Package 'tensr'

August 15, 2018

Type Package

Title Covariance Inference and Decompositions for Tensor Datasets

Version 1.0.1

Date 2018-08-13

Description A collection of functions for Kronecker structured covariance estimation and testing under the array normal model. For estimation, maximum likelihood and Bayesian equivariant estimation procedures are implemented. For testing, a likelihood ratio testing procedure is available. This package also contains additional functions for manipulating and decomposing tensor data sets. This work was partially supported by NSF grant DMS-1505136. Details of the methods are described in Gerard and Hoff (2015) <doi:10.1016/j.jmva.2015.01.020> and Gerard and Hoff (2016) <doi:10.1016/j.laa.2016.04.033>.

License GPL-3

LazyData TRUE

RoxygenNote 6.0.1

BugReports http://github.com/dcgerard/tensr/issues

Suggests knitr, rmarkdown, covr, testthat

VignetteBuilder knitr

Imports assertthat

NeedsCompilation no

Author David Gerard [aut, cre] (<https://orcid.org/0000-0001-9450-5023>), Peter Hoff [aut]

Maintainer David Gerard <gerard.1787@gmail.com>

Repository CRAN

Date/Publication 2018-08-15 18:00:08 UTC

R topics documented:

`amprod` *k-mode product.*

Description

`amprod` returns the k-mode product of an array with a matrix.

Usage

```
amprod(A, M, k)
```

Arguments

`A`	A real valued array.
`M`	A real matrix.
`k`	An integer. The mode along which `M` is to be multiplied to `A`.

Details

The k-mode product of a tensor A with a matrix M results in a tensor whose k-mode unfolding is M times the k-mode unfolding of A. That is `mat(amprod(A,M,k)) = M %*% mat(A,k)`. More details of the k-mode product can be found in Kolda and Bader (2009).

Value

An array whose k-mode unfolding is `M %*% mat(A,k)`.

Author(s)

Peter Hoff.

References

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3), 455-500.

See Also

`atrans` for applying multiple k-mode products.

Examples

```
A <- array(1:8, dim = c(2,2,2))
M <- matrix(1:4, nrow = 2, ncol = 2)
Y <- amprod(A, M, 2)
Y
identical(M %*% mat(A,2), mat(Y,2))
```

anorm_cd *Array normal conditional distributions.*

Description

Conditional mean and variance of a subarray.

Usage

```
anorm_cd(Y, M, S, saidx)
```

Arguments

Y	A real valued array.
M	Mean of Y.
S	List of mode-specific covariance matrices of Y.
saidx	List of indices for indexing sub-array for which the conditional mean and variance should be computed. For example, `said_x = list(1:2,1:2, 1:2)` will compute the conditional means and variances for the 2 by 2 by 2 sub-array Y[1:2, 1:2, 1:2]. This is conditional on every other element in Y.

Details

This function calculates the conditional mean and variance in the array normal model. Let Y be array normal and let Y_a be a subarray of Y. Then this function will calculate the conditional means and variances of Y_a, conditional on every other element in Y.

Author(s)

Peter Hoff.

References

Hoff, P. D. (2011). Separable covariance arrays via the Tucker product, with applications to multivariate relational data. *Bayesian Analysis*, 6(2), 179-196.

Examples

```
p <- c(4, 4, 4)
Y <- array(stats::rnorm(prod(p)), dim = p)
saidx <- list(1:2, 1:2, 1:2)
true_cov <- tensr::start_ident(p)
true_mean <- array(0, dim = p)
cond_params <- anorm_cd(Y = Y, M = true_mean, S = true_cov, saidx = saidx)

## Since data are independent standard normals, conditional mean is 0 and
##    conditional covariance matrices are identities.
cond_params$Mab
cond_params$Sab
```

array_bic_aic *Calculate the AIC and BIC.*

Description

Calculate the AIC and BIC for Kronecker structured covariance models, assuming the array normal distribution.

