Package 'rexpokit'

February 20, 2015

Type Package

Title R wrappers for EXPOKIT; other matrix functions

Version 0.24.1

Date 2013-07-08

Maintainer Nicholas J. Matzke <matzke@berkeley.edu>

Depends methods, SparseM, Rcpp (>= 0.9.10)

LinkingTo Rcpp

Description This package wraps some of the matrix exponentiation utilities from EXPOKIT (http://www.maths.uq.edu.au/expokit/), a FORTRAN library that is widely recommended for matrix exponentiation (Sidje RB, 1998. ``Expokit: A Software Package for Computing Matrix Exponentials.'' ACM Trans. Math. Softw. 24(1): 130-156). EXPOKIT includes functions for exponentiating both small, dense matrices, and large, sparse matrices (in sparse matrices, most of the cells have value 0). Rapid matrix exponentiation is useful in phylogenetics when we have a large number of states (as we do when we are inferring the history of transitions between the possible geographic ranges of a species), but is probably useful in other ways as well.

URL http://phylo.wikidot.com/rexpokit

License GPL (>= 2)

LazyLoad yes

ByteCompile true

Author Nicholas J. Matzke [aut, cre, cph],
Roger B. Sidje [aut, cph]

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-07-15 07:25:20

R topics documented:

`rexpokit-package` *Matrix exponentiation with EXPOKIT in R*

Description

Matrix exponentiation with EXPOKIT in R

Details

Package:	rexpokit
Type:	Package
Version:	0.24.1
Date:	2013-07-08
License:	GPL (>= 2)
LazyLoad:	yes

This package wraps some of the matrix exponentiation utilities from EXPOKIT (http://www.maths.uq.edu.au/expokit/), a FORTRAN library that is widely recommended for fast matrix exponentiation (Sidje RB, 1998. "Expokit: A Software Package for Computing Matrix Exponentials." *ACM Trans. Math. Softw.* 24(1): 130-156).

The FORTRAN package was developed by Roger B. Sidje, see http://www.maths.uq.edu.au/expokit/. Nicholas J. Matzke adapted the package for use with R and wrote the R interface. Permission to distribute the EXPOKIT source under GPL was obtained from Roger B. Sidje.

EXPOKIT includes functions for exponentiating both small, dense matrices, and large, sparse matrices (in sparse matrices, most of the cells have value 0). Rapid matrix exponentiation is useful in phylogenetics when we have a large number of states (as we do when we are inferring the history of transitions between the possible geographic ranges of a species), but is probably useful in other

ways as well.

Background

Various messages on discussion boards have asked whether or not there is an R package that uses EXPOKIT. There are only two as of this writing (January 2013) – `diversitree` and `ctarma`. However, diversitree's usage is nested deeply in a series of dynamic functions and integrated with additional libraries (e.g. deSolve) and so is very difficult to extract for general usage, and `ctarma` implements only ZEXPM via `ctarma::zexpm`. Niels Richard Hansen <Niels.R.Hansen@math.ku.dk> is also working on an implementation of certain EXPOKIT functions.

(See the additional notes file EXPOKIT_For_Dummies_notes_v1.txt for additional notes on wrappers for EXPOKIT in Python etc.)

As it turns out, the EXPOKIT documentation and code is far from trivial to figure out, since the code as published does not run "out of the box" – in particular, the Q transition matrix ("matvec"), which is the major input into an exponentiation algorithm, is not input directly, but rather via another function, which requires the user to put together some FORTRAN code to do this and make a wrapper for the core function. I couldn't figure it out in a short amount of time, but Stephen Smith did for his "LAGRANGE" biogeography package, so I copied and modified this chunk of his code to get started.

Installation hints

Installing `rexpokit` from source will require a gfortran compiler to convert the FORTRAN code files in /src (*.f) to object files (*.o), and g++ to compile and link the C++ wrapper. `rexpokit` was developed on an Intel Mac running OS X 10.7. I (NJM) successfully compiled it using g++ and gfortran from (gcc version 4.2.1).

Citation

This code was developed for the following publication. Please cite if used: Matzke, Nicholas J. (2012). "Founder-event speciation in BioGeoBEARS package dramatically improves likelihoods and alters parameter inference in Dispersal-Extinction-Cladogenesis (DEC) analyses." *Frontiers of Biogeography* 4(suppl. 1): 210. Link to abstract and PDF of poster: http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster. (Poster abstract published in the Conference Program and Abstracts of the International Biogeography Society 6th Biannual Meeting, Miami, Florida. Poster Session P10: Historical and Paleo-Biogeography. Poster 129B. January 11, 2013.)

Please also cite Sidje (1998).

Acknowledgements/sources

1. Niels Richard Hansen <Niels.R.Hansen@math.ku.dk> helped greatly with the initial setup of the package. See his `expoRkit` for another R implementation of EXPOKIT routines.

2. EXPOKIT, original FORTRAN package, by Roger B. Sidje <rbs@maths.uq.edu.au>, Department of Mathematics, University of Queensland, Brisbane, QLD-4072, Australia, (c) 1996-2013 All Rights Reserved

Sidje has given permission to include EXPOKIT code in this R package under the usual GPL license

for CRAN R packages. For the full EXPOKIT copyright and license, see `expokit_copyright.txt` under `rexpokit/notes/`.

EXPOKIT was published by Sidje in: Sidje RB (1998). "Expokit. A Software Package for Computing Matrix Exponentials." *ACM-Transactions on Mathematical Software*, 24(1):130-156. http://tinyurl.com/bwa87rq

3. A small amount of C++ code wrapping EXPOKIT was modified from a file in LAGRANGE, C++ version by Stephen Smith:
http://code.google.com/p/lagrange/
https://github.com/blackrim/lagrange

Specifically:

```
* RateMatrix.cpp
*
* Created on: Aug 14, 2009
*   Author: smitty
*
```

...and the my_*.f wrappers for the EXPOKIT *.f code files.

