Package 'rexpokit'

July 5, 2020

Type Package

Title R Wrappers for EXPOKIT; Other Matrix Functions

Version 0.26.6.7

Date 2020-07-03

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

Maintainer Nicholas J. Matzke <nickmatzke.ncse@gmail.com>

Depends R (>= 3.0.2)

Imports stats, Rcpp (>= 0.11.0)

LinkingTo Rcpp (>= 0.11.0)

Description 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 yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-07-04 22:20:04 UTC

R topics documented:

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

Description

Matrix exponentiation with EXPOKIT in R

Details

Package:	rexpokit
Type:	Package
Version:	0.26.6.7
Date:	2020-07-03

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.

Help

For help with rexpokit or BioGeoBEARS, see (1) the PhyloWiki websites for rexpokit and BioGeoBEARS (http://phylo.wikidot.com/rexpokit, http://phylo.wikidot.com/biogeobears) and (2) the BioGeoBEARS Google Group (https://groups.google.com/forum/#!forum/biogeobears). Minor updates may get posted first to the rexpokit GitHub repository (https://github.com/nmatzke/rexpokit)

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 in the package "inst" directory, 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, 2013) successfully compiled it using g++ and gfortran from (gcc version 4.2.1). (Update 2017-08-13: repeated with gccgfortran version 7.1.0.)

Citation

This code was developed for the following publications. Please cite if used:

Matzke, Nicholas J. (2014). "Model Selection in Historical Biogeography Reveals that Founder-event Speciation is a Crucial Process in Island Clades." *Systematic Biology*, 63(6), 951-970. http://dx.doi.org/10.1093/sysbio/syu056

Matzke, Nicholas J. (2013). "Probabilistic historical biogeography: new models for founder-event speciation, imperfect detection, and fossils allow improved accuracy and model-testing." *Frontiers of Biogeography*, 5(4), 242-248. http://escholarship.org/uc/item/44j7n141

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 inst/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. Revisions for version 0.26, which fixed many issues with warnings about obsolescence in F77 code, were aided by email help/discussions with: Kurt Hornik, Doug Nychka, Marcello Chiodi, Meabh McCurdy. Also, thanks to these incredibly helpful websites: "On-Line Fortran F77 - F90 Converter" (www.fortran.uk/plusfortonline.php), "Building and checking R source packages for Windows" (https://win-builder.r-project.org/), "Modernizing Old Fortran" (fortran-wiki.org/fortran/show/Modernizing+Old+Fortran), "Registering the C++ and FORTRAN calls" (stat.ethz.ch/pipermail/r-devel/2017-February/073755.html)

4. 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.

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

Python package "Pyprop":

http://code.google.com/p/pyprop/
(old URL) pyprop.googlecode.com/svn/trunk/core/krylov/expokit/expokitpropagator.cpp
(old URL) www.koders.com/python/fidCA95B5A4B2FB77455A72B8A361CF684FFE48F4DC.aspx?s=fourier+transform

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

6. 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 (see package "inst" directory)

7. EXPOKIT included some LAPACK and BLAS code for portability. This has been slightly modified to pass new CRAN checks and compilers. The original copyright is at: /inst/LAPACK_LICENSE.txt

8. itscale.f was copied from the R package "FD" in order to avoid an unnecessary dependency (and associated issues with compilation, updates, etc.). See R function "maxent" for more details, citations for package "FD":

Laliberte, E., and P. Legendre (2010) A distance-based framework for measuring functional diversity from multiple traits. *Ecology* 91:299-305.

Laliberte, E., Legendre, P., and B. Shipley. (2014). FD: measuring functional diversity from multiple traits, and other tools for functional ecology. R package version 1.0-12. https://CRAN.R-project.org/package=FD

Author(s)

Nicholas J. Matzke <nickmatzke.ncse@gmail.com>, Roger B. Sidje <roger.b.sidje@ua.edu>, Drew Schmidt <schmidt@math.utk.edu>

References

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

Matzke, Nicholas J. (2014). "Model Selection in Historical Biogeography Reveals that Founder-event Speciation is a Crucial Process in Island Clades." *Systematic Biology*, 63(6), 951-970. http://dx.doi.org/10.1093/sysbio/syu056

Matzke, Nicholas J. (2013). "Probabilistic historical biogeography: new models for founder-event speciation, imperfect detection, and fossils allow improved accuracy and model-testing." *Frontiers of Biogeography*, 5(4), 242-248. http://escholarship.org/uc/item/44j7n141

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://epubs.siam.org/doi/10.1137/S00361445024180>.

