Package ‘spdm’

February 3, 2019

Type Package

Title Functions for working with symmetric positive-definite matrices

Version 1.0.0

Author Corson N. Areshenkoff

Maintainer Corson N. Areshenkoff <areshenk@protonmail.com>

Description Means, estimation, and operations for symmetric positive-definite matrices.

Depends R (>= 3.0.2),

Imports expm, huge, nlshrink, kernlab, geigen

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

is.spd	*Check if a matrix is symmetric and positive definite*

Description

Check if a matrix is symmetric and positive definite

Usage

```
is.spd(..., tol = 1e-08)
```

Arguments

`...`	One or more numeric matrices
`tol`	Numerical tolerance for checking symmetry and positive definiteness

Details

Function checks whether the matrix `x` is symmetric and positive definite. Symmetry is evaluated up to an entrywise tolerance of `tol`, so that differences smaller than `tol` are ignored. Positive-definiteness is checked by computing the eigenvalues of `x` using `eigen`, setting eigenvalues smaller than `tol` in absolute value to zero, and then checking whether any are less than or equal to zero.

Value

A Boolean value indicating whether the matrix is symmetric and positive definite

spd.correlation	*Compute (partial) correlations*

Description

Transforms an SPD matrix into a matrix of partial correlations

Usage

```
spd.correlation(x, method = "correlation")
```

Arguments

`x`	An SPD matrix to be whitened
`method`	A string specifying either "correlation" (default) orf "partial" correlation.

Value

A symmetric, positive-definite matrix.

`spd.dist` *Distance between two symmetric, positive-definite matrices*

Description

Function implements several distance measures between symmetric, positive-definite matrices.

Usage

```
spd.dist(x, y, method = "euclidean", ...)
```

Arguments

`x, y`	Symmetric, positive-definite matrices
`method`	The distance measure. See details.

Details

Allowable distance measures are

- "euclidean": The Frobenius norm of the difference `x-y`.
- "logeuclidean": The Frobenius norm of the difference `log(x)-log(y)` in the tangent space..
- "cholesky": The Frobenius norm of the difference between the cholesky factors of `x` and `y`. Not affinely invariant.
- "riemannian": The Riemmanian distance proposed by Barachant, et al. (2013)
- "stein": The square root of Jensen;Bregman Log determinant divergence

`spd.estimate` *Covariance estimation*

Description

Function implements several forms of covariance estimation.

Usage

```
spd.estimate(x, method = "linshrink", ...)
```

Arguments

`x`	A data matrix, where rows are observations and columns are variables
`method`	Method of covariance estimation. See details
`...`	Additional arguments passed toe stimation functions. See details.

Details

Allowable estimation methods are:

- "sample": The ordinary sample covariance. Generally a poor choice in anything but very low dimensional settings, and is not guaranteed to be positive-definite.
- "linshrink": Linear shrinkage estimator proposed by Ledoit and Wolf (2004)
- "nlshrink": Non-linear shrinkage estimator proposed by Ledoit and Wolf (2012)
- "glasso": Graphical lasso (glasso) estimation using the huge package. Typically generates sparse estimates.

Additional arguments may be passed to the functions which perform estimation. Specifically:

- "sample": Uses `cov(x, ...)`
- "linshrink": Uses `nlshrink::linshrink_cov(x, ...)`
- "nlshrink": Uses `nlshrink::nlshrink_cov(x, ...)`
- "glasso": Uses `huge(x, method = 'glasso', cov.output = T, ...)` followed by `huge.select(x, ...)`. Note that `method` cannot be overridden, as other estimation methods do not return covariance estimates. Additional arguments to `huge()` or `huge.select()` should be prepended with `huge.` or `select.`, respectively.

In all cases, function will generate a warning if the estimated matrix is not positive definite.

Value

A covariance matrix

`spd.expmap` *Projection from the tangent space*

Description

Function projects a tangent vector x (a symmetric matrix) at a point p onto the space of SPD matrices.

Usage

```
spd.expmap(x, p = NULL)
```

Arguments

`x`	A symmetric matrix
`p`	The point to whose tangent space x belongs

Value

A symmetric, positive-definite matrix.

spd.interpolate — *Interpolation between two symmetric, positive-definite matrices*

Description

Function smoothly interpolates between two SPD matrices `x` and `y`. The interpolation is parametrized by `t`, where `t=0` returns `x`, `t=1` returns `y`, and `t=.5` returns the mean of `x` and `y`

Usage

```
spd.interpolate(x, y, t, method = "euclidean", ...)
```

Arguments

`x, y`	Symmetric, positive-definite matrices
`t`	Interpolation parameter in `[0,infinity)`
`method`	Type of interpolation (see details)

Details

Allowable distance measures are

- "euclidean": Euclidean interpolation `(1-t)x + ty`
- "logeuclidean": Euclidean interpolation in the tangent space.
- "riemannian": Interpolation along the geodesic path from `x` to `y`

spd.logmap — *Projection onto the tangent space*

Description

Function projects an SPD matrix x onto the tangent space at a point p

Usage

```
spd.logmap(x, p = NULL)
```

Arguments

`x`	A symmetric positive definite matrix
`p`	The point on whose tangent space to project x

Value

A symmetric matrix.