4. Also copied in part (to get the .h file) from:

Python package "Pyprop":
http://code.google.com/p/pyprop/
http://pyprop.googlecode.com/svn/trunk/core/krylov/expokit/expokitpropagator.cpp
http://www.koders.com/python/fidCA95B5A4B2FB77455A72B8A361CF684FFE48F4DC.aspx?s=fourier+transform

Specifically:
pyprop/core/krylov/expokit/f2c/expokit.h

5. The EXPOKIT FORTRAN package is available at:
http://www.maths.uq.edu.au/expokit/

Copyright:
http://www.maths.uq.edu.au/expokit/copyright
...or...
expokit_copyright.txt in this install

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>, Roger B. Sidje <roger.b.sidje@ua.edu>

References

http://www.maths.uq.edu.au/expokit/

http://www.maths.uq.edu.au/expokit/copyright

Matzke N (2012). "Founder-event speciation in BioGeoBEARS package dramatically improves likelihoods and alters parameter inference in Dispersal-Extinction-Cladogenesis (DEC) analyses." _Frontiers of Biogeography_, *4*(suppl. 1), pp. 210. ISSN 1948-6596, Poster abstract published in the Conference Program and Abstracts of the International Biogeography Society 6th Biannual Meeting, Miami, Florida. Poster Session P10: Historical and Paleo-Biogeography. Poster 129B. January 11, 2013, <URL: http://phylo.wikidot.com/matzke-2013-international-biogeography-society-poster

Sidje RB (1998). "Expokit. A Software Package for Computing Matrix Exponentials." _ACM Trans. Math. Softw._, *24*(1), pp. 130-156. <URL: http://dx.doi.org/10.1145/285861.285868>, <URL: http://dl.acm.org/citation.cfm?id=285868>.

Eddelbuettel D and Francois R (2011). "Rcpp: Seamless R and C++ Integration." _Journal of Statistical Software_, *40*(8), pp. 1-18. ISSN 1548-7660, See also: <URL: http://cran.r-project.org/web/packages/Rcpp/vignettes/introduction.pdf>, <URL: http://cran.r-project.org/web/packages/Rcpp/index.html>. , <URL: http://www.jstatsoft.org/v40/i08>.

Moler C and Loan CV (2003). "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later." _SIAM review_, *45*(1), pp. 3-49. <URL: http://www.jstor.org/stable/25054364>.

Foster PG (2001). "The Idiot's Guide to the Zen of Likelihood in a Nutshell in Seven Days for Dummies, Unleashed." Online PDF, widely copied, <URL: http://www.bioinf.org/molsys/data/idiots.pdf>.

See Also

expoRkit expokit_wrapalldmexpv_tvals

Examples

```
# Example code
# For background and basic principles, see rexpokit/notes/EXPOKIT_For_Dummies_notes_v1.txt

library(rexpokit)

# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504,
0.168, 0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)

# Exponentiate each with EXPOKIT's dgpadm (good for small dense matrices)
for (t in tvals)
{
```

```
Pmat = expokit_dgpadm_Qmat(Qmat=Qmat, t=t, transpose_needed=TRUE)
cat("\n\nTime=", t, "\n", sep="")
print(Pmat)
}

# Exponentiate each with EXPOKIT's dmexpv (should be fast for large sparse matrices)
for (t in tvals)
{
Pmat = expokit_dmexpv_Qmat(Qmat=Qmat, t=t, transpose_needed=TRUE)
cat("\n\nTime=", t, "\n", sep="")
print(Pmat)
}

# DMEXPV and DGEXPV are designed for large, sparse Q matrices (sparse = lots of zeros).
# DMEXPV is specifically designed for Markov chains and so may be slower, but more accurate.

# DMEXPV, single t-value
expokit_wrapalldmexpv_tvals(Qmat=Qmat, tvals=tvals[1], transpose_needed=TRUE)
expokit_wrapalldmexpv_tvals(Qmat=Qmat, tvals=2)

# DGEXPV, single t-value
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=tvals[1], transpose_needed=TRUE)
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=2)

# These functions runs the for-loop itself (sadly, we could not get mapply() to work
# on a function that calls dmexpv/dgexpv), returning a list of probability matrices.

# DMEXPV functions
list_of_P_matrices_dmexpv = expokit_wrapalldmexpv_tvals(Qmat=Qmat,
tvals=tvals, transpose_needed=TRUE)
list_of_P_matrices_dmexpv

# DGEXPV functions
list_of_P_matrices_dgexpv = expokit_wrapalldgexpv_tvals(Qmat=Qmat,
tvals=tvals, transpose_needed=TRUE)
list_of_P_matrices_dgexpv

# Check if there are differences in the results (might only happen for large problems)
cat("\n")
cat("Differences between dmexpv and dgexpv\n")

for (i in 1:length(list_of_P_matrices_dmexpv))
{
diffs = list_of_P_matrices_dmexpv[[i]] - list_of_P_matrices_dgexpv[[i]]
print(diffs)
cat("\n")
}
```

---

coo2mat *Convert a COO-formated matrix to standard square format*

Description

EXPOKIT's dmexp-type functions deal with sparse matrices. These have a lot of zeros, and thus can be compressed into COO (coordinated list) format, which is described here:

Usage

```
coo2mat(coomat,
  n = max(max(coomat[, 1]), max(coomat[, 2])),
  transpose_needed = FALSE)
```

Arguments

coomat	a 3-column matrix or data.frame (basically cbind(ia, ja, a))
n	the order of the matrix
transpose_needed	If TRUE (default), matrix will be transposed (apparently EXPOKIT needs the input matrix to be transposed compared to normal)

Details

http://en.wikipedia.org/wiki/Sparse_matrix#Coordinate_list_.28COO.29

In EXPOKIT and its wrapper functions, a COO-formated matrix is input as 3 vectors (first two integer, the third double):

ia = row number
ja = column number
a = value of that cell in the matrix (skipping 0 cells)

This function takes a 3-column matrix or data.frame (basically cbind(ia, ja, a)) and the order of the matrix, n (n = the order of the matrix, i.e. number of rows/columns) and converts back to standard square format.

