Package: ksm (via r-universe)

May 23, 2026

Type Package

Title Kernel Density Estimation for Random Symmetric Positive Definite Matrices

Version 1.1

Description Kernel smoothing for Wishart random matrices described in Daayeb, Khardani and Ouimet (2025) <doi:10.48550/arXiv.2506.08816>, Gaussian and log-Gaussian models using least square or likelihood cross validation criteria for optimal bandwidth selection.

BugReports `https://github.com/lbelzile/ksm/issues`

Imports Rcpp (>= 1.0.12)

Suggests cubature, tinytest

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

License MIT + file LICENSE

RoxygenNote 7.3.3

LazyData true

Depends R (>= 2.10)

Repository https://lbelzile.r-universe.dev

Date/Publication 2025-11-24 10:17:54 UTC

RemoteUrl https://github.com/lbelzile/ksm

RemoteRef HEAD

RemoteSha b14ebb612ee3f3faf8c48ad2d727533845fd48af

Contents

`bandwidth_optim` *Bandwidth optimization for symmetric matrix kernels*

Description

Given a sample of positive definite matrices, perform numerical maximization of the `h`-block least square (`lscv`) or leave-one-out likelihood (`lcv`) cross-validation criteria using a root search.

Usage

```
bandwidth_optim(
  x,
  criterion = c("lscv", "lcv"),
  kernel = c("Wishart", "smlnorm", "smnorm"),
  tol = 1e-04,
  h = 1L,
  bounds = c(1e-04, 10)
)
```

Arguments

`x`	sample of symmetric matrix observations from which to build the kernel density kernel
`criterion`	optimization criterion, one of `lscv` for least square cross-validation at lag `h` or `lcv` for leave-one-out cross-validation.
`kernel`	string, one of `Wishart`, `smlnorm` (log-Gaussian) or `smnorm` (Gaussian).
`tol`	double, tolerance of optimization (root search)
`h`	lag step for consideration of observations, for the case `criterion=lscv`
`bounds`	vector of length 2 containing the bounds for the search

Value

double, the optimal bandwidth up to `tol`

Examples

```
x <- simu_rdens(n = 100, model = 3, d = 3)
bandwidth_optim(x = x,
 criterion = "lscv",
 kernel = "Wishart",
 h = 2L)
```

`dinvWishart` *Density of inverse Wishart random matrix*

Description

Density of inverse Wishart random matrix

Usage

```
dinvWishart(x, df, S, log = FALSE)
```

Arguments

`x`	array of dimension `d` by `d` by `n`
`df`	degrees of freedom
`S`	symmetric positive definite matrix of dimension `d` by `d`
`log`	logical; if `TRUE`, returns the log density

Value

a vector of length `n` containing the log-density of the inverse Wishart.

`dmbeta2` *Matrix beta type II density function*

Description

Given a random matrix `x`, compute the density for arguments `shape1` and `shape2`

Usage

```
dmbeta2(x, shape1, shape2, log = TRUE)
```

Arguments

`x`	cube of dimension `d` by `d` by `n` containing the random matrix samples
`shape1`	positive shape parameter, strictly larger than $(d-1)/2$.
`shape2`	positive shape parameter, strictly larger than $(d-1)/2$.
`log`	[logical] if `TRUE` (default), returns the log density.

Value

a vector of length `n`

`dsmlnorm` *Symmetric matrix-variate lognormal density*

Description

Density of the lognormal matrix-variate density, defined through the matrix logarithm, with the Jacobian resulting from the transformation

Usage

```
dsmlnorm(x, b, M, log = TRUE)
```

Arguments

`x`	[cube] array of dimension `d` by `d` by `n`
`b`	[numeric] scale parameter, strictly positive
`M`	[matrix] location matrix, positive definite
`log`	[logical] if `TRUE` (default), returns the log density

Value

a vector of length `n`

`dsmnorm` *Symmetric matrix-variate normal density*

Description

Symmetric matrix-variate normal density

Usage

```
dsmnorm(x, b, M, log = TRUE)
```

Arguments

`x`	[cube] array of dimension `d` by `d` by `n`
`b`	[numeric] scale parameter, strictly positive
`M`	[matrix] location matrix, positive definite
`log`	[logical] if `TRUE` (default), returns the log density

Value

a vector of length `n`

`dWishart` *Density of Wishart random matrix*

Description

Density of Wishart random matrix

Usage

```
dWishart(x, df, S, log = FALSE)
```

Arguments

`x`	array of dimension `d` by `d` by `n`
`df`	degrees of freedom
`S`	symmetric positive definite matrix of dimension `d` by `d`
`log`	logical; if `TRUE`, returns the log density

Value

a vector of length `n` containing the log-density of the Wishart.

