Package 'WoodburyMatrix'

August 10, 2020

Title Fast Matrix Operations via the Woodbury Matrix Identity

Version 0.0.1

Description A hierarchy of classes and methods for manipulating matrices formed implicitly from the sums of the inverses of other matrices, a situation commonly encountered in spatial statistics and related fields. Enables easy use of the Woodbury matrix identity and the matrix determinant lemma to allow computation (e.g., solving linear systems) without having to form the actual matrix. More information on the underlying linear algebra can be found in Harville, D. A. (1997) <doi:10.1007/b98818>.

URL https://github.com/mbertolacci/WoodburyMatrix

BugReports https://github.com/mbertolacci/WoodburyMatrix/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports Matrix, methods

Suggests covr, lintr, testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Michael Bertolacci [aut, cre, cph]
(<https://orcid.org/0000-0003-0317-5941>)

Maintainer Michael Bertolacci <m.bertolacci@gmail.com>

Repository CRAN

Date/Publication 2020-08-10 12:30:02 UTC

R topics documented:

`determinant,WoodburyMatrix,logical-method` *Calculate the determinant of a WoodburyMatrix object*

Description

Calculates the (log) determinant of a `WoodburyMatrix` using the matrix determinant lemma.

Usage

```
## S4 method for signature 'WoodburyMatrix,logical'
determinant(x, logarithm)
```

Arguments

`x`	A object that is a subclass of `WoodburyMatrix`
`logarithm`	Logical indicating whether to return the logarithm of the matrix.

Value

Same as base::determinant.

See Also

base::determinant

`instantiate` *Instantiate a matrix*

Description

This is a generic to represent direct instantiation of an implicitly defined matrix. In general this is a bad idea, because it removes the whole purpose of using an implicit representation.

Usage

```
instantiate(x)

## S4 method for signature 'GWoodburyMatrix'
instantiate(x)

## S4 method for signature 'SWoodburyMatrix'
instantiate(x)
```

Arguments

x	Implicit matrix to directly instantiate.

Value

The directly instantiated matrix.

Functions

- `instantiate,GWoodburyMatrix-method`: Method for general matrices.
- `instantiate,SWoodburyMatrix-method`: Method for symmetric matrices.

See Also

WoodburyMatrix, WoodburyMatrix

---

mahalanobis *Mahalanobis distance*

Description

Generic for computing the squared Mahalanobis distance.

Usage

```
mahalanobis(x, center, cov, inverted = FALSE, ...)

## S4 method for signature 'ANY,ANY,SWoodburyMatrix'
mahalanobis(x, center, cov, inverted = FALSE, ...)
```

Arguments

x	Numeric vector or matrix
center	Numeric vector representing the mean; if omitted, defaults to zero mean
cov	Covariance matrix
inverted	Whether to treat cov as a precision matrix; must be FALSE for SWoodburyMatrix objects.
...	Passed to the Cholesky function.

Methods (by class)

- x = ANY,center = ANY,cov = SWoodburyMatrix: Use the Woodbury matrix identity to compute the squared Mahalanobis distance with the implicit matrix as the covariance.

See Also

mahalanobis

normal-distribution-methods
Normal distribution methods for SWoodburyMatrix *objects*

Description

Draw samples and compute density functions for the multivariate normal distribution with an SWoodburyMatrix object as its covariance matrix.

Usage

```
dwnorm(x, mean, covariance, log = FALSE)

rwnorm(n, mean, covariance)
```

Arguments

x	A numeric vector or matrix.
mean	Optional mean vector; defaults to zero mean.
covariance	WoodburyMatrix object.
log	Logical indicating whether to return log of density.
n	Number of samples to return. If n = 1, returns a vector, otherwise returns an n by nrow(W) matrix.

