Package ‘sparseinv’

August 23, 2018

Type Package
Title Computation of the Sparse Inverse Subset
Version 0.1.3
Date 2018-08-23
Maintainer Andrew Zammit-Mangion <andrewzm@gmail.com>
Suggests covr, testthat
Imports Matrix, methods, Rcpp, spam
Description Creates a wrapper for the 'SuiteSparse' routines that execute the Takahashi equations. These equations compute the elements of the inverse of a sparse matrix at locations where the its Cholesky factor is structurally non-zero. The resulting matrix is known as a sparse inverse subset. Some helper functions are also implemented. Support for spam matrices is currently limited and will be implemented in the future. See Rue and Martino (2007) <doi:10.1016/j.jspi.2006.07.016> and Zammit-Mangion and Rougier (2018) <doi:10.1016/j.csda.2018.02.001> for the application of these equations to statistics.
Depends R (>= 3.1)
License GPL (>= 2.1)
NeedsCompilation yes
LazyData true
RoxygenNote 6.0.1
LinkingTo Rcpp
Author Andrew Zammit-Mangion [aut, cre], Timothy Davis [ctb], Patrick Amestoy [ctb], Iain Duff [ctb], John K. Reid [ctb]
Repository CRAN
Date/Publication 2018-08-23 04:50:03 UTC
Description

This package creates a wrapper for the SuiteSparse routines in C that use the Takahashi equations to compute the elements of the inverse of a sparse matrix at locations where the (permuted) Cholesky factor is structurally non-zero. The resulting matrix is known as a sparse inverse subset. Some helper functions (like the permuted Cholesky factorisation) are also implemented. Support for spam matrices is currently limited and will be implemented in the future.

Usage

cholPermute(Q)

Arguments

Q precision matrix of class matrix, Matrix (column-compressed, i.e., dgCMatrix or dsCMatrix), or spam

Value

A list with two elements, Qpermchol (the permuted Cholesky factor) and P (the permutation matrix) of class Matrix. Note that spam matrices are not returned to comply with the Takahashi_Davis function which requires objects of class Matrix.
cholsolve

References


Examples

```r
require(Matrix)
choiPermute(sparseMatrix(i = c(1, 1, 2, 2),
j = c(1, 2, 1, 2),
x = c(0.1, 0.2, 0.2, 1)))
```

cholsolve

Solve the equation $Qx = y$

Description

This function is similar to `solve(Q, y)` but with the added benefit that it allows for permuted matrices. This function does the job in order to minimise user error when attempting to re-permute the matrices prior or after solving. The user also has an option for the permuted Cholesky factorisation of Q to be carried out internally.

Usage

```r
cholsolve(Q = NULL, y = NULL, perm = FALSE, cholQ = NULL,
        cholQp = NULL, P = NULL)
```

Arguments

- **Q**: matrix (if of class `Matrix` needs to be column-compressed, i.e., `dgCMatrix` or `dsCMatrix`), the Cholesky factor of which needs to be found
- **y**: matrix with the same number of rows as Q
- **perm**: if FALSE no permutation is carried out, if TRUE permuted Cholesky factors are used
- **cholQ**: the lower Cholesky factor of Q (if known already)
- **cholQp**: the lower Cholesky factor of a permuted Q (if known already)
- **P**: the permutation matrix (if known already)

Value

- **x**: solution to $Qx = y$

References

cholsolveAQinvAT

Solve the equation $X = A Q^{-1} t(A)$ under permutations

Examples

```r
require(Matrix)
Q = sparseMatrix(i = c(1, 1, 2, 2),
    j = c(1, 2, 1, 2),
    x = c(0.1, 0.2, 0.2, 1))
y = matrix(c(1, 2), 2, 1)
cholsolve(Q, y)
```

Description

This function is a wrapper of `solve()` for finding $X = A Q^{-1} t(A)$ when the permuted Cholesky factor of $Q$ is known.

Usage

```r
cholsolveAQinvAT(Q = NULL, A = NULL, Lp = NULL, P = NULL)
```

Arguments

- **Q**: matrix (if of class `Matrix` needs to be column-compressed, i.e., `dgCMatrix` or `dsCMatrix`), the Cholesky factor of which needs to be found
- **A**: sparse or dense matrix
- **Lp**: the lower Cholesky factor of a permuted $Q$
- **P**: the permutation matrix

Value

$x$ solution to $X = A Q^{-1} t(A)$

References


Examples

```r
require(Matrix)
Q = sparseMatrix(i = c(1, 1, 2, 2),
    j = c(1, 2, 1, 2),
    x = c(0.1, 0.2, 0.2, 1))
X <- cholPermute(Q)
y <- matrix(c(1, 2), 2, 1)
A <- y %*% t(y)
cholsolveAQinvAT(Q, A, X, permchol, X, P)
```
densify

Densify with explicit zeroes

Description

This function takes two sparse matrices and returns the first matrix padded with explicit zeros so that it is at least dense (probably denser) than the second matrix. This function only works with matrices of class Matrix.