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 <nickmatzke.ncse@gmail.com>

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)
```

---

expm *Simple Interface*

Description

A simple interface for matrix exponentiation, similar to that found in the Matrix package.

Usage

```
expm(x, t=1)
```

Arguments

x	A square matrix.
t	Time value; see details section below. Default value is 1.

Details

This routine computes exp(t*x), where t in a single numeric value, and x is a square matrix and exp refers to the matrix exponential. Formally, the matrix exponential is given by the power series:

$expm(x) = x/1! + x^2/2! + x^3/3! + \ldots$

where the powers on the matrix correspond to matrix-matrix multiplications.

expm() directly computes the matrix exponential of a dense matrix. The implementation uses an order 6 Pade' approximation with a scaling-and-squaring technique.

Value

The matrix exponential.

Author(s)

Drew Schmidt <schmidt@math.utk.edu>

References

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>.

Examples

```
# Example use:
x <- matrix(1:25, nrow=5)/100

expm(x)
```

`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 <nickmatzke.ncse@gmail.com> and Drew Schmidt <schmidt@math.utk.edu>

See Also

mat2coo

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_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 <nickmatzke.ncse@gmail.com> and Drew Schmidt <schmidt@math.utk.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 <nickmatzke.ncse@gmail.com> and Drew Schmidt <schmidt@math.utk.edu>

See Also

mat2coo

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_itscale5_wrapper

wrapper function for FORTRAN itscale5, for FD's maxent

Description

This function wraps the .C call to EXPOKIT for the itscale5 function.

Usage

```
expokit_itscale5_wrapper(SXT, ngroups, ntraits, const, prior,
  prob, entropy, niter, tol, denom)
```

Arguments

SXT	is a Groups (rows) X Traits (columns) matrix
ngroups	is an integer (nb: NJM's interpretation)
ntraits	is an integer (nb: NJM's interpretation)
const	is a vector of the constraint values (means, variances)
prior	is the prior distribution
prob	is the return vector of the maximum entropy
entropy	is the maximum entropy probabilities
niter	is the number of iterations required
tol	is the convergence tolerance value; tolerance is mean square difference
denom	are final moments

Details

The itscale5 function is in the "itscale5.f" FORTRAN file. itscale5 is used by the FD::maxent function.

The maxent function is used by BioGeoBEARS, merely to provide a simple method of putting flat or skewed probability distributions on the ordered categorical variable "size of smaller daughter range").

As the package FD has a number of other dependencies, some of which cause problems on some machines, I am just including maxent and itscale5 here, in order to avoid "dependency hell".

I am putting it in rexpokit rather than BioGeoBEARS, to make rexpokit the only package using FORTRAN code (which has a list of its own issues).

Value

res A list of outputs

Author(s)

Nicholas J. Matzke <nickmatzke.ncse@gmail.com>

See Also

maxent

Examples

```
# See maxent() function
test=1
```

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 <nickmatzke.ncse@gmail.com> and Drew Schmidt <schmidt@math.utk.edu>

See Also

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 <nickmatzke.ncse@gmail.com> and Drew Schmidt <schmidt@math.utk.edu>

See Also

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

---

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 <nickmatzke.ncse@gmail.com>

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 <nickmatzke.ncse@gmail.com>

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
```

`maxent` *Estimating Probabilities via Maximum Entropy: Improved Iterative Scaling*

Description

NOTE: This is a copy of the FD::maxent function, included in rexpokit to avoid the dependency on the package FD. `maxent` returns the probabilities that maximize the entropy conditional on a series of constraints that are linear in the features. It relies on the Improved Iterative Scaling algorithm of Della Pietra et al. (1997). It has been used to predict the relative abundances of a set of species given the trait values of each species and the community-aggregated trait values at a site (Shipley et al. 2006; Shipley 2009; Sonnier et al. 2009).

Usage

```
maxent(constr, states, prior, tol = 1e-07, lambda = FALSE)
```

Arguments

`constr`	vector of macroscopical constraints (e.g. community-aggregated trait values). Can also be a matrix or data frame, with constraints as columns and data sets (e.g. sites) as rows.
`states`	vector, matrix or data frame of states (columns) and their attributes (rows).
`prior`	vector, matrix or data frame of prior probabilities of states (columns). Can be missing, in which case a maximally uninformative prior is assumed (i.e. uniform distribution).
`tol`	tolerance threshold to determine convergence. See 'details' section.
`lambda`	Logical. Should λ-values be returned?