Value

outmat

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

Examples

```
# Example use:
ia = c(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4)
ja = c(1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4)
a  = c(-1.218, 0.126, 0.168, 0.126, 0.504, -0.882, 0.504,
0.672, 0.336, 0.252, -1.050, 0.252, 0.378, 0.504, 0.378, -1.050)
coomat = cbind(ia, ja, a)
print(coomat)
n = 4
Qmat = coo2mat(coomat, n)
print(Qmat)
```

`expokit_dgexpv_Qmat` *EXPOKIT dgexpv matrix exponentiation on Q matrix*

Description

This function converts a matrix to COO format and exponentiates it via the EXPOKIT dgexpv function (designed for sparse matrices) and wrapper functions `wrapalldgexpv_` around dgexpv.

Usage

```
expokit_dgexpv_Qmat(Qmat = NULL, t = 2.1,
  inputprobs_for_fast = NULL, transpose_needed = TRUE,
  transform_to_coo_TF = TRUE, coo_n = NULL, anorm = NULL,
  check_for_0_rows = TRUE)
```

Arguments

`Qmat`	an input Q transition matrix
`t`	a time value to exponentiate by
`inputprobs_for_fast`	If NULL (default), the full probability matrix (Pmat) is returned. However, the full speed of EXPOKIT on sparse matrices will be exploited if inputprobs_for_fast=c(starting probabilities). In this case these starting probabilities are input to `myDMEXPV` directly, as `v`, and `w`, the output probabilities, are returned.
`transpose_needed`	If TRUE (default), matrix will be transposed (apparently EXPOKIT needs the input matrix to be transposed compared to normal)
`transform_to_coo_TF`	Should the matrix be tranposed to COO? COO format is required for EXPOKIT's sparse-matrix functions (like dmexpv and unlike the padm-related functions. Default TRUE; if FALSE, user must put a COO-formated matrix in `Qmat`. Supplying the coo matrix is probably faster for repeated calculations on large matrices.

coo_n If a COO matrix is input, coo_n specified the order (# rows, equals # columns) of the matrix.

anorm dgexpv requires an initial guess at the norm of the matrix. Using the R function norm might get slow with large matrices. If so, the user can input a guess manually (Lagrange seems to just use 1 or 0, if I recall correctly).

check_for_0_rows
If TRUE or a numeric value, the input Qmat is checked for all-zero rows, since these will crash the FORTRAN wrapalldmexpv function. A small nonzero value set to check_for_0_rows or the default (0.0000000000001) is input to off-diagonal cells in the row (and the diagonal value is normalized), which should fix the problem.

Details

NOTE: DGEXPV vs. DMEXPV. According to the EXPOKIT documentation, DGEXPV should be faster than DMEXPV, however DMEXPV runs an accuracy check appropriate for Markov chains, which is not done in DGEXPV.

From EXPOKIT:

```
* The method used is based on Krylov subspace   projection
* techniques and the matrix under   consideration interacts only
* via the external   routine 'matvec' performing the matrix-vector
* product (matrix-free method).
*
* This [DMEXPV, not DGEXPV -- NJM] is a customised  version for Markov Chains. This means that a
*   check is done within this code to ensure that the   resulting vector
* w is a probability vector,  i.e., w must have all its components
* in   [0,1], with sum equal to 1. This check is done at some   expense
* and the user may try DGEXPV which is   cheaper since it ignores
* probability   constraints.
```

I (NJM) have not noticed a difference between the outputs of these two functions, but it might occur with large matrices.

COO (coordinated list) format is a compressed format that is required for EXPOKIT's sparse-matrix functions (like dgexpv and unlike EXPOKIT's padm-related functions. COO format is described here:

http://en.wikipedia.org/wiki/Sparse_matrix#Coordinate_list_.28COO.29

If Qmat is NULL (default), a default matrix is input.

Value

tmpoutmat the output matrix. wrapalldgexpv_ produces additional output relating to accuracy of the output matrix etc.; these can be by a direct call of dgexpv.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

mat2coo

expokit_dgexpv_wrapper

expokit_wrapalldgexpv_tvals

Examples

```
# Example:
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)

# Exponentiate each with EXPOKIT's dgexpv (should be fast for large sparse matrices)
for (t in tvals)
{
Pmat = expokit_dgexpv_Qmat(Qmat=Qmat, t=t, transpose_needed=TRUE)
cat("\n\nTime=", t, "\n", sep="")
print(Pmat)
}

# DMEXPV and DGEXPV are designed for large, sparse Q matrices (sparse = lots of zeros).
# DMEXPV is specifically designed for Markov chains and so may be slower, but more accurate.

# DGEXPV, single t-value
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=tvals[1], transpose_needed=TRUE)
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=2)

# This function runs the for-loop itself (sadly, we could not get mapply() to work
# on a function that calls dmexpv/dgexpv), returning a list of probability matrices.

# DGEXPV functions
list_of_P_matrices_dgexpv = expokit_wrapalldgexpv_tvals(Qmat=Qmat,
tvals=tvals, transpose_needed=TRUE)
list_of_P_matrices_dgexpv
```

`expokit_dgexpv_wrapper` *EXPOKIT dgexpv wrapper function*

Description

This function wraps the .C call to EXPOKIT for the dgexpv function. Only the output probability matrix is returned.

Usage

```
expokit_dgexpv_wrapper(n, m, timeval, v, w, tol, anorm,
  wsp, lwsp, iwsp, liwsp, itrace, iflag, ia, ja, a, nz,
  res)
```

Arguments

`n`	number of rows in Q matrix
`m`	n-1
`timeval`	the value to exponentiate the rate matrix by (often e.g. a time value)
`v`	variable to store some results in; should have n elements (and perhaps start with 1)
`w`	same length as v
`tol`	tolerance for approximations; usually set to 0.01
`anorm`	the norm of the Q matrix
`lwsp`	length of workspace (wsp); for dgexpv, lwsp=n*(m+2)+5*(m+2)^2+ideg+1
`wsp`	workspace to store some results in; should be a double with lwsp elements
`liwsp`	length of integer workspace; for dgexpv, liwsp=m+2
`iwsp`	integer workspace
`itrace`	option, set to 0
`iflag`	option, set to 0
`ia`	i indices of Qmat nonzero values
`ja`	j indices of Qmat nonzero values
`a`	nonzero values of Qmat (ia, ja, a are columns of a COO-formatted Q matrix)
`nz`	number of non-zeros in Qmat
`res`	space for output probability matrix (n x n)
	EXPOKIT needs the input matrix to be transposed compared to normal) COO format is required for EXPOKIT.