Usage

```
array_bic_aic(sig_squared, p, mode_ident = NULL, mode_diag = NULL,
  mode_unstructured = NULL)
```

Arguments

sig_squared	A numeric. The MLE of sigma^2 in the array normal model (the 'variance' form of the total variation parameter).
p	A vector of integers. The dimension of the data array (including replication modes).
mode_ident	A vector of integers. The modes assumed to have identity covariances.
mode_diag	A vector of integers. The modes assumed to have diagional covariances.
mode_unstructured	A vector of integers. The modes of assumed to have unstructured covariances.

Details

The AIC and BIC depend only on the data through the MLE of the total variation parameter. Given this, the dimension of the array, and a specification of which modes are the identity and which are unstructured, this function will calculate the AIC and BIC.

Value

AIC A numeric. The AIC of the model.

BIC A numeric. The BIC of the model.

Author(s)

David Gerard.

See Also

holq for obtaining sig_squared.

Examples

```
# Generate random array data with first mode having unstructured covariance
#  second having diagonal covariance structure and third mode having identity
#  covariance structure.
set.seed(857)
p <- c(4, 4, 4)
Z <- array(stats::rnorm(prod(p)), dim = p)
Y <- atrans(Z, list(tensr:::rwish(diag(p[1])), diag(1:p[2]), diag(p[3])))

# Use holq() to fit various models.
false_fit1 <- holq(Y, mode_rep = 1:3) ## identity for all modes
false_fit2 <- holq(Y, mode_rep = 2:3) ## unstructured first mode
true_fit <- holq(Y, mode_rep = 3, mode_diag = 2) ## correct model

# Get AIC and BIC values.
false_aic1 <- array_bic_aic(false_fit1$sig ^ 2, p, mode_ident = 1:length(p))
false_aic2 <- array_bic_aic(false_fit2$sig ^ 2, p, mode_ident = 2:length(p),
                            mode_unstructured = 1)
true_aic <- array_bic_aic(true_fit$sig ^ 2, p, mode_ident = 2:length(p), mode_diag = 1)

# Plot the results.
plot(c(false_aic1$AIC, false_aic2$AIC, true_aic$AIC), type = "l",
     xaxt = "n", xlab = "Model", ylab = "AIC", main = "AIC")
axis(side = 1, at = 1:3, labels = c("Wrong Model 1", "Wrong Model 2", "Right Model"))

plot(c(false_aic1$BIC, false_aic2$BIC, true_aic$BIC), type = "l", xaxt = "n",
     xlab = "Model", ylab = "BIC", main = "BIC")
axis(side = 1, at = 1:3, labels = c("Wrong Model 1", "Wrong Model 2", "Right Model"))
```

---

arrIndices *Array indices.*

Description

Generates indices corresponding to subarrays.

Usage

```
arrIndices(saidx)
```

Arguments

saidx	either a vector of the dimensions of a potential array, or a list of the indices in the subarray.

Details

This function generates a matrix corresponding to all combinations of a list of indices, to be used in subsetting arrays.

Author(s)

Peter Hoff.

Examples

```
# all indices of an array
arrIndices(c(4, 3, 2))
# indices of a subarray
arrIndices(list(c(1, 3), c(4, 5), c(2, 3, 6)))
```

`atrans` *Tucker product.*

Description

Performs the Tucker product between an array and a list of matrices.

Usage

```
atrans(A, B)
```

Arguments

`A`	An array of dimension K.
`B`	A list of matrices of length K. It must be that `ncol(B[[k]]) == dim(A)[k]`.

Details

The Tucker product between a list of matrices `B` and an array `A` is formally equivalent to performing the k-mode product between `A` and each list element in `B`. For example, if the dimension of `A` is three, then `atrans(A,B) = amprod(amprod(amprod(A,B[[1]],1),B[[2]],2),B[[3]],3)`. The ordering of this k-mode product does not matter. See Kolda and Bader (2009) for details.

Author(s)

Peter Hoff.

References

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3), 455-500.

See Also

`amprod` for multiplying one matrix along one mode of an array.

Examples

```
A <- array(1:8, dim = c(2,2,2))
B <- list()
B[[1]] <-matrix(1:4, nrow = 2)
B[[2]] <- matrix(1:6, nrow = 3)
B[[3]] <- matrix(1:2, nrow = 1)
atrans(A,B)
```

`collapse_mode` *Collapse multiple modes into one mode.*

Description

Given an array `X` and a vector of integers `m`, `collapse_mode` returns an array of lower order where the first mode indexes the modes indicated in `m`.

