrays.
- Basic arithmetic operations and mathematical and statistical functions are vectorized.
- Code for special functions and linear algebra computations
  - is written in C or Fortran
  - is based on publicly available libraries and interfaces
- Most analysis method implementations assume the date used will be in memory.
- Most return result structures that facilitate further analysis of fit models.

R Language and Implementation

- The R integer data type is equivalent to C int.
- This is now essentially universally a signed 32-bit type.
- This type is also used for the length of a vector or total size of an array.
- This design decision made sense when R started out nearly 20 years ago:
  - most machines and operating systems were 32-bit
  - this matched the interface provided by external C/FORTRAN code

R Language and Implementation

- This design limits the number of elements in an array to $2^{31} - 1 = 2,147,483,647$.
- For numeric (double precision) data this means the largest possible vector is about 16 GB.
- This is not yet a major limitation for typical users.
- It is a limitation for some users and will become more limiting over time.

Large Vector Support

- *Big Data* is a hot topic these days.
- Some categories:
  - fit into memory
  - fit on one machine's disk storage
  - require multiple machines to store
- Smaller large data sets can be handled by standard methods if enough memory is available.
- Very large data sets require specialized methods and algorithms.
- R can be and has been used for programming such methods.
- But standard R methods should be able to handle smaller large data problems on machines with enough memory.

Large Vector Support

- Through R 2.15.2 the total number of elements in a vector cannot exceed $2^{31} - 1 = 2,147,483,647$
- We need a way to raise this limit that meets several goals:
  - avoid having to rewrite too much of R itself
  - avoid requiring package authors to rewrite too much C code
  - avoid having existing compiled C code fail if possible
  - allow incrementally adding support for procedures where it makes sense

Large Vector Support

- C level changes:
  - Preserve existing memory layout
  - Use special marker in length field to identify long vectors
  - LENGTH accessor (returning int) signals an error for long vectors
  - Long vector aware code uses XLENGTH to return R_xlen_t.
- For now, keep $2^{31} - 1$ limit on matrix rows and columns.
  - This allows much existing C/FORTRAN code to be used unchanged
  - In particular, it simplifies use of external BLAS libraries.

Large Vector Support

- R code should not need to be changed:
  - double precision indices can be used for subsetting
  - length will return double for long vectors
  - .C and .Fortran will signal errors for long vectors.
- A document describing how to add long vector support to a package should be available soon.

Large Vector Support

- Most basic vector operations and vectorized functions now support long vectors.
- A number of basic linear algebra functions also work; a few do not.
- Some statistical functions with long vector support:
  - random number generators
  - summaries, such as mean and fivenum
  - sort and related functions
  - lm.fit and glm.fit
- The function dist can handle more than 2^{16} observations by returning a long vector result.
- Many matrix and array functions already support large arrays:
  - colSums, colMeans
  - rowSums, rowMeans

Large Vector Support

- Converting existing methods to support large vectors is fairly straight forward
- However, there are some issues to consider for these methods to be effective:
  - more numerically stable algorithms may be needed
  - faster/parallel algorithms may be needed
  - the ability to interrupt computations may becomes important
- Results returned should probably be streamlined.
- Statistical usefulness may not scale to larger data
- The size where these issues become relevant is likely much lower!

Current Status and Future Work

- The results described so far will be released in R 3.0.0 in April 2013.
- Work will continue on adding long vector support to more core functions.
- Efforts are underway to make more extensive use of parallel computing for vectorized functions and matrix functions.
- Future work will also consider
  - whether to add a separate 64-bit integer type, or change the basic R integer type to 64 bits
  - possibly adding 8 and 16 bit integer types
  - arithmetic and overflow issues that these raise
  - whether to allow numbers of rows and columns in matrices to exceed $2^{31} - 1$ as well
- We will also continue to look for other ways to enhance R's ability to work with big data.