Details

NOTE: DGEXPV vs. DMEXPV. According to the EXPOKIT documentation, DGEXPV should be faster than DMEXPV, however DMEXPV runs an accuracy check appropriate for Markov chains, which is not done in DGEXPV.

Value

`tmpoutmat` the output matrix for the (first) input t-value

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

expokit_dgexpv_Qmat

expokit_wrapalldgexpv_tvals

Examples

```
# Example building the inputs from scratch:

# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)
timeval = tvals[2]

ideg = as.integer(6)
n=nrow(Qmat)
m=n-1
# t=as.numeric(2.1)

# v should have as many elements as n; first element = 1 (?)
v=double(n)
v[1] = 1

# w is the same length
w = double(length=n)
tol=as.numeric(0.01)

# length of wsp
```

```
#lwsp = as.integer(n*(m+1)+n+(m+2)^2 + 4*(m+2)^2+ideg+1)
#lwsp = as.integer(n*(m+1)+n+(m+2)^2 + 5*(m+2)^2+ideg+1)
lwsp = as.integer(n*(m+2)+5*(m+2)^2+ideg+1)

#lwsp = 100
wsp = double(length=lwsp)

# length of iwsp
liwsp = max(m+2, 7)
iwsp = integer(length=liwsp)

res = double(length=n*n)

#matvec = matrix(data=Q, nrow=n, byrow=TRUE)
matvec = Qmat
tmatvec = t(matvec)
rowSums(tmatvec)
colSums(tmatvec)

# type="O" is being used here, this is supposed to be the
# default for norm(), although it throws an error if not
# specified
#
# From the help:
# type - character string, specifying the type of matrix norm to be
# computed. A character indicating the type of norm desired.
#  "O", "o" or "1"
#  specifies the one norm, (maximum absolute column sum);
anorm = as.numeric(norm(matvec, type="O"))
#anorm = 1


itrace = 0
iflag = 0


#a = as.numeric(tmatvec)
#a = as.numeric(matvec)
tmpmat = tmatvec
tmpmat_in_REXPOKIT_coo_fmt = mat2coo(tmpmat)
ia = tmpmat_in_REXPOKIT_coo_fmt[,"ia"]
ja = tmpmat_in_REXPOKIT_coo_fmt[,"ja"]
a = tmpmat_in_REXPOKIT_coo_fmt[,"a"]

 # Number of non-zeros
 nz = nrow(Qmat) * ncol(Qmat)

# Run the wrapper function

tmpoutmat = expokit_dgexpv_wrapper(n, m, timeval, v, w, tol, anorm, wsp,
lwsp, iwsp, liwsp, itrace, iflag, ia, ja, a, nz, res)

print(tmpoutmat)
```

`expokit_dgpadm_Qmat` *EXPOKIT dgpadm matrix exponentiation on Q matrix*

Description

This function exponentiates a matrix via the EXPOKIT padm function (designed for small dense matrices) and wrapper function `wrapalldgpadm_` around dmexpv.

Usage

```
expokit_dgpadm_Qmat(Qmat = NULL, t = 2.1,
  transpose_needed = TRUE)
```

Arguments

`Qmat`	an input Q transition matrix
`t`	one or more time values to exponentiate by
`transpose_needed`	If TRUE (default), matrix will be transposed (apparently EXPOKIT needs the input matrix to be transposed compared to normal)

Details

From EXPOKIT:

```
* Computes exp(t*H), the matrix exponential of a   general matrix in
* full, using the   irreducible rational Pade approximation to the
* exponential function exp(x) = r(x) = (+/-)( I +   2*(q(x)/p(x)) ),
* combined with   scaling-and-squaring.
```

If `Qmat` is NULL (default), a default matrix is input.

Value

`tmpoutmat` the output matrix. `wrapalldmexpv_` produces additional output relating to accuracy of the output matrix etc.; these can be obtained by a direct call of wrapalldmexpv_.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

`mat2coo`

Examples

```
# Example:
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)

# Exponentiate each with EXPOKIT's dgpadm (good for small dense matrices)
for (t in tvals)
{
Pmat = expokit_dgpadm_Qmat(Qmat=Qmat, t=t, transpose_needed=TRUE)
cat("\n\nTime=", t, "\n", sep="")
print(Pmat)
}
```

`expokit_dmexpv_Qmat` *EXPOKIT dmexpv matrix exponentiation on Q matrix*

Description

This function converts a matrix to COO format and exponentiates it via the EXPOKIT dmexpv function (designed for sparse matrices) and wrapper functions `wrapalldmexpv_` around dmexpv.

Usage

```
expokit_dmexpv_Qmat(Qmat = NULL, t = 2.1,
  inputprobs_for_fast = NULL, transpose_needed = TRUE,
  transform_to_coo_TF = TRUE, coo_n = NULL, anorm = NULL,
  check_for_0_rows = TRUE)
```

Arguments

`Qmat`	an input Q transition matrix
`t`	one or more time values to exponentiate by
`inputprobs_for_fast`	If NULL (default), the full probability matrix (Pmat) is returned. However, the full speed of EXPOKIT on sparse matrices will be exploited if inputprobs_for_fast=c(starting probabilities). In this case these starting probabilities are input to `myDMEXPV` directly, as `v`, and `w`, the output probabilities, are returned.