Usage

```
collapse_mode(X, m)
```

Arguments

`X`	An array whose modes we are collapsing.
`m`	A vector of integers giving the modes to collapse.

Details

Transforms an array into another array where the provided modes are collapsed into one mode. The indexing along this new mode is in lexicographical order of the indices of the collapsed modes. The collapsed mode is the first mode unless `length(m) == 1`, then `collapse_mode` simply returns `X`.

Value

If X is of order K and `length(m) = q`, then returns an array Y of order $K - q + 1$, where the modes indicated in `m` are combined to be the first mode in Y.

Author(s)

David Gerard.

Examples

```
X <- array(rep(c(1, 2), 8), dim = c(2, 2, 2, 2))
X
#mode 1 is now mode 2, modes 2, 3, and 4 are combined to be mode 1.
collapse_mode(X, c(2, 3, 4))
collapse_mode(X, c(2, 4)) ## another example.
collapse_mode(X, 4) #returns X
```

convert_cov *Convert the output from* equi_mcmc *to component covariance matrices.*

Description

This takes the output from equi_mcmc, which are the inverses of the lower-triangular Cholesky square roots of the component covariance matrices, and returns the component covariance matrices. These are the more useful posterior draws to use in actual data analysis.

Usage

```
convert_cov(equi_mcmc_obj)
```

Arguments

equi_mcmc_obj	The output from equi_mcmc, which contains a list. The first element is a list containing the posterior draws of the inverses of the lower-triangular Cholesky square roots of each component covariance matrix. The second list element is a total variation parameter, but the square root of the version used in calculating the overall covariance matrix.

Details

The output from equi_mcmc is the inverse of the lower-triangular Cholesky square root of each component covariance matrix. This output is convenient for calculating the Bayes rule under multiway-Stein's loss (see get_equi_bayes). Call one of these outputs from equi_mcmc Ψ. Then this function calculates $\Sigma = \Psi^{-}1\Psi^{-}T$, which are the posterior draws of the component covariance matrices. These component covariance matrices are constrained to have determinant one, hence there is a total variation parameter σ^2.

Value

cov_post A list containing the posterior draws of each component covariance matrix.

sig2_post A vector containing the posterior draws of the total variation parameter.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

See Also

equi_mcmc.

Examples

```
#Generate data whose true covariance is just the identity.
p <- c(4,4,4)
X <- array(stats::rnorm(prod(p)),dim = p)
#Then run the Gibbs sampler.
mcmc_out <- equi_mcmc(X)
cov_out <- convert_cov(mcmc_out)

# Some trace plots.
plot(cov_out[[2]], type = 'l', xlab = 'Iteration',
     ylab = expression(sigma ^ 2), main = 'Trace Plot')
abline(h = 1, col = 2, lty = 2)
legend('topleft', 'True Value', col = 2, lty = 2, bty = 'n')

k <- sample(1:length(p), size = 1)
i <- sample(1:p[k], size = 1)
j <- sample(1:p[k], size = 1)
plot(cov_out[[1]][[k]][i, j, ], type = 'l', xlab = 'Iteration',
     main = 'Trace Plot',
     ylab = substitute(Sigma[k][group('[', list(i, j), ']')],
                       list(k = k, i = i, j = j)))
abline(h = 1 * (i == j), lty =  2, col = 2)
legend('topleft', 'True Value', col = 2, lty = 2, bty = 'n')
```

---

demean_tensor *Demeans array data.*

Description

Rotates an array into two parts, one of which has mean zero.

Usage

```
demean_tensor(X, mode_reps)
```

Arguments

X	An array, one of whose modes is assumed to be samples from the array normal model.
mode_reps	The mode(s) that contain(s) the samples, or repetitions, from the array normal model.

Details

If one mode contains samples (or repetitions), then this function will rotate the array into two parts, a mean part and a covariance part. The 'covariance part' has mean zero and the rest of the methods in this package apply. The 'mean part' is simply the sample mean. If the data are array normal, then the 'covariance part' will also be array normal with the *exact* same covariance structure as the original tensor, except that there are one fewer samples.