`spd.mean` *Compute the mean of a set of spd matrices*

Description

Function computes the mean of a set of symmetric positive-definite matrices. Several methods are implemented, as described in `Details`.

Usage

```
spd.mean(x, method = "euclidean", ...)
```

Arguments

`x`	A list of symmetric, positive-definite matrices
`method`	The type of mean to compute. See details
`...`	Further arguments. See details

Details

Function computes the mean of a set of symmetrix, positive-definite matrices. Several methods are implemented:

- "euclidean": The ordinary arithmetic mean – the sum of the matrices in `x`, divided by the number of matrices. This is guaranteed to be an spd matrix, but does not necessarily preserve the spectral characteristics of the individual matrices.
- "logeuclidean": Computed by taking the arithmetic mean of the logarithms of the matrices in `x`, and then projecting back onto the space of spd matrices. In general, better behaved than the arithmetic mean.
- "riemannian": The Riemmanian p-mean. Smoothly interpolates between the harmonic mean at `p = -1` to the geometric mean at `p = 0` to the arithmetic mean at `p = 1`. Is approximated iteratively using the fixed point algorithm described by Congedo, Barachant and Koopaei (2017). Requies a parameter `p` in the interval `[-1,1]` specifying the type of mean, a maximum error tolerance `tol` (default .01), and a maximum number of iterations (default 50). The case `p = 0` is approximated by computing the p-means at -.1 and .1, and then returning the midpoint between them using `spd.interpolate(..., method = 'riemannian')`

Value

The mean of the matrices in `x`.

spd.pca — *Kernel pca for SPD matrices*

Description

Function performs kernel principal component analysis on a set of symmetric, positive-definite matrices using an rbf kernel: `exp(-sigma * d(i,j)^2)`, where `d(i,j)` is a distance function implemented by `spd.dist`. This function is more or less a wrapper around the kernlab function `kpca`.

Usage

```
spd.pca(x, method = "euclidean", sigma = 1, ...)
```

Arguments

`x`	A list of symmetric, positive-definite matrices
`method`	The type of distance. See `spd.dist`, but also see details. Defaults to "logeuclidean".
`sigma`	scale parameter for rbf kernel
`...`	Further arguments for kernlab::kpca.

Details

Function performs kpca using a rbf kernel, where the distance between two inputs is given by `method`. Note that only "euclidean" and "logeuclidean" have been proven to give rise to positive-definite kernels for all values of sigma, although any distance implemented in `spd.dist` may be used. Anecdotally, `method = "riemannian"` often achieves superior performance.

Value

An S4 object of class kpca.

spd.pca.predict — *Prediction for SPD kernel PCA*

Description

Function accepts a fitted kpca object returned by `spd.pca` and a list of SPD matrices and returns a matrix of estimated principal component scores.

Usage

```
spd.pca.predict(fit, x)
```

Arguments

`fit`	An object of class kpca return by spd.pca
`x`	A list of SPD matrices for which to derive component scores.

Value

A matrix of principal component scores

`spd.transport` *Parallel transport of a tangent vector*

Description

Parallel transports a vector `x` in the tangent space at `"from"` to the tangent space at `"to"` using the Schild's ladder algorithm.

Usage

```
spd.transport(x, to, from, nsteps = 10)
```

Arguments

`x`	A tangent vector (symmetric matrix)
`to`	The SPD matrix to whose tangent space to move x
`from`	The SPD matrix to whose tangent space x belongs
`nsteps`	The number of steps in the geodesic connecting to and from

Value

A symmetric matrix – a tangent vector at to.

`spd.vectorize` *Vectorize and unvectorize a matrix*

Description

Function converts between a square symmetric matrix and a vector of the lower triangular elements + diagonal. Allows optional scaling of the off-diagonal entries in order to preserve the norm.

Usage

```
spd.vectorize(x, scaling = F)
```

Arguments

`x`	Either a square matrix, or a vector contains the lower triangular elements of x (including the diagonal elements)
`scaling`	If TRUE, scales the off diagonal elements by sqrt(2) to preserve the norm of the resulting vector

Details

If input is a square matrix, converts the lower triangular elements (including the diagonal) into a numeric vector. If input is a numeric vector, converts the input to a symmetric matrix. Note that, if the input is a vector, its length must be a triangular number.

`spd.whiten` *Covariance whitening transform*

Description

Whitens an SPD matrix using the procedure advocated by Ng, et al. (2014)

Usage

```
spd.whiten(x, p, unwhiten = F)
```

Arguments

`x`	An SPD matrix to be whitened
`p`	The SPD baseline whitening matrix
`unwhiten`	Logical. If TRUE, reverses a previously applied whitening transform.

Value

A symmetric, positive-definite matrix.

Index