`transpose_needed`
: If TRUE (default), matrix will be transposed (apparently EXPOKIT needs the input matrix to be transposed compared to normal)

`transform_to_coo_TF`
: Should the matrix be tranposed to COO? COO format is required for EXPOKIT's sparse-matrix functions (like dmexpv and unlike the padm-related functions. Default TRUE; if FALSE, user must put a COO-formated matrix in `Qmat`. Supplying the coo matrix is probably faster for repeated calculations on large matrices.

`coo_n`
: If a COO matrix is input, `coo_n` specified the order (# rows, equals # columns) of the matrix.

`anorm`
: `dmexpv` requires an initial guess at the norm of the matrix. Using the

`check_for_0_rows`
: If TRUE or a numeric value, the input Qmat is checked for all-zero rows, since these will crash the FORTRAN wrapalldmexpv function. A small nonzero value set to check_for_0_rows or the default (0.0000000000001) is input to off-diagonal cells in the row (and the diagonal value is normalized), which should fix the problem. R function `norm` might get slow with large matrices. If so, the user can input a guess manually (`Lagrange` seems to just use 1 or 0, if I recall correctly).

Details

From EXPOKIT:

```
* The method used is based on   Krylov subspace projection
* techniques and the   matrix under consideration interacts only
* via   the external routine 'matvec' performing the   matrix-vector
* product (matrix-free   method).
*
* This is a customised   version for Markov Chains. This means that a
*   check is done within this code to ensure that the   resulting vector
* w is a probability vector,  i.e., w must have all its components
* in   [0,1], with sum equal to 1. This check is done at some   expense
* and the user may try DGEXPV which is   cheaper since it ignores
* probability   constraints.
```

COO (coordinated list) format is a compressed format that is
required for EXPOKIT's sparse-matrix functions (like dmexpv and
unlike EXPOKIT's padm-related functions.

COO (coordinated list) format is described here:

http://en.wikipedia.org/wiki/Sparse_matrix#Coordinate_list_.28COO.29

If `Qmat` is NULL (default), a default matrix is input.

Value

tmpoutmat the output matrix. wrapalldmexpv_ produces additional output relating to accuracy of the output matrix etc.; these can be by a direct call of dmexpv.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

mat2coo

expokit_dmexpv_wrapper

expokit_wrapalldmexpv_tvals

Examples

```
# Example:
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)

# Exponentiate each with EXPOKIT's dmexpv (should be fast for large sparse matrices)
for (t in tvals)
{
Pmat = expokit_dmexpv_Qmat(Qmat=Qmat, t=t, transpose_needed=TRUE)
cat("\n\nTime=", t, "\n", sep="")
print(Pmat)
}
```

---

expokit_dmexpv_wrapper

EXPOKIT dmexpv wrapper function

Description

This function wraps the .C call to EXPOKIT for the dmexpv function. Only the output probability matrix is returned.

Usage

```
expokit_dmexpv_wrapper(n, m, t, v, w, tol, anorm, wsp,
  lwsp, iwsp, liwsp, itrace, iflag, ia, ja, a, nz, res)
```

Arguments

n	number of rows in Q matrix
m	n-1
t	the value to exponentiate the rate matrix by (often e.g. a time value)
v	variable to store some results in; should have n elements (and perhaps start with 1)
w	same length as v
tol	tolerance for approximations; usually set to 0.01
anorm	the norm of the Q matrix
lwsp	length of workspace (wsp); for dmexpv, lwsp=n*(m+2)+5*(m+2)^2+ideg+1
wsp	workspace to store some results in; should be a double with lwsp elements
liwsp	length of integer workspace; for dmexpv, liwsp=m+2
iwsp	integer workspace
itrace	option, set to 0
iflag	option, set to 0
ia	i indices of Qmat nonzero values
ja	j indices of Qmat nonzero values
a	nonzero values of Qmat (ia, ja, a are columns of a COO-formatted Q matrix)
nz	number of non-zeros in Qmat
res	space for output probability matrix (n x n)
	EXPOKIT needs the input matrix to be transposed compared to normal. COO format is required for EXPOKIT.

Value

`tmpoutmat` the output matrix for the (first) input t-value

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

`expokit_dmexpv_Qmat`

`expokit_wrapalldmexpv_tvals`

Examples

```
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)

# DMEXPV and DGEXPV are designed for large, sparse Q matrices (sparse = lots of zeros).
# DMEXPV is specifically designed for Markov chains and so may be slower, but more accurate.

# DMEXPV, single t-value
expokit_wrapalldmexpv_tvals(Qmat=Qmat, tvals=tvals[1], transpose_needed=TRUE)
expokit_wrapalldmexpv_tvals(Qmat=Qmat, tvals=2)

# This function runs a for-loop itself (sadly, we could not get mapply() to work
# on a function that calls dmexpv/dgexpv), returning a list of probability matrices.

# DMEXPV functions
list_of_P_matrices_dmexpv = expokit_wrapalldmexpv_tvals(Qmat=Qmat,
tvals=tvals, transpose_needed=TRUE)
list_of_P_matrices_dmexpv
```

`expokit_mydgexpv_wrapper` *EXPOKIT dgexpv wrapper function, return just output probs*

Description

This function wraps the .C call to EXPOKIT for the dgexpv function. Only the output probabilities not the Pmat probability matrix, are returned.

Usage

```
expokit_mydgexpv_wrapper(n, m, t, v, w, tol, anorm, wsp,
  lwsp, iwsp, liwsp, itrace, iflag, ia, ja, a, nz)
```

Arguments

n	number of rows in Q matrix
m	n-1

t	the value to exponentiate the rate matrix by (often e.g. a time value)
v	variable to store some results in; should have n elements (and perhaps start with 1)
w	same length as v
tol	tolerance for approximations; usually set to 0.01
anorm	the norm of the Q matrix
lwsp	length of workspace (wsp); for dgexpv, lwsp=n*(m+2)+5*(m+2)^2+ideg+1
wsp	workspace to store some results in; should be a double with lwsp elements
liwsp	length of integer workspace; for dgexpv, liwsp=m+2
iwsp	integer workspace
itrace	option, set to 0
iflag	option, set to 0
ia	i indices of Qmat nonzero values
ja	j indices of Qmat nonzero values
a	nonzero values of Qmat (ia, ja, a are columns of a COO-formatted Q matrix)
nz	number of non-zeros in Qmat EXPOKIT needs the input matrix to be transposed compared to normal. COO format is required for EXPOKIT.