Value

`Y` An array that has the same dimensions as `X` except that the mode `mode_reps` has dimension one smaller. This array is mean 0 array normal with the same covariance structure as `X`.

`X_bar` The sample mean of `X`. Under the array normal model, `X` and `Y` are statistically independent.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

---

`equi_mcmc` *Gibbs sampler using an invariant prior.*

Description

`equi_mcmc` obtains posterior draws that are useful in optimal equivariant estimation under the array normal model.

Usage

```
equi_mcmc(X, itermax = 1000, start_identity = FALSE, print_iter = FALSE,
  mode_rep = NULL)
```

Arguments

`X`	A tensor.
`itermax`	The number of iterations in the Gibb's sampler.
`start_identity`	Should we start the component covariance matrices at the identity (TRUE) or the sample covariance matrices (FALSE)?
`print_iter`	Should we print the iteration number at each iteration?
`mode_rep`	The mode that contains samples. I.e., the mode whose component covariance matrix is the identity. If NULL then no modes are assumed to have identity covariance.

Details

equi_mcmc obtains posterior samples of the component covariance matrices from the array normal model. This is with respect to using the right Haar measure over a product group of lower triangular matrices as the prior.

This returns only the upper triangular Cholesky square root of the inverses of the component covariance matrices. Equivalently, these are the inverses of the lower triangular Cholesky square roots of the component covariance matrices. This is because sampling the inverse is faster computationally and the Bayes rules (based on multiway Stein's loss) only depend on the inverse.

Value

Phi_inv List of posterior draws of the inverse of the cholesky square roots of each component covariance matrix. Phi_inv[[i]][,,j] provides the jth sample of the ith component.

sigma Vector of posterior samples of the overall scale paramater.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

See Also

sample_right_wishart and sample_sig for the Gibbs updates. convert_cov and get_equi_bayes for getting posterior summaries based on the output of equi_mcmc. multiway_takemura for an improvement on this procedure.

Examples

```
#Generate data whose true covariance is just the identity.
p <- c(2,2,2)
X <- array(stats::rnorm(prod(p)),dim = p)
#Then run the Gibbs sampler.
mcmc_out <- equi_mcmc(X)
plot(mcmc_out$sigma, type = 'l', lwd = 2, ylab = expression(sigma),
     xlab = 'Iteration', main = 'Trace Plot')
abline(h = 1,col = 2,lty = 2)
```

`fnorm` *Frobenius norm of an array.*

Description

Calculates the Frobenius norm of an array.

Usage

```
fnorm(X)
```

Arguments

`X`	An array, a matrix, or a vector.

Details

The Frobenius norm of an array is the square root of the sum of its squared elements. This function works for vector and matrix arguments as well.

Value

The square root of the sum of the squared elements of `X`.

Author(s)

David Gerard.

Examples

```
X <- c(1:8)
Y <- matrix(1:8, nrow = 2)
Z <- array(1:8, dim = c(2, 2, 2))
fnorm(X)
fnorm(Y)
fnorm(Z)
```

`get_equi_bayes` *Get the Bayes rule under multiway Stein's loss.*

Description

Given the output of `equi_mcmc`, this function will calculate the Bayes rule under multiway Stein's loss.

Usage

```
get_equi_bayes(psi_inv, sigma, burnin = NULL)
```

Arguments

`psi_inv`	A list of arrays where `psi_inv[[i]][[, , j]]` is the jth update of the `i`th component. These components are the inverses of the lower-triangular Cholesky square roots of the component covariance matrices. You can just use the `Phi_inv` output from `equi_mcmc`.
`sigma`	A vector of posteior draws of the total variation parameter. This is just `sigma` from the output of `equi_mcmc`.
`burnin`	A numeric between 0 and 1. What proportion of the posterior samples do you want to discard as burnin? The default is 0.25.

Details

Multiway Stein's loss is a generalization of Stein's loss to more than two dimensions. The Bayes rule under this loss is simply represented in terms of the posterior moments of the component precision matrices. These moments can be approximated by using the output of `equi_mcmc`. When using the invariant prior that is used in `equi_mcmc`, the resulting Bayes rule is the uniformly minimum risk equivariant estimator.