`integrate_spd` *Integration with respect to symmetric positive definite matrices*

Description

Given a function `f` defined over the space of symmetric positive definite matrices, compute an integral via numerical integration using the routine cubintegrate.

Usage

```
integrate_spd(
  f,
  dim,
  tol = 0.001,
  lb = 1e-08,
  ub = Inf,
  neval = 1000000L,
  method = c("suave", "hcubature"),
  ...
)
```

Arguments

`f`	function to evaluate that takes as arguments array of size `dim` by `dim` by 1.
`dim`	dimension of integral, only two or three dimensions are supported
`tol`	double for tolerance of numerical integral
`lb`	lower bound for integration range of eigenvalues
`ub`	upper bound for integration range of eigenvalues
`neval`	maximum number of evaluations
`method`	string indicating the method from `cubature`
`...`	additional arguments for the function `f`

Value

list returned by the integration routine. See the documentation of cubintegrate for more details.

Examples

```
integrate_spd(
  dim = 2L,
  neval = 1e4L,
  f = function(x, S){
   dWishart(x, df = 10, S = S, log = FALSE)},
  S = diag(2))
```

`kdens_smlnorm` *Symmetric matrix log-normal kernel density*

Description

Given a sample of `m` points `xs` from an original sample and a set of `n` new sample matrices `x` at which to evaluate the symmetric matrix normal log kernel, return the density with bandwidth parameter `b`.

Usage

```
kdens_smlnorm(x, xs, b, log = TRUE)
```

Arguments

`x`	cube of size `d` by `d` by `n` of points at which to evaluate the density
`xs`	cube of size `d` by `d` by `m` of sample matrices which are used to construct the kernel
`b`	positive double giving the bandwidth parameter
`log`	bool; if `TRUE`, return the log density

Value

a vector of length `n` containing the (log) density of the sample `x`

`kdens_smnorm` *Symmetric matrix normal kernel density*

Description

Given a sample of `m` points `xs` from an original sample and a set of `n` new sample matrices `x` at which to evaluate the symmetric matrix normal kernel, return the density with bandwidth parameter `b`. Note that this kernel suffers from boundary spillover.

Usage

```
kdens_smnorm(x, xs, b, log = TRUE)
```

Arguments

`x`	cube of size `d` by `d` by `n` of points at which to evaluate the density
`xs`	cube of size `d` by `d` by `m` of sample matrices which are used to construct the kernel
`b`	positive double giving the bandwidth parameter
`log`	bool; if `TRUE`, return the log density

Value

a vector of length `n` containing the (log) density of the sample `x`

`kdens_symmat` *Kernel density estimators for symmetric matrices*

Description

Given a sample of `m` points `xs` from an original sample and a set of `n` new sample symmetric positive definite matrices `x` at which to evaluate the kernel, return the density with bandwidth parameter `b`.

Usage

```
kdens_symmat(x, xs, kernel = "Wishart", b = 1, log = TRUE)
```

Arguments

`x`	cube of size `d` by `d` by `n` of points at which to evaluate the density
`xs`	cube of size `d` by `d` by `m` of sample matrices which are used to construct the kernel
`kernel`	string, one of `Wishart`, `smnorm` or `smlnorm`.
`b`	positive double giving the bandwidth parameter
`log`	bool; if `TRUE`, return the log density

Value

a vector of length `n` containing the (log) density of the sample `x`

`kdens_Wishart` *Wishart kernel density*

Description

Given a sample of `m` points `xs` from an original sample and a set of `n` new sample matrices `x` at which to evaluate the Wishart kernel, return the density with bandwidth parameter `b`.

Usage

```
kdens_Wishart(x, xs, b, log = TRUE)
```

Arguments

`x`	cube of size `d` by `d` by `n` of points at which to evaluate the density
`xs`	cube of size `d` by `d` by `m` of sample matrices which are used to construct the kernel
`b`	positive double giving the bandwidth parameter
`log`	bool; if `TRUE`, return the log density

Value

a vector of length `n` containing the (log) density of the sample `x`

`lcv_kdens_symmat` *Likelihood cross-validation for symmetric positive definite matrix kernels*

Description

Given a cube of sample observations (consisting of random symmetric positive definite matrices), and a vector of candidate bandwidth parameters `b`, compute the leave-one-out likelihood cross-validation criterion and return the bandwidth among the choices that maximizes the criterion.

Usage

```
lcv_kdens_symmat(x, b, h = 1L, kernel = "Wishart")
```

Arguments

`x`	array of dimension `d` by `d` by `n`
`b`	vector of candidate bandwidth, strictly positive
`h`	integer for the lag vector for determining which observation to exclude, any data point in a radius of `h`
`kernel`	string indicating the kernel, one of `Wishart`, `smlnorm` or `smnorm`.