Value

w_output_probs the output probabilities (= myDGEXPV variable w, or the fifth output in the output from .Call("mydgexpv_", ...), given the (first) input t-value.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

expokit_dgexpv_Qmat

expokit_wrapalldgexpv_tvals

Examples

```
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)
```

```
# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)

# dgexpv and DGEXPV are designed for large, sparse Q matrices (sparse = lots of zeros).
# dgexpv is specifically designed for Markov chains and so may be slower, but more accurate.

# dgexpv, single t-value
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=tvals[1], transpose_needed=TRUE)
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=2)

# This function runs a for-loop itself (sadly, we could not get mapply() to work
# on a function that calls dgexpv/dgexpv), returning a list of probability matrices.

# dgexpv functions
list_of_P_matrices_dgexpv = expokit_wrapalldgexpv_tvals(Qmat=Qmat,
tvals=tvals, transpose_needed=TRUE)
list_of_P_matrices_dgexpv
```

`expokit_mydmexpv_wrapper` *EXPOKIT dmexpv wrapper function, return just output probs*

Description

This function wraps the .C call to EXPOKIT for the dmexpv function. Only the output probabilities not the Pmat probability matrix, are returned.

Usage

```
expokit_mydmexpv_wrapper(n, m, t, v, w, tol, anorm, wsp,
  lwsp, iwsp, liwsp, itrace, iflag, ia, ja, a, nz)
```

Arguments

`n`	number of rows in Q matrix
`m`	n-1
`t`	the value to exponentiate the rate matrix by (often e.g. a time value)
`v`	variable to store some results in; should have n elements (and perhaps start with 1)
`w`	same length as v
`tol`	tolerance for approximations; usually set to 0.01
`anorm`	the norm of the Q matrix
`lwsp`	length of workspace (wsp); for dmexpv, lwsp=n*(m+2)+5*(m+2)^2+ideg+1
`wsp`	workspace to store some results in; should be a double with lwsp elements
`liwsp`	length of integer workspace; for dmexpv, liwsp=m+2

iwsp	integer workspace
itrace	option, set to 0
iflag	option, set to 0
ia	i indices of Qmat nonzero values
ja	j indices of Qmat nonzero values
a	nonzero values of Qmat (ia, ja, a are columns of a COO-formatted Q matrix)
nz	number of non-zeros in Qmat EXPOKIT needs the input matrix to be transposed compared to normal. COO format is required for EXPOKIT.

Value

w_output_probs the output probabilities (= myDMEXPV variable w, or the fifth output in the output from .Call("mydmexpv_", ...), given the (first) input t-value.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

expokit_dmexpv_Qmat

expokit_wrapalldmexpv_tvals

Examples

```
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)

# DMEXPV and DGEXPV are designed for large, sparse Q matrices (sparse = lots of zeros).
# DMEXPV is specifically designed for Markov chains and so may be slower, but more accurate.

# DMEXPV, single t-value
expokit_wrapalldmexpv_tvals(Qmat=Qmat, tvals=tvals[1], transpose_needed=TRUE)
expokit_wrapalldmexpv_tvals(Qmat=Qmat, tvals=2)

# This function runs a for-loop itself (sadly, we could not get mapply() to work
# on a function that calls dmexpv/dgexpv), returning a list of probability matrices.
```

```
# DMEXPV functions
list_of_P_matrices_dmexpv = expokit_wrapalldmexpv_tvals(Qmat=Qmat,
tvals=tvals, transpose_needed=TRUE)
list_of_P_matrices_dmexpv
```

`expokit_wrapalldgexpv_tvals` *Run EXPOKIT's dgexpv on one or more t-values*

Description

The function runs EXPOKIT's dgexpv function on a Q matrix and *one or more* time values. If `Qmat` is NULL (default), a default matrix is input.

Usage

```
expokit_wrapalldgexpv_tvals(Qmat = NULL, tvals = c(2.1),
  inputprobs_for_fast = NULL, transpose_needed = TRUE,
  transform_to_coo_TF = TRUE, coo_n = NULL,
  force_list_if_1_tval = FALSE, check_for_0_rows = TRUE)
```

Arguments

`Qmat`	an input Q transition matrix
`tvals`	one or more time values to exponentiate by (doesn't have to literally be a time value, obviously)
`inputprobs_for_fast`	If NULL (default), the full probability matrix (Pmat) is returned. However, the full speed of EXPOKIT on sparse matrices will be exploited if inputprobs_for_fast=c(starting probabilities). In this case these starting probabilities are input to `myDMEXPV` directly, as `v`, and `w`, the output probabilities, are returned.
`transpose_needed`	If TRUE (default), matrix will be transposed (apparently EXPOKIT needs the input matrix to be transposed compared to normal)
`transform_to_coo_TF`	Should the matrix be tranposed to COO? COO format is required for EXPOKIT's sparse-matrix functions (like dmexpv and unlike the padm-related functions. Default TRUE; if FALSE, user must put a COO-formated matrix in `Qmat`. Supplying the coo matrix is probably faster for repeated calculations on large matrices.
`coo_n`	If a COO matrix is input, `coo_n` specified the order (# rows, equals # columns) of the matrix.
`force_list_if_1_tval`	Default FALSE, but set to TRUE if you want a single matrix to be returned inside a list

check_for_0_rows
: If TRUE or a numeric value, the input Qmat is checked for all-zero rows, since these will crash the FORTRAN wrapalldmexpv function. A small nonzero value set to check_for_0_rows or the default (0.0000000000001) is input to off-diagonal cells in the row (and the diagonal value is normalized), which should fix the problem.

Details

NOTE: DGEXPV vs. DMEXPV. According to the EXPOKIT documentation, DGEXPV should be faster than DMEXPV, however DMEXPV runs an accuracy check appropriate for Markov chains, which is not done in DGEXPV.