More details on multiway Stein's loss and the Bayes rules under it can be found in Gerard and Hoff (2015).

Value

`Sig_hat` A list of the Bayes rules of the component covariance matrices under multiway Stein's loss.

`B` A list of the lower-triangular Cholesky square roots of the Bayes rules of the component covariance matrices under multiway Stein's loss. We have that `Sig_hat[[i]]` is equal to `B[[i]] %*% t(B[[i]])`.

`b` A numeric. This is the bayes rule of the total variation parameter. This is the 'standard deviation' version. That is, the `b ^ 2` would be used to calculate the overall covariance matrix.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

See Also

equi_mcmc.

Examples

```
#Generate data whose true covariance is just the identity.
p <- c(4,4,4)
X <- array(stats::rnorm(prod(p)),dim = p)
#Then run the Gibbs sampler.
mcmc_out <- equi_mcmc(X)
bayes_rules <- get_equi_bayes(mcmc_out$Phi_inv, mcmc_out$sigma)
bayes_rules$Sig_hat[[1]]
```

---

get_isvd *Calculate the incredible SVD (ISVD).*

Description

The ISVD is a generalization of the SVD to tensors. It is derived from the incredible HOLQ.

Usage

```
get_isvd(x_holq)
```

Arguments

x_holq	The output from holq.

Details

Let `sig * atrans(Z, L)` be the HOLQ of `X`. Then the ISVD calculates the SVD of each `L[[i]]`, call it `U[[i]] %*% D[[i]] %*% t(W[[i]])`. It then returns `l = sig`, `U`, `D`, and `V = atrans(Z, W)`. These values have the property that `X` is equal to `l * atrans(atrans(V, D), U)`, up to numerical precision. `V` is also scaled all-orthonormal.

For more details on the ISVD, see Gerard and Hoff (2016).

Value

l A numeric.

U A list of orthogonal matrices.

D A list of diagonal matrices with positive diagonal entries and unit determinant. The diagonal entries are in descending order.

V A scaled all-orthonormal array.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2016). A higher-order LQ decomposition for separable covariance models. *Linear Algebra and its Applications*, 505, 57-84. https://doi.org/10.1016/j.laa.2016.04.033 http://arxiv.org/pdf/1410.1094v1.pdf

Examples

```
#Generate random data.
p <- c(4,4,4)
X <- array(stats::rnorm(prod(p)), dim = p)

#Calculate HOLQ, then ISVD
holq_x <- holq(X)
isvd_x <- get_isvd(holq_x)
l <- isvd_x$l
U <- isvd_x$U
D <- isvd_x$D
V <- isvd_x$V

#Recover X
trim(X - l * atrans(atrans(V, D), U))

#V is scaled all-orthonormal
trim(mat(V, 1) %*% t(mat(V, 1)), epsilon = 10^-5)

trim(mat(V, 2) %*% t(mat(V, 2)), epsilon = 10^-5)

trim(mat(V, 3) %*% t(mat(V, 3)), epsilon = 10^-5)
```

---

holq *Calculate the incredible higher-order LQ decomposition (HOLQ).*

Description

This function calculates a generalization of the LQ decomposition to tensors. This decomposition has a close connection to likelihood inference in Kronecker structured covariande models.

Usage

```
holq(X, tol = 10^-9, itermax = 1000, mode_rep = NULL, mode_diag = NULL,
  mode_ldu = NULL, print_diff = TRUE, start_vals = "identity",
  use_sig = TRUE)
```

Arguments

X	An array of numerics.
tol	A numeric. The maximum difference in frobenius norm between two successive core arrays before stopping. Or maximum difference of the ratio of sigs from 1 before stopping (which depends on the value of use_sig).
itermax	An integer. The maximum number of iterations of the LQ decomposition to do before stopping.
mode_rep	A vector of integers. The optional mode(s) to be considered identity matrices.
mode_diag	A vector of integers. The optional mode(s) to be considered as independent but heteroscedastic.
mode_ldu	A vector of integers. The optional modes(s) to be considered as having unit diagonal.
print_diff	A logical. Should the updates be printed to the screen each iteration?
start_vals	Determines how to start the optimization algorithm. Either 'identity' (default), or 'residuals', which results in using the cholesky square roots of the sample covariance matrices along each mode scaled to have unit determinant. You can also use your own start values.
use_sig	A logical. What stopping criterion should we use? Frobenius norm of difference of cores (FALSE) or absolute value of difference of ratio of sig from 1 (TRUE).