Value

a list with arguments

- `lcv` vector of likelihood cross validation criterion
- `b` vector of candidate bandwidth
- `h` lag for leave-one-out
- `bandwidth` optimal bandwidth among candidates
- `kernel` string indicating the choice of kernel function

`lcv_kern_smlnorm` *Likelihood cross validation criterion for symmetric matrix lognormal kernel*

Description

Given a cube `x` and a bandwidth `b`, compute the leave-one-out cross validation criterion by taking out a slice and evaluating the kernel at the holdout value, excluding points that are at distance at least `h-1` apart.

Usage

```
lcv_kern_smlnorm(x, b, h = 1L)
```

Arguments

`x`	[cube] array of dimension `d` by `d` by `n`
`b`	[numeric] scale parameter, strictly positive
`h`	integer lag for excluding observations

Value

the value of the log objective function

`lcv_kern_smnorm` *Likelihood cross validation criterion for symmetric matrix normal kernel*

Description

Given a cube `x` and a bandwidth `b`, compute the leave-one-out cross validation criterion by taking out a slice and evaluating the kernel at the holdout values, excluding points that are at distance at least `h-1` apart.

Usage

```
lcv_kern_smnorm(x, b, h = 1L)
```

Arguments

`x`	[cube] array of dimension `d` by `d` by `n`
`b`	[numeric] scale parameter, strictly positive
`h`	integer lag for excluding observations

Value

the value of the log objective function

`lcv_kern_Wishart` *Likelihood cross validation criterion for Wishart kernel*

Description

Given a cube `x` and a bandwidth `b`, compute the leave-one-out cross validation criterion by taking out a slice and evaluating the kernel at the holdout value, excluding points that are at distance at least `h-1` apart.

Usage

```
lcv_kern_Wishart(x, b, h = 1)
```

Arguments

`x`	[cube] array of dimension `d` by `d` by `n`
`b`	[numeric] scale parameter, strictly positive
`h`	integer lag for excluding observations

Value

the value of the log objective function

`lscv_kdens_symmat` *Least square cross-validation for symmetric positive definite matrix kernels*

Description

Given a cube of sample observations (consisting of random symmetric positive definite matrices), and a vector of candidate bandwidth parameters `b`, compute the least square likelihood cross-validation criterion and return the bandwidth among the choices that minimizes the criterion.

Usage

```
lscv_kdens_symmat(x, b, h = 1L, kernel = "Wishart")
```

Arguments

`x`	array of dimension `d` by `d` by `n`
`b`	vector of candidate bandwidth, strictly positive
`h`	integer for the lag vector for determining which observation to exclude, any data point in a radius of `h`
`kernel`	string indicating the kernel, one of `Wishart` or `smlnorm`.

Value

a list with arguments

- `lscv` vector of likelihood cross validation criterion
- `b` vector of candidate bandwidth
- `h` lag for leave-one-out
- `bandwidth` optimal bandwidth among candidates
- `kernel` string indicating the choice of kernel function

`lscv_kern_smlnorm` *Least square cross validation criterion for log symmetric matrix normal kernel*

Description

Finite sample h-block leave-one-out approximation to the least square criterion, omitting constant term. Only pairs that are $|i - j| \leq h$ apart are considered.

Usage

```
lscv_kern_smlnorm(x, b, h = 1L)
```

Arguments

`x`	[cube] array of dimension `d` by `d` by `n`
`b`	[numeric] scale parameter, strictly positive
`h`	[int] integer indicating the separation lag

Value

a double containing the log of the least square cross validation criterion

---

`lscv_kern_Wishart` *Least square cross validation criterion for Wishart kernel*

Description

Finite sample h-block leave-one-out approximation to the least square criterion, omitting constant term.

Usage

```
lscv_kern_Wishart(x, b, h = 1L)
```

Arguments

`x`	[cube] array of dimension `d` by `d` by `n`
`b`	[numeric] scale parameter, strictly positive
`h`	separation vector; only pairs that are $\|i-j\| \leq h$ apart are considered

Value

a double containing the a vector of length two containing the log of the summands

---

`mgamma` *Multivariate gamma function*

Description

Given a vector of points `x` and an order `p`, compute the multivariate gamma function. The function is defined as

$$\gamma_p(x) = \pi^{p(p-1)/4} \prod_{i=1}^{p} \Gamma\{x + (1-i)/2\}.$$

Usage

```
mgamma(x, p, log = FALSE)
```

Arguments

`x`	[vector] of points at which to evaluate the function
`p`	[int] dimension of the multivariate gamma function, strictly positive.
`log`	[logical] if `TRUE`, returns the log multivariate gamma function.