Details

This is a copy of the FD::maxent function, included in rexpokit to avoid the dependency on the package FD. Its authorship information is Authored by: Bill Shipley <bill.shipley@usherbrooke.ca> (http://pages.usherbrooke.ca/jshipley/recherche/); Ported to FD by Etienne Laliberte. It was copied to rexpokit by Nick Matzke (just in order to avoid the dependency on package "FD").

Having BioGeoBEARS depend on package "FD" was sometimes problematic, as it had a variety of FORTRAN code and dependencies that could slow/stall installation, particularly on older Windows machines or machines without appropriate compilers. The maxent function uses only the FORTRAN file itscale5.f, so that code was included in rexpokit, in order to include all of the FORTRAN code in a single package (greatly simplifying the compilation and code-review process for BioGeoBEARS, which is pure R.)

The function maxent is used in BioGeoBEARS only for the simple purpose of putting a probability distribution on the ordered variable "number of areas in the smaller daughter range" at cladogenesis. For example, if mx01v = 0.0001 (the DEC model default), then the smaller daughter range will have a 100 percent probability of being of size 1 area during a vicariance event (thus, the "v" in

"mx01v"). If mx01v = 0.5 (the DIVALIKE model default), then the smaller daughter range will have an equal chance of being any range of size less than the parent range. If mx01y = 0.9999 (the BAYAREALIKE default), then the "smaller" daughter at sympatry (mx01y, y is sYmpatry) will have 100 percent probability of being the same size as its sister (i.e., the same range as the sister, i.e. "perfect sympatry" or "sympatry across all areas").

Original description from FD::maxent follows for completeness, but is not relevant for rexpokit/BioGeoBEARS.

The biological model of community assembly through trait-based habitat filtering (Keddy 1992) has been translated mathematically via a maximum entropy (maxent) model by Shipley et al. (2006) and Shipley (2009). A maxent model contains three components: (i) a set of possible states and their attributes, (ii) a set of macroscopic empirical constraints, and (iii) a prior probability distribution $\mathbf{q} = [q_j]$.

In the context of community assembly, states are species, macroscopic empirical constraints are community-aggregated traits, and prior probabilities $\mathbf{q}$ are the relative abundances of species of the regional pool (Shipley et al. 2006, Shipley 2009). By default, these prior probabilities $\mathbf{q}$ are maximally uninformative (i.e. a uniform distribution), but can be specificied otherwise (Shipley 2009, Sonnier et al. 2009).

To facilitate the link between the biological model and the mathematical model, in the following description of the algorithm states are species and constraints are traits.

Note that if `constr` is a matrix or data frame containing several sets (rows), a maxent model is run on each individual set. In this case if `prior` is a vector, the same prior is used for each set. A different prior can also be specified for each set. In this case, the number of rows in `prior` must be equal to the number of rows in `constr`.

If $\mathbf{q}$ is not specified, set $p_j = 1/S$ for each of the S species (i.e. a uniform distribution), where p_j is the probability of species j, otherwise $p_j = q_j$.

Calulate a vector $\mathbf{c} = [c_i] = \{c_1,\ c_2,\ \ldots,\ c_T\}$, where $c_i = \sum_{j=1}^{S} t_{ij}$; i.e. each c_i is the sum of the values of trait i over all species, and T is the number of traits.

Repeat for each iteration k until convergence:

1. For each trait t_i (i.e. row of the constraint matrix) calculate:

$$\gamma_i(k) = ln\left(\frac{\bar{t}_i}{\sum_{j=1}^{S}\left(p_j(k)\ t_{ij}\right)}\right)\left(\frac{1}{c_i}\right)$$

This is simply the natural log of the known community-aggregated trait value to the calculated community-aggregated trait value at this step in the iteration, given the current values of the probabilities. The whole thing is divided by the sum of the known values of the trait over all species.

2. Calculate the normalization term Z:

$$Z(k) = \left(\sum_{j=1}^{S} p_j(k)\ e^{\left(\sum_{i=1}^{T}\gamma_i(k)\ t_{ij}\right)}\right)$$

3. Calculate the new probabilities p_j of each species at iteration $k+1$:

$$p_j(k+1) = \frac{p_j(k)\ e^{\left(\sum_{i=1}^{T} \gamma_i(k)\ t_{ij}\right)}}{Z(k)}$$

4. If $|max\left(p\left(k+1\right) - p\left(k\right)\right)| \leq$ tolerance threshold (i.e. argument `tol`) then stop, else repeat steps 1 to 3.

When convergence is achieved then the resulting probabilities ($\hat{p}_j$) are those that are as close as possible to q_j while simultaneously maximize the entropy conditional on the community-aggregated traits. The solution to this problem is the Gibbs distribution:

$$\hat{p}_j = \frac{q_j e^{\left(-\sum_{i=1}^{T} \lambda_i t_{ij}\right)}}{\sum_{j=1}^{S} q_j e^{\left(-\sum_{i=1}^{T} \lambda_i t_{ij}\right)}} = \frac{q_j e^{\left(-\sum_{i=1}^{T} \lambda_i t_{ij}\right)}}{Z}$$

This means that one can solve for the Langrange multipliers (i.e. weights on the traits, λ_i) by solving the linear system of equations:

$$\begin{pmatrix} ln\left(\hat{p}_1\right) \\ ln\left(\hat{p}_2\right) \\ \vdots \\ ln\left(\hat{p}_S\right) \end{pmatrix} = (\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_T) \begin{bmatrix} t_{11} & t_{12} & \ldots & t_{1S} - ln(Z) \\ t_{21} & t_{22} & \vdots & t_{2S} - ln(Z) \\ \vdots & \vdots & \vdots & \vdots \\ t_{T1} & t_{T2} & \ldots & t_{TS} - ln(Z) \end{bmatrix} - ln(Z)$$

This system of linear equations has $T+1$ unknowns (the T values of λ plus $ln(Z)$) and S equations. So long as the number of traits is less than $S-1$, this system is soluble. In fact, the solution is the well-known least squares regression: simply regress the values $ln(\hat{p}_j)$ of each species on the trait values of each species in a multiple regression.

The intercept is the value of $ln(Z)$ and the slopes are the values of λ_i and these slopes (Lagrange multipliers) measure by how much the $ln(\hat{p}_j)$, i.e. the ln(relative abundances), changes as the value of the trait changes.

`FD::maxent.test` provides permutation tests for maxent models (Shipley 2010).

Value

`prob`	vector of predicted probabilities
`moments`	vector of final moments
`entropy`	Shannon entropy of `prob`
`iter`	number of iterations required to reach convergence
`lambda`	λ-values, only returned if `lambda = T`
`constr`	macroscopical constraints
`states`	states and their attributes
`prior`	prior probabilities

Author(s)

Bill Shipley <bill.shipley@usherbrooke.ca>, original URL: pages.usherbrooke.ca/jshipley/recherche/

Ported to **FD** by Etienne Laliberte.

References

Della Pietra, S., V. Della Pietra, and J. Lafferty (1997) Inducing features of random fields. *IEEE Transactions Pattern Analysis and Machine Intelligence* **19**:1-13.

Keddy, P. A. (1992) Assembly and response rules: two goals for predictive community ecology. *Journal of Vegetation Science* **3**:157-164.

Shipley, B., D. Vile, and E. Garnier (2006) From plant traits to plant communities: a statistical mechanistic approach to biodiversity. *Science* **314**: 812–814.

Shipley, B. (2009) From Plant Traits to Vegetation Structure: Chance and Selection in the Assembly of Ecological Communities. Cambridge University Press, Cambridge, UK. 290 pages.

Shipley, B. (2010) Inferential permutation tests for maximum entropy models in ecology. *Ecology* **in press**.

Sonnier, G., Shipley, B., and M. L. Navas. 2009. Plant traits, species pools and the prediction of relative abundance in plant communities: a maximum entropy approach. *Journal of Vegetation Science* **in press**.

See Also

`FD::functcomp` to compute community-aggregated traits, and `FD::maxent.test` for the permutation tests proposed by Shipley (2010).

Another faster version of `maxent` for multicore processors called `maxentMC` is available from Etienne Laliberte (<etiennelaliberte@gmail.com>). It's exactly the same as `maxent` but makes use of the **multicore**, **doMC**, and **foreach** packages. Because of this, `maxentMC` only works on POSIX-compliant OS's (essentially anything but Windows).

Examples

```
# an unbiased 6-sided dice, with mean = 3.5
# what is the probability associated with each side,
# given this constraint?
maxent(3.5, 1:6)

# a biased 6-sided dice, with mean = 4
maxent(4, 1:6)
```

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_REXPOKIT_coo_fmt <- mat2coo(tmpmat)
```

Index