Value

`tmpoutmat` the output matrix, if 1 t-value is input; `list_of_matrices_output`, if more than 1 t-value is input; to get a single output matrix in a list, set `force_list_if_1_tval=TRUE`

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

expokit_dgexpv_wrapper

expokit_dgexpv_Qmat

Examples

```
# Example:
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)

# Exponentiate each with EXPOKIT's dgexpv (should be fast for large sparse matrices)
for (t in tvals)
{
Pmat = expokit_dgexpv_Qmat(Qmat=Qmat, t=t, transpose_needed=TRUE)
cat("\n\nTime=", t, "\n", sep="")
print(Pmat)
```

```
}

# DMEXPV and DGEXPV are designed for large, sparse Q matrices (sparse = lots of zeros).
# DMEXPV is specifically designed for Markov chains and so may be slower, but more accurate.

# DMEXPV, single t-value

# DGEXPV, single t-value
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=tvals[1], transpose_needed=TRUE)
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=2)

# These functions runs the for-loop itself (sadly, we could not get mapply() to work
# on a function that calls dmexpv/dgexpv), returning a list of probability matrices.

# DGEXPV functions
list_of_P_matrices_dgexpv = expokit_wrapalldgexpv_tvals(Qmat=Qmat,
tvals=tvals, transpose_needed=TRUE)
list_of_P_matrices_dgexpv
```

`expokit_wrapalldmexpv_tvals` *Run EXPOKIT's dmexpv on one or more t-values*

Description

The function runs EXPOKIT's `dmexpv` function on a Q matrix and *one or more* time values. If `Qmat` is NULL (default), a default matrix is input.

Usage

```
expokit_wrapalldmexpv_tvals(Qmat = NULL, tvals = c(2.1),
  inputprobs_for_fast = NULL, transpose_needed = TRUE,
  transform_to_coo_TF = TRUE, coo_n = NULL,
  force_list_if_1_tval = FALSE, check_for_0_rows = TRUE)
```

Arguments

`Qmat`	an input Q transition matrix
`tvals`	one or more time values to exponentiate by (doesn't have to literally be a time value, obviously)
`inputprobs_for_fast`	If NULL (default), the full probability matrix (Pmat) is returned. However, the full speed of EXPOKIT on sparse matrices will be exploited if inputprobs_for_fast=c(starting probabilities). In this case these starting probabilities are input to `myDMEXPV` directly, as `v`, and `w`, the output probabilities, are returned.
`transpose_needed`	If TRUE (default), matrix will be transposed (apparently EXPOKIT needs the input matrix to be transposed compared to normal)

`transform_to_coo_TF`
: Should the matrix be tranposed to COO? COO format is required for EXPOKIT's sparse-matrix functions (like dmexpv and unlike the padm-related functions. Default TRUE; if FALSE, user must put a COO-formated matrix in `Qmat`. Supplying the coo matrix is probably faster for repeated calculations on large matrices.

`coo_n`
: If a COO matrix is input, `coo_n` specified the order (# rows, equals # columns) of the matrix.

`force_list_if_1_tval`
: Default FALSE, but set to TRUE if you want a single matrix to be returned inside a list

`check_for_0_rows`
: If TRUE or a numeric value, the input Qmat is checked for all-zero rows, since these will crash the FORTRAN wrapalldmexpv function. A small nonzero value set to check_for_0_rows or the default (0.0000000000001) is input to off-diagonal cells in the row (and the diagonal value is normalized), which should fix the problem.

Value

`tmpoutmat` the output matrix, if 1 t-value is input; `list_of_matrices_output`, if more than 1 t-value is input; to get a single output matrix in a list, set `force_list_if_1_tval=TRUE`

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

expokit_dmexpv_wrapper

expokit_dmexpv_Qmat

Examples

```
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)

# DMEXPV and DGEXPV are designed for large, sparse Q matrices (sparse = lots of zeros).
# DMEXPV is specifically designed for Markov chains and so may be slower, but more accurate.
```

```
# DGEXPV, single t-value
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=tvals[1], transpose_needed=TRUE)
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=2)

# This function runs the for-loop itself (sadly, we could not get mapply() to work
# on a function that calls dmexpv/dgexpv), returning a list of probability matrices.

# DGEXPV functions
list_of_P_matrices_dgexpv = expokit_wrapalldgexpv_tvals(Qmat=Qmat,
tvals=tvals, transpose_needed=TRUE)
list_of_P_matrices_dgexpv
```

`fermat.test` *Test an integer for primality with Fermat's little theorem.*

Description

Fermat's little theorem states that if n is a prime number and a is any positive integer less than n, then a raised to the nth power is congruent to $a \ modulo \ n$.

Usage

```
fermat.test(n)
```

Arguments

n	the integer to test for primality

Value

Whether the integer passes the Fermat test for a randomized $0 < a < n$ no workyATcallGraph-Primitives

Note

`fermat.test` doesn't work for integers above approximately fifteen because modulus loses precision.

Author(s)

Peter Danenberg <pcd@roxygen.org>

References

http://en.wikipedia.org/wiki/Fermat's_little_theorem

findrows_w_all_zeros *Check if a Q matrix has rows with all zeros*

Description

Q matrices with all-zero rows will crash .Call(wrapalldmexpv_, ...) and .Call(wrapalldgexpv_, ...), and therefore will crash expokit_wrapalldmexpv_tvals() and expokit_wrapalldgexpv_tvals() when these are set (the default) to return the full P matrix. These functions work fine with zero rows if `inputprobs_for_fast` is supplied, meaning that only the output probabilities of each state are returned.