Usage

densify(A, B)

Arguments

A  object of class Matrix
B  object of class Matrix

Value

object of class Matrix

Examples

require(Matrix)
Q1 <- sparseMatrix(i = c(1, 2, 2), j = c(1, 1, 2), x = c(0.1, 0.2, 1))
Q2 <- sparseMatrix(i = c(1, 1, 2, 2), j = c(1, 2, 1, 2), x = c(0.1, 0.3, 0.2, 1))
Q1dens <- densify(Q1, Q2)
Q1
class(Q1)
Q1dens
class(Q1dens)

symb

Return the symbolic representation of a Matrix

Description

This function takes an object of class Matrix and returns the same Matrix with all elements replaced with 1.

Usage

symb(A)

Arguments

A  object of class Matrix
Value

object of class Matrix

Examples

```
require(Matrix)
Q = sparseMatrix(i = c(1L, 1L, 2L, 2L),
                j = c(1L, 2L, 1L, 2L),
                x = c(0.1L, 0.2L, 0.2L, 1L))
Qsymb <- symb(Q)
Qsymb
```

Description

Computes the sparse inverse subset of a sparse matrix \( Q \) using the Takahashi equations.

Usage

```
Takahashi_Davis(Q = NULL, cholQp = NULL, return_perm_chol = 0, P = 0, gc = 0)
```

Arguments

- **Q**: precision matrix of class matrix, Matrix (column-compressed, i.e., dgCMatrix or dsCMatrix), or spam
- **cholQp**: the Cholesky factor of class dtCMatrix of the permuted \( Q \) (if known already). If both \( Q \) and \( cholQp \) are specified, \( Q \) is ignored
- **return_perm_chol**: if 1, the Cholesky factor of the permuted \( Q \) is returned
- **P**: the permutation matrix of class dgCMatrix (if known already)
- **gc**: do garbage collection throughout (may increase computational time but useful for small memory machines)

Details

This function first computes the Cholesky factor of \( Q \). The fill-in reduction permutation is the approximate minimum degree permutation (amd) of Timothy Davis’ SuiteSparse package configured to be slightly more aggressive than that in the Matrix package. The function then uses the Takahashi equations to compute the variances at the non-zero locations in the Cholesky factor from the factor itself. The equations themselves are implemented in C using the SparseSuite package of Timothy Davis.
**Value**

if return_perm_chol == 0, the sparse inverse subset of Q is returned, where the non-zero elements correspond to those in the Cholesky factor of its permutation. If !(return_perm_chol == 0), a list with three elements is returned: S (the sparse inverse subset), Lp (the Cholesky factor of the permuted matrix) and P (the permutation matrix)

**Note**

This package is a wrapper for C functions implemented by Timothy Davis in SuiteSparse. The author of this package has done no work on the sparse inverse routines themselves and any acknowledgment should include one to SuiteSparse (see below for reference). The author of this package was made aware of this methodology by Botond Cseke.

**References**


**Examples**

```r
require(matrix)
Q = sparseMatrix(i = c(1, 1, 2, 2),
    j = c(1, 2, 1, 2),
    x = c(0.1, 0.2, 0.2, 1))
X <- cholPermute(Q)
S_partial = Takahashi_Davis(Q, cholQp = X$Qpermchol, P = X$P)
```
Index

*Topic **Cholesky**
  - cholPermute, 2
  - cholsolve, 3
  - cholsolveAQinvAT, 4
  - Takahashi_Davis, 6

*Topic **factor**,
  - cholsolve, 3
  - cholsolveAQinvAT, 4
  - Takahashi_Davis, 6

*Topic **factor**
  - cholPermute, 2

*Topic **inverse**
  - Takahashi_Davis, 6

*Topic **linear**
  - cholsolve, 3
  - cholsolveAQinvAT, 4

*Topic **solve**
  - cholsolve, 3
  - cholsolveAQinvAT, 4

*Topic **sparse**
  - Takahashi_Davis, 6

*Topic **subset**
  - Takahashi_Davis, 6

  cholPermute, 2
  cholsolve, 3
  cholsolveAQinvAT, 4

  densify, 5

  sparseinv-package, 2
  symb, 5

  Takahashi_Davis, 6