Details

Given an array X, the default version of this function will calculate (1) L a list of lower triangular matricies with positive diagonal elements and unit determinant, Z an array of the same dimensions as X that has special orthogonal properties, and (3) sig a numeric such that X is the same as sig * atrans(Z,L) up to numeric precision.

This output (1) can be considered a generalization of the LQ decomposition to tensors, (2) solves an optimization problem which the matrix LQ decomposition solves, and (3) has a special connection to likelihood inference in the array normal model.

There are options to constrain the matrices in L to either be diagonal, lower triangular with unit diagonal, or the identity matrix. Each of these correspond to submodels in Kronecker structured covariance models. The core array corresponding to each of these options has different properities (see Gerard and Hoff (2016)). These more constrained tensor decompositions are called HOLQ juniors.

The MLE of the ith component covariance matrix under *any* elliptically contoured Kronecker structured covariance model is given by L[[i]] %*% t(L[[i]]). The MLE for the total variation pamarameter will be different depending on the distribution of the array, but for the array normal it is sig ^ 2 / prod(p) (the "variance" form for the total variation parameter).

The likelihood ratio test statistic depends only on sig and can be implemented in lrt_stat.

The algorithm used to fit the HOLQ iteratively repeats the LQ decomposition along each mode.

For more details on the incredible HOLQ, see Gerard and Hoff (2016).

Value

Z The core array with scaled all-orthonormality property.

A A list of matrices.

sig A numeric. The total variation parameter. This is the "standard deviation" form.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2016). A higher-order LQ decomposition for separable covariance models. *Linear Algebra and its Applications*, 505, 57-84. https://doi.org/10.1016/j.laa.2016.04.033 http://arxiv.org/pdf/1410.1094v1.pdf

See Also

array_bic_aic for using the output of holq to calculate AIC and BIC,

get_isvd for using the output of holq to calculate a tensor generalization of the singular value decomposition.

lq for the matrix LQ decomposition.

lrt_stat for using the output of holq to calculate likelihood ratio test statistics.

mle_from_holq for using the output of holq to calculate the maximum likelihood estimates of the component covariance matrices under the array normal model.

Examples

```
#Genrate random data.
p <- c(2, 3, 4)
X <- array(stats::rnorm(prod(p)), dim = p)

#Calculate HOLQ with unit diagonal on 2nd mode,
#  and diagonal along 3rd mode.
holq_x <- holq(X, mode_ldu = 2, mode_diag = 3)
Z <- holq_x$Z
A <- holq_x$A
sig <- holq_x$sig

#Reconstruct X
trim(X - sig * atrans(Z, A), 10^-5)

#Properties of core
#First mode has orthonormal rows.
trim(mat(Z, 1) %*% t(mat(Z, 1)), 10^-5)

#Second mode has orthogonal rows.
trim(mat(Z, 2) %*% t(mat(Z, 2)), 10^-5)

#Third mode has unit diagonal (up to scale).
```

```
diag(mat(Z, 3) %*% t(mat(Z, 3)))
```

`hooi` *Calculate the higher-order orthogonal iteration (HOOI).*

Description

This function will calculate the best rank `r` (where `r` is a vector) approximation (in terms of sum of squared differences) to a given data array.

Usage

```
hooi(X, r, tol = 10^-6, print_fnorm = FALSE, itermax = 500)
```

Arguments

`X`	An array of numerics.
`r`	A vector of integers. This is the given low multilinear rank of the approximation.
`tol`	A numeric. Stopping criterion.
`print_fnorm`	Should updates of the optimization procedure be printed? This number should get larger during the optimizaton procedure.
`itermax`	The maximum number of iterations to run the optimization procedure.

Details

Given an array `X`, this code will find a core array `G` and a list of matrices with orthonormal columns `U` that minimizes `fnorm(X -atrans(G, U))`. If `r` is equal to the dimension of `X`, then it returns the HOSVD (see `hosvd`).

For details on the HOOI see Lathauwer et al (2000).