Usage

```
findrows_w_all_zeros(matvec)
```

Arguments

`matvec`	Q transition matrix

Value

A list of TRUE/FALSE, as long as the number of rows. `TRUE`=the is all zeros, `FALSE`=the row has nonzero values.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

expokit_wrapalldmexpv_tvals

expokit_wrapalldgexpv_tvals

Examples

```
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)
```

```
# DMEXPV and DGEXPV are designed for large, sparse Q matrices (sparse = lots of zeros).
# DMEXPV is specifically designed for Markov chains and so may be slower, but more accurate.

# DGEXPV, single t-value
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=tvals[1], transpose_needed=TRUE)
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=2)

# This function runs the for-loop itself (sadly, we could not get mapply() to work
# on a function that calls dmexpv/dgexpv), returning a list of probability matrices.

# DGEXPV functions
list_of_P_matrices_dgexpv = expokit_wrapalldgexpv_tvals(Qmat=Qmat,
tvals=tvals, transpose_needed=TRUE)
list_of_P_matrices_dgexpv
```

`mat2coo` *Convert matrix to COO format using SparseM function*

Description

Converts a matrix to COO format using the SparseM function, presumably this is faster than using a for-loop.

Usage

```
mat2coo(tmpmat)
```

Arguments

`tmpmat`	A square matrix

Details

EXPOKIT's dmexp-type functions deal with sparse matrices. These have a lot of zeros, and thus can be compressed into COO (coordinated list) format, which is described here:

http://en.wikipedia.org/wiki/Sparse_matrix#Coordinate_list_.28COO.29

In EXPOKIT and its wrapper functions, a COO-formated matrix is input as 3 vectors (first two integer, the third double):

ia = row number
ja = column number
a = value of that cell in the matrix (skipping 0 cells)

Value

tmpmat_in_REXPOKIT_coo_fmt A cbind of ia, ja, and a

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

mat2coo_forloop

Examples

```
# Example use:
```

---

mat2coo_forloop *Convert matrix to COO format using nested for-loops*

Description

Converts a matrix to COO format. This version of the function uses for-loops, which is presumably less efficient than mat2coo.

Usage

```
mat2coo_forloop(tmpmat)
```

Arguments

tmpmat	A square matrix

Value

tmpmat_in_REXPOKIT_coo_fmt A cbind of ia, ja, and a

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

mat2coo

Examples

```
# Example use:
# Make a Q matrix
tmpmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Convert to REXPOKIT coo format
tmpmat_in_REXPOKIT_coo_fmt = mat2coo_forloop(tmpmat)
tmpmat_in_REXPOKIT_coo_fmt
```

---

row_allzero_TF *Check if a row is all zeros*

Description

Q matrices with all-zero rows will crash .Call(wrapalldmexpv_, ...) and .Call(wrapalldgexpv_, ...), and therefore will crash expokit_wrapalldmexpv_tvals() and expokit_wrapalldgexpv_tvals() when these are set (the default) to return the full P matrix. These functions work fine with zero rows if `inputprobs_for_fast` is supplied, meaning that only the output probabilities of each state are returned.

Usage

```
row_allzero_TF(tmprow)
```

Arguments

`tmprow`	A row of a Q transition matrix

Value

`TRUE` if tmprow is all zeros, FALSE if not.

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

`expokit_wrapalldmexpv_tvals`

`expokit_wrapalldgexpv_tvals`

Examples

```
# Make a square instantaneous rate matrix (Q matrix)
# This matrix is taken from Peter Foster's (2001) "The Idiot's Guide
# to the Zen of Likelihood in a Nutshell in Seven Days for Dummies,
# Unleashed" at:
# \url{http://www.bioinf.org/molsys/data/idiots.pdf}
#
# The Q matrix includes the stationary base freqencies, which Pmat
# converges to as t becomes large.
Qmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Make a series of t values
tvals = c(0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 14)

# DMEXPV and DGEXPV are designed for large, sparse Q matrices (sparse = lots of zeros).
# DMEXPV is specifically designed for Markov chains and so may be slower, but more accurate.

# DGEXPV, single t-value
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=tvals[1], transpose_needed=TRUE)
expokit_wrapalldgexpv_tvals(Qmat=Qmat, tvals=2)

# This function runs the for-loop itself (sadly, we could not get mapply() to work
# on a function that calls dmexpv/dgexpv), returning a list of probability matrices.

# DGEXPV functions
list_of_P_matrices_dgexpv = expokit_wrapalldgexpv_tvals(Qmat=Qmat,
tvals=tvals, transpose_needed=TRUE)
list_of_P_matrices_dgexpv
```

`SparseM_coo_to_REXPOKIT_coo` *Convert a SparseM COO matrix to a plain matrix*

Description

Converts a SparseM COO-formatted matrix (an S4 object) to a plain matrix, with
column #1 = ia = i index
column #2 = ja = j index
column #3 = a = nonzero values of the matrix

Usage

```
SparseM_coo_to_REXPOKIT_coo(tmpmat_in_SparseMcoo_fmt)
```

Arguments

`tmpmat_in_SparseMcoo_fmt` A square matrix S4 object derived from SparseM's as.matrix.coo

Details

Background: COO (coordinated list) format, is described here:

http://en.wikipedia.org/wiki/Sparse_matrix#Coordinate_list_.28COO.29

In EXPOKIT and its wrapper functions, a COO-formated matrix is input as 3 vectors (first two integer, the third double):

ia = row number
ja = column number
a = value of that cell in the matrix (skipping 0 cells)

Value

tmpmat_in_REXPOKIT_coo_fmt A cbind of ia, ja, and a

Author(s)

Nicholas J. Matzke <matzke@berkeley.edu>

See Also

mat2coo_forloop

Examples

```
# Example use:
# Make a Q matrix
tmpmat = matrix(c(-1.218, 0.504, 0.336, 0.378, 0.126, -0.882, 0.252, 0.504, 0.168,
0.504, -1.05, 0.378, 0.126, 0.672, 0.252, -1.05), nrow=4, byrow=TRUE)

# Covert to SparseM coo format
tmpmat_in_SparseMcoo_fmt = as.matrix.coo(tmpmat)

# Convert to REXPOKIT coo format
tmpmat_in_REXPOKIT_coo_fmt = SparseM_coo_to_REXPOKIT_coo(tmpmat_in_SparseMcoo_fmt)
tmpmat_in_REXPOKIT_coo_fmt
```

Index