Using **expm** in packages

Christophe Dutang
ENSIMAG, Grenoble INP

Vincent Goulet
École d'actuariat, Université Laval

Jan. 2008 (added note in June 2010)

1 Introduction

The **expm** package provides an R function `expm` to compute the matrix exponential of a real, square matrix. The matrix exponential of a matrix $\mathbf{A}$ is defined as

$$\begin{aligned} e^{\mathbf{A}} &= \mathbf{I} + \mathbf{A} + \frac{\mathbf{A}^2}{2!} + \dots \\ &= \sum_{k=0}^{\infty} \frac{\mathbf{A}^k}{k!}. \end{aligned}$$

The actual computations are done in C by a function of the same name that is callable by other packages. Therefore, package authors can use these functions and avoid duplication of efforts.

2 Description of the functions

The R function `expm` takes as argument a real, square matrix and returns its exponential. Dimension names are preserved:

```
> library(expm)
> m <- matrix(c(4, 1, 1, 2, 4, 1, 0, 1, 4), 3, 3)
> expm(m)

         [,1]     [,2]      [,3]
[1,] 147.8666 183.7651  71.79703
[2,] 127.7811 183.7651  91.88257
[3,] 127.7811 163.6796 111.96811

> dimnames(m) <- list(letters[1:3], LETTERS[1:3])
> m
```

```
  A B C
a 4 2 0
b 1 4 1
c 1 1 4
```

```
> expm(m)
```

```
         A        B         C
A 147.8666 183.7651  71.79703
B 127.7811 183.7651  91.88257
C 127.7811 163.6796 111.96811
```

Note that the remainder of this text **mainly** relates to `expm(., method = "Ward77")`, i.e., the method of Ward (1977) which is no longer the default method, as e.g., `method = "Higham08"` has found to be ("uniformly") superior, see Higham (2008).

The actual computational work is done in C by a routine defined as

```
void expm(double *x, int n, double *z)
```

where `x` is the vector underlying the R matrix and `n` is the number of lines (or columns) of the matrix. The matrix exponential is returned in `z`. The routine uses the algorithm of Ward (1977) based on diagonal Padé table approximations in conjunction with three step preconditioning. The Padé approximation to $e^{\mathbf{A}}$ is

$$e^{\mathbf{A}} \approx R(\mathbf{A}),$$

with

$$R_{pq}(\mathbf{A}) = (D_{pq}(\mathbf{A}))^{-1} N_{pq}(\mathbf{A})$$

where

$$D_{pq}(\mathbf{A}) = \sum_{j=1}^{p} \frac{(p+q-j)!p!}{(p+q)!j!(p-j)!} \mathbf{A}^j$$

and

$$N_{pq}(\mathbf{A}) = \sum_{j=1}^{q} \frac{(p+q-j)!q!}{(p+q)!j!(q-j)!} \mathbf{A}^j.$$

See Moler and Van Loan (1978) for an exhaustive treatment of the subject.

The C routine is based on a translation made by **?** of the implementation of the corresponding Octave function (Eaton, 2002).

3 Calling the functions from other packages

Package authors can use facilities from **expm** in two (possibly simultaneous) ways:

1. call the R level function `expm` in R code;

2. if matrix exponential calculations are needed in C, call the routine `expm`.

Using R level function `expm` in a package simply requires the following two import directives:

```
Imports: expm
```

in file `DESCRIPTION` and

```
import(expm)
```

in file `NAMESPACE`.

Accessing the C level routine further requires to prototype `expm` and to retrieve its pointer in the package initialization function `R_init_`*`pkg`*, where *`pkg`* is the name of the package:

```
void (*expm)(double *x, int n, double *z);

void R_init_pkg(DllInfo *dll)
{
    expm = (void (*) (double, int, double))  \
            R_GetCCallable("expm", "expm");
}
```

The definitive reference for these matters remains the *Writing R Extensions* manual.

References

J. W. Eaton. *GNU Octave Manual*. Network Theory Limited, 2002. ISBN 0-9541617-2-6. URL `http://www.octave.org`.

N. J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20:801–836, 1978.

R. C. Ward. Numerical computation of the matrix exponential with accuracy estimate. *SIAM Journal on Numerical Analysis*, 14:600–610, 1977.