Value

`G` An all-orthogonal core array.

`U` A vector of matrices with orthonormal columns.

Author(s)

David Gerard.

References

De Lathauwer, L., De Moor, B., & Vandewalle, J. (2000). On the best rank-1 and rank-($r_1, r_2, ..., r_n$) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4), 1324-1342.

Examples

```
## Generate random data.
p <- c(2, 3, 4)
X <- array(stats::rnorm(prod(p)), dim = p)

## Calculate HOOI
r <- c(2, 2, 2)
hooi_x <- hooi(X, r = r)
G <- hooi_x$G
U <- hooi_x$U

## Reconstruct the hooi approximation.
X_approx <- atrans(G, U)
fnorm(X - X_approx)
```

`hosvd` *Calculate the (truncated) higher-order SVD (HOSVD).*

Description

Calculates the left singular vectors of each matrix unfolding of an array, then calculates the core array. The resulting output is a Tucker decomposition.

Usage

```
hosvd(Y, r = NULL)
```

Arguments

`Y`	An array of numerics.
`r`	A vector of integers. The rank of the truncated HOSVD.

Details

If `r` is equal to the rank of `Y`, then `Y` is equal to `atrans(S, U)`, up to numerical accuracy.

More details on the HOSVD can be found in De Lathauwer et. al. (2000).

Value

`U` A list of matrices with orthonormal columns. Each matrix contains the mode-specific singular vectors of its mode.

`S` An all-orthogonal array. This is the core array from the HOSVD.

Author(s)

Peter Hoff.

References

De Lathauwer, L., De Moor, B., & Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4), 1253-1278.

Examples

```
#Generate random data.
p <- c(2, 3, 4)
X <- array(stats::rnorm(prod(p)), dim = p)

#Calculate HOSVD.
hosvd_x <- hosvd(X)
S <- hosvd_x$S
U <- hosvd_x$U

#Recover X.
trim(X - atrans(S, U))

#S is all-orthogonal.
trim(mat(S, 1) %*% t(mat(S, 1)))
trim(mat(S, 2) %*% t(mat(S, 2)))
trim(mat(S, 3) %*% t(mat(S, 3)))
```

---

ihop *The incredible higher-order polar decomposition (IHOP).*

Description

Mmm, pancakes.

Usage

```
ihop(X, itermax = 100, tol = 10^-9, print_diff = TRUE, mode_rep = NULL,
  use_sig = TRUE)
```

Arguments

`X`	An array of numerics.
`itermax`	An integer. The maximum number of iterations to perform during the optimization procedure.
`tol`	A numeric. The algorithm will stop when the Frobenius norm of the difference of core arrays between subsequent iterations is below `tol` (for `use_sig = FALSE`) or when the absolute difference between the ratio of subsequent values of `sig` and 1 is less than `tol` (for `use_sig = TRUE`).
`print_diff`	A logical. Should we print the updates of the algorithm?
`mode_rep`	A vector. Which component matrices should be set to be the identity?
`use_sig`	A logical. See `tol`.

Details

This function will calculate the higher-order polar decomposition, a generalization of the polar decomposition to tensors. It generalizes a minimization formulation of the polar decomposition.

Given an array `X`, `ihop` will output `L` a list of lower triangular matrices with positive diagonal elements and unit Frobenius norm, `R` a core array with certain orthogonality properties, and `sig` a total variation parameter. We have that `X` is equal to `sig * atrans(R, L)` up to numerical precision.

`t(solve(L[[i]])) %*% mat(R, i)` will have orthonormal rows for all `i`.

For more details on the IHOP, see Gerard and Hoff (2016).

Value

`R` A core array which, in combination with `L`, has certain orthogonality properties.

`L` A list of lower triangular matrices with unit Frobenius norm.