Functions

- dwnorm: Compute the density of the distribution
- rwnorm: Draw samples from the distribution

See Also

WoodburyMatrix

Examples

```
library(Matrix)
# Trivial example with diagonal covariance matrices
W <- WoodburyMatrix(Diagonal(10), Diagonal(10))
x <- rwnorm(10, covariance = W)
print(dwnorm(x, covariance = W, log = TRUE))
```

solve-methods *Solve methods for* WoodburyMatrix *objects*

Description

Methods based on solve to solve a linear system of equations involving WoodburyMatrix objects. These methods take advantage of the Woodbury matrix identity and therefore can be much more time and memory efficient than forming the matrix directly.

Calling this function while omitting the b argument returns the inverse of a. This is NOT recommended, since it removes any benefit from using an implicit representation of a.

Usage

```
## S4 method for signature 'GWoodburyMatrix,missing'
solve(a)

## S4 method for signature 'GWoodburyMatrix,ANY'
solve(a, b)

## S4 method for signature 'SWoodburyMatrix,missing'
solve(a)

## S4 method for signature 'SWoodburyMatrix,ANY'
solve(a, b)
```

Arguments

a	WoodburyMatrix object.
b	Matrix, vector, or similar (needs to be compatible with the submatrices a@A and a@V or a@X that define the WoodburyMatrix).

Value

The solution to the linear system, or the inverse of the matrix. The class of the return value will be a subclass of Matrix, with the specific subclass determined by a and b.

Functions

- solve,GWoodburyMatrix,missing-method: Invert the matrix
- solve,GWoodburyMatrix,ANY-method: Solve the linear system
- solve,SWoodburyMatrix,missing-method: Invert the symmetric matrix
- solve,SWoodburyMatrix,ANY-method: Solve the linear system

See Also

WoodburyMatrix, WoodburyMatrix

`WoodburyMatrix` *Create a Woodbury matrix identity matrix*

Description

Creates an implicitly defined matrix representing the equation

$$A^{-1} + UB^{-1}V,$$

where A, U, B and V are n x n, n x p, p x p and p x n matrices, respectively. A symmetric special case is also possible with

$$A^{-1} + XB^{-1}X',$$

where X is n x p and A and B are additionally symmetric. The available methods are described in WoodburyMatrix-class and in solve. Multiple B / U / V / X matrices are also supported; see below

Usage

```
WoodburyMatrix(A, B, U, V, X, O, symmetric)
```

Arguments

`A`	Matrix A in the definition above.
`B`	Matrix B in the definition above, or list of matrices.
`U`	Matrix U in the definition above, or list of matrices. Defaults to a diagonal matrix/matrices.
`V`	Matrix V in the definition above, or list of matrices. Defaults to a diagonal matrix/matrices.
`X`	Matrix X in the definition above, or list of matrices. Defaults to a diagonal matrix/matrices.
`O`	Optional, precomputed value of O, as defined above. THIS IS NOT CHECKED FOR CORRECTNESS, and this argument is only provided for advanced users who have precomputed the matrix for other purposes.
`symmetric`	Logical value, whether to create a symmetric or general matrix. See Details section for more information.

Details

The benefit of using an implicit representation is that the inverse of this matrix can be efficiently calculated via

$$A - AUO^{-1}VA$$

where $O = B + VAU$, and its determinant by

$$det(O)det(A)^{-1}det(B)^{-1}.$$

These relationships are often called the Woodbury matrix identity and the matrix determinant lemma, respectively. If A and B are sparse or otherwise easy to deal with, and/or when $p < n$,

manipulating the matrices via these relationships rather than forming W directly can have huge advantageous because it avoids having to create the (typically dense) matrix

$$A^{-1} + UB^{-1}V$$

directly.

Value

A GWoodburyMatrix object for a non-symmetric matrix, SWoodburyMatrix for a symmetric matrix.

Symmetric form

Where applicable, it's worth using the symmetric form of the matrix. This takes advantage of the symmetry where possible to speed up operations, takes less memory, and sometimes has numerical benefits. This function will create the symmetric form in the following circumstances:

- `symmetry = TRUE`; or
- the argument `X` is provided; or
- `A` and `B` are symmetric (according to `isSymmetric`) and the arguments `U` and `V` are NOT provided.