Value

a matrix with one column of the same length as `x`

`realvar` *Realized variance of Amazon and SPY*

Description

Intraday realized covariances of the returns between the Amazon stock (`rvarAMZN`) and the SPDR S&P 500 ETF (`rvarSPY`) using five minutes data, for the period of September 13th, 2023 to September 12, 2024.

Usage

```
realvar
```

Format

A 2 by 2 by 250 array

Source

Anne MacKay

Examples

```
data(realvar, package = "ksm")
bopt <- bandwidth_optim(
 x = realvar,
 criterion = "lscv",
 kernel = "Wishart",
 h = 4L
 )
```

`Riccati` *Solver for Riccati equation*

Description

Given two matrices `M` and `S`, solve Riccati equation by iterative updating to find the solution $\mathbf{R}$, where the latter satisfies

$$\mathbf{R} = \mathbf{M}\mathbf{R}\mathbf{M}^\top + \mathbf{S}$$

until convergence (i.e., when the Frobenius norm is less than `tol`, or the maximum number of iterations `maxiter` is reached.

Usage

```
Riccati(M, S, tol = 1e-08, maxiter = 10000L)
```

Arguments

`M`	matrix
`S`	matrix
`tol`	double for tolerance
`maxiter`	integer, the maximum number of iterations

Value

a list containing

- `solution` matrix solution to Riccati's equation
- `error` numerical error
- `niter` number of iteration
- `convergence` bool indicating convergence (`TRUE`) if `niter` < `maxiter`

`rinvWishart` *Random matrix generation from the inverse Wishart distribution*

Description

Random matrix generation from the inverse Wishart distribution

Usage

```
rinvWishart(n, df, S)
```

Arguments

`n`	[integer] sample size
`df`	[double] degrees of freedom, positive
`S`	[matrix] a `d` by `d` positive definite scale matrix

Value

an array of dimension `d` by `d` by `n` containing the samples

---

`rmbeta2` *Random matrix generation from matrix beta type II distribution*

Description

This function only supports the case of diagonal matrices

Usage

```
rmbeta2(n, d, shape1, shape2)
```

Arguments

`n`	sample size
`d`	dimension of the matrix
`shape1`	positive shape parameter, strictly larger than $(d-1)/2$.
`shape2`	positive shape parameter, strictly larger than $(d-1)/2$.

Value

a cube of dimension `d` by `d` by `n`

`rmnorm` *Random vector generation from the multivariate normal distribution*

Description

Sampler derived using the eigendecomposition of the covariance matrix `vcov`.

Usage

```
rmnorm(n, mean, vcov)
```

Arguments

`n`	sample size
`mean`	mean vector of length `d`
`vcov`	a square positive definite covariance matrix, of the same dimension as `mean`.

Value

an `n` by `d` matrix of samples

Examples

```
rmnorm(n = 10, mean = c(0, 2), vcov = diag(2))
```

`rWAR` *Random matrix generation from first-order autoregressive Wishart process*

Description

Given a matrix of coefficients `M` and a covariance matrix `Sigma`, simulate `n` random matrices from a first-order autoregressive Wishart process by simulating from cross-products of vector autoregressions

Usage

```
rWAR(n, M, Sigma, K = NULL, order = 1L, burnin = 25L)
```

Arguments

`n`	sample size
`M`	matrix of autoregressive coefficients
`Sigma`	covariance matrix
`K`	integer, degrees of freedom
`order`	order of autoregressive process, only `1` is supported at current.
`burnin`	number of iterations discarded

Value

an array of size `d` by `d` by `n` containing the samples

References

C. Gourieroux, J. Jasiak, and R. Sufana (2009). The Wishart Autoregressive process of multivariate stochastic volatility, *Journal of Econometrics*, 150(**2**), 167-181, <doi:10.1016/j.jeconom.2008.12.016>.

Examples

```
M <- matrix(c(0.3, -0.3, -0.3, 0.3), nrow = 2)
Sigma <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
rWAR(n = 10, M = M, Sigma = Sigma, K = 5)
```

`rWishart` *Random matrix generation from Wishart distribution*

Description

Random matrix generation from Wishart distribution

Usage

```
rWishart(n, df, S)
```

Arguments

`n`	[integer] sample size
`df`	[double] degrees of freedom, positive
`S`	[matrix] a `d` by `d` positive definite scale matrix

Value

an array of dimension `d` by `d` by `n` containing the samples

---

`symmetrize` *Symmetrize matrix*

Description

Given an input matrix, symmetrize by taking average of lower and upper triangular components as $A + A^\top$.

Usage

```
symmetrize(A)
```

Arguments

`A`	square matrix

Value

symmetrized version of `A`

Index