`sig` A numeric.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2016). A higher-order LQ decomposition for separable covariance models. *Linear Algebra and its Applications*, 505, 57-84. https://doi.org/10.1016/j.laa.2016.04.033 http://arxiv.org/pdf/1410.1094v1.pdf

Examples

```
#Generate random data.
p <- c(2, 3, 4)
X <- array(stats::rnorm(prod(p)), dim = p)

#Calculate IHOP.
ihop_x <- ihop(X)
R <- ihop_x$R
L <- ihop_x$L
sig <- ihop_x$sig

#Reconstruct X
trim(X - sig * atrans(R, L))

#Orthogonality properties
ortho_1 <- t(solve(L[[1]])) %*% mat(R, 1)
trim(ortho_1 %*% t(ortho_1))

ortho_2 <- t(solve(L[[2]])) %*% mat(R, 2)
trim(ortho_2 %*% t(ortho_2))

ortho_3 <- t(solve(L[[3]])) %*% mat(R, 3)
```

```
trim(ortho_3 %*% t(ortho_3))
```

`kendalltau` *Kendall's tau measure of association.*

Description

This function provides a Monte Carlo approximation to Kendall's tau measure of association.

Usage

```
kendalltau(x, y, nmc = 1e+05)
```

Arguments

`x`	a vector.
`y`	a vector.
`nmc`	an integer number of Monte Carlo simulations.

Value

A Monte Carlo approximation to Kendall's tau measure of association.

Author(s)

Peter Hoff.

Examples

```
mu <- rexp(30)
tensr:::kendalltau(rpois(30, mu), rpois(30, mu))
```

`Kom` *Commutation matrix.*

Description

Construct the communtation matrix.

Usage

```
Kom(m, n)
```

Arguments

m	a natural number.
n	another natural number.

Details

This function constructs the commutation matrix K, which maps c(A) to c(t(A)) for an m by n matrix A.

Value

K The m * n by m * n commutation matrix.

Author(s)

Peter Hoff.

References

Magnus, J. R., & Neudecker, H. (1979). The commutation matrix: some properties and applications. *The Annals of Statistics*, 381-394.

Tracy, D. S., & Dwyer, P. S. (1969). Multivariate maxima and minima with matrix derivatives. *Journal of the American Statistical Association*, 64(328), 1576-1594.

Examples

```
m <- 5 ; n <- 4
A <- matrix(stats::rnorm(m * n), m, n)
Kom(5, 4) %*% c(A) - c(t(A))
```

---

ldan *Log-likelihood of array normal model.*

Description

ldan calculates the log-likelihood of the array normal model, minus a constant.

Usage

```
ldan(E, Sig)
```

Arguments

E	An array. This is the data.
Sig	A list of symmetric positive definite matrices. These are the component covariance matrices.

Author(s)

David Gerard.

`listprod` *Element-wise matrix products between two lists.*

Description

Given two lists of matrices with conformable dimensions, `listprod` returns a list whose elements are the matrix products of the elements of these two lists.

Usage

```
listprod(A, B)
```

Arguments

`A`	A list of matrices.
`B`	A second list of matrices.

Value

A list `C` such that `C[[i]] = A[[i]] %*% B[[i]]`.

Author(s)

David Gerard.

`lq` *LQ decomposition.*

Description

Computes the LQ decomposition of a matrix.

Usage

```
lq(X)
```

Arguments

`X`	A n by p matrix of rank n.

Details

If X is an n by p matrix with $n \leq p$, then `lq` computes the LQ decomposition of X. That is, $X = LQ'$ where Q is p by n with orthonormal columns and L is n by n lower triangular with positive diaognal entries.

Value

`L` An n by n lower triangular matrix with positive diagonal entries.

`Q` An n by p matrix with orthonormal columns.

The returned values satisfy `X = L %*% t(Q)`, up to numerical precision.

Author(s)

David Gerard.

See Also

`qr2` for the related QR decomposition.

Examples

```
X <- matrix(stats::rnorm(12), nrow = 3)
lq_X <- lq(X)
L <- lq_X$L
Q <- lq_X$Q
L
Q
trim(t(Q) %*% Q)
trim(X - L%*%t(Q))
```

`lrt_null_dist_dim_same` *Draw from null distribution of likelihood ratio test statistic.*

Description

When testing for the covariance structure of modes, this function may be used to draw a sample from the null distribution of the likelihood ratio test stistics, whose distribution doesn't depend on any unknown parameters under the null.

Usage

```
lrt_null_dist_dim_same(p, null_ident = NULL, alt_ident = NULL,
  null_diag = NULL, alt_diