Introduction to the Matrix package

Douglas Bates
R Core Development Group
bates@r-project.org

August 31, 2004

Abstract

Linear algebra is at the core of many areas of statistical computing and from its inception the S language has supported numerical linear algebra via a matrix data type and several functions and operators, such as `%*%`, `qr`, `chol`, and `solve`. However, these data types and functions do not provide direct access to all of the facilities for efficient manipulation of dense matrices, as provided by the Lapack subroutines, and they do not provide for manipulation of sparse matrices.

The `Matrix` package provides a set of S4 classes for dense and sparse matrices that extend the basic matrix data type. Methods for a wide variety of functions and operators applied to objects from these classes provide efficient access to BLAS (Basic Linear Algebra Subroutines), Lapack (dense matrix), TAUCS (sparse matrix) and UMFPACK (sparse matrix) routines. One notable characteristic of the package is that whenever a matrix is factored, the factorization is stored as part of the original matrix so that further operations on the matrix can reuse this factorization.

1 Introduction

Linear algebra is at the core of many statistical computing techniques and, from its inception, the S language has supported numerical linear algebra via a matrix data type and several functions and operators, such as `%*%`, `qr`, `chol`, and `solve`. Initially the numerical linear algebra functions in R called underlying Fortran routines from the Linpack (Dongarra et al., 1979) and Eispack Smith et al. (1976) libraries but over the years most of these functions have been switched to use routines from the Lapack Anderson et al. (1999) library. Furthermore, R can be configured to use accelerated BLAS (Basic Linear Algebra Subroutines), such as those from the Atlas Whaley et al. (2001) project or Goto's BLAS Goto and van de Geijn (2002).

Lapack provides routines for operating on several special forms of matrices, such as triangular matrices and symmetric matrices. Furthermore,matrix decompositions like the QR decompositions produce multiple output components that should be regarded as parts of a single object. There is some support in R

for operations on special forms of matrices (e.g. the `backsolve`, `forwardsolve` and `chol2inv` functions) and for special structures (e.g. a QR structure is implicitly defined as a list by the `qr`, `qr.qy`, `qr.qty`, and related functions) but it is not as fully developed as it could be.

Also there is no direct support for sparse matrices in R although Koenker and Ng (2003) have developed a contributed package for sparse matrices based on SparseKit.

The `Matrix` package provides S4 classes and methods for dense and sparse matrices. The methods for dense matrices use Lapack and BLAS. The sparse matrix methods use TAUCS (Toledo, 2003), UMFPACK (Davis, 2003), and Metis (Karapis, 2003).

2 Classes for dense matrices

The `Matrix` package will provide classes for real (stored as double precision) and complex (stored as double precision complex) dense matrices. At present only the real classes have been implemented. These classes are

geMatrix Real matrices in general storage mode

syMatrix Symmetric real matrices in non-packed storage

trMatrix Triangular real matrices in non-packaged storage

poMatrix Positive semi-definite symmetric real matrices in non-package storage

Methods for these classes include coersion between these classes, when appropriate, and coersion to the `matrix` class; methods for matrix multiplication (`%*%`); cross products (`crossprod`), matrix norm (`norm`); reciprocal condition number (`rcond`); LU factorization (`lu`) or, for the `poMatrix` class, the Cholesky decomposition (`chol`); and solutions of linear systems of equations (`solve`).

Whenever a factorization or a decomposition is calculated it is preserved as a (list) element in the `factorization` slot of the original object. In this way a sequence of operations, such as determining the condition number of a matrix then solving a linear system based on the matrix, do not require multiple factorizations of the same matrix nor do they require the user to store the intermediate results.

3 Classes for sparse matrices

3.1 Representations of sparse matrices

Conceptually, the simplest representation of a sparse matrix is as a triplet of an integer vector `i` giving the row numbers, an integer vector `j` giving the column numbers, and a numeric vector `x` giving the non-zero values in the matrix. An S4 class definition might be

```
setClass("tripletMatrix",
         representation(i = "integer", j = "integer", x = "numeric",
                        Dim = "integer"))
```

The triplet representation is row-oriented if elements in the same row were adjacent and column-oriented if elements in the same column were adjacent. The compressed sparse row (csr) (or compressed sparse column - csc) representation is similar to row-oriented triplet (column-oriented triplet) except that `i` (`j`) just stores the index of the first element in the row (column). (There are a couple of other details but that is the gist of it.) These compressed representations remove the redundant row (column) indices and provide faster access to a given location in the matrix because you only need to check one row (column).

The preferred representation of sparse matrices in the SparseM package is csr. Matlab uses csc. We hope that Octave will also use this representation. There are certain advantages to csc in systems like R and Matlab where dense matrices are stored in column-major order. For example, Sivan Toledo's TAUCS Toledo (2003) library and Tim Davis's UMFPACK Davis (2003) library are both based on csc and can both use level-3 BLAS in certain sparse matrix computations.

The Matrix package provides the following classes for sparse matrices

tripletMatrix general, real, sparse matrices in (a possibly redundant) triplet form. This can be a convenient form in which to construct sparse matrices.

cscMatrix general, real, sparse matrices in the (sorted) compressed sparse column format.

sscMatrix symmetric, real, sparse matrices in the (sorted) compressed sparse column format. Only the upper or the lower triangle is stored. Although there is provision for both forms, the lower triangle form works best with TAUCS.

tscMatrix triangular, real, sparse matrices in the (sorted) compressed sparse column format.

References

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, 3rd edition, 1999.

Tim Davis. Umfpack: Unified multifrontal package. http://www.cise.ufl.edu/research/sparse/umfpack, 2003.

Jack Dongarra, Cleve Moler, Bunch, and G.W. Stewart. *Linpack Users' Guide*. SIAM, 1979.

Kazushige Goto and Robert van de Geijn. On reducing tlb misses in matrix multiplication. Technical Report TR02-55, Department of Computer Sciences, U. of Texas at Austin, 2002.

George Karapis. Metis: Family of multilevel partioning algorithms. http://www-users.cs.umn.edu/ karypis/metis/, 2003.

Roger Koenker and Pin Ng. SparseM: A sparse matrix package for R. *J. of Statistical Software*, 8(6), 2003.

B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystem Routines. EISPACK Guide*, volume 6 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, 1976.

Sivan Toledo. Taucs: A library of sparse linear solvers. http://www.tau.ac.il/ stoledo/taucs/, 2003.

R. Clint Whaley, Antoine Petitet, and Jack J. Dongarra. Automated empirical optimization of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, 2001. Also available as University of Tennessee LAPACK Working Note #147, UT-CS-00-448, 2000 (`www.netlib.org/lapack/lawns/lawn147.ps`).