Multiple B matrices

A more general form allows for multiple B matrices:

$$A^{-1} + \sum_{i=1}^{n} U_i B_i^{-1} V_i,$$

and analogously for the symmetric form. You can use this form by providing a list of matrices as the `B` (or `U`, `V` or `X`) arguments. Internally, this is implemented by converting to the standard form by letting `B = bdiag(...the B matrices...)`, `U = cbind(..the U matrices...)`, and so on.

The `B`, `U`, `V` and `X` values are recycled to the length of the longest list, so you can, for instance, provide multiple B matrices but only one U matrix (and vice-versa).

References

More information on the underlying linear algebra can be found in Harville, D. A. (1997) <doi:10.1007/b98818>.

See Also

WoodburyMatrix, solve, instantiate

Examples

```
library(Matrix)
# Example solving a linear system with general matrices
A <- Diagonal(100)
B <- rsparsematrix(100, 100, 0.5)
W <- WoodburyMatrix(A, B)
```

```
str(solve(W, rnorm(100)))

# Calculating the determinant of a symmetric system
A <- Diagonal(100)
B <- rsparsematrix(100, 100, 0.5, symmetric = TRUE)
W <- WoodburyMatrix(A, B, symmetric = TRUE)
print(determinant(W))

# Having a lower rank B matrix and an X matrix
A <- Diagonal(100)
B <- rsparsematrix(10, 10, 1, symmetric = TRUE)
X <- rsparsematrix(100, 10, 1)
W <- WoodburyMatrix(A, B, X = X)
str(solve(W, rnorm(100)))

# Multiple B matrices
A <- Diagonal(100)
B1 <- rsparsematrix(100, 100, 1, symmetric = TRUE)
B2 <- rsparsematrix(100, 100, 1, symmetric = TRUE)
W <- WoodburyMatrix(A, B = list(B1, B2))
str(solve(W, rnorm(100)))
```

`WoodburyMatrix-class` *Virtual class for Woodbury identity matrices*

Description

The `WoodburyMatrix` is a virtual class, contained by both `GWoodburyMatrix` (for general matrices) and `SWoodburyMatrix` (for symmetric matrices). See WoodburyMatrix for construction of these classes. The methods available for these classes are described below; see also the solve methods. This class is itself a subclass of `Matrix`, so basic matrix methods like `nrow`, `ncol`, `dim` and so on also work.

Usage

```
## S4 method for signature 'GWoodburyMatrix'
isSymmetric(object)

## S4 method for signature 'SWoodburyMatrix'
isSymmetric(object)

## S4 method for signature 'GWoodburyMatrix,ANY'
x %*% y

## S4 method for signature 'SWoodburyMatrix,ANY'
x %*% y

## S4 method for signature 'GWoodburyMatrix'
t(x)
```

```
## S4 method for signature 'SWoodburyMatrix'
t(x)
```

Arguments

object	WoodburyMatrix object
x	WoodburyMatrix object
y	Matrix or vector

Functions

- GWoodburyMatrix-class: Sub-class representing a generic matrix.
- SWoodburyMatrix-class: Sub-class representing a symmetric matrix. Also subclasses symmetricMatrix.
- isSymmetric,GWoodburyMatrix-method: Check for symmetry of matrix; always returns FALSE.
- isSymmetric,SWoodburyMatrix-method: Check for symmetry of matrix; always returns TRUE.
- %*%,GWoodburyMatrix,ANY-method: Matrix multiplication (generally fast and
- %*%,SWoodburyMatrix,ANY-method: Matrix multiplication (generally fast and
- t,GWoodburyMatrix-method: Return the transpose of the matrix as another GWoodburyMatrix.
- t,SWoodburyMatrix-method: Does nothing, just returns x.

Slots

A n x n subclass of Matrix (GWoodburyMatrix) or symmetricMatrix (SWoodburyMatrix).

B p x p subclass of Matrix (GWoodburyMatrix) or symmetricMatrix (SWoodburyMatrix).

U n x p subclass of Matrix (only for

V p x m subclass of Matrix (only for

X n x p subclass of Matrix (only for

O p x p subclass of Matrix

See Also

WoodburyMatrix for object construction, Matrix (the parent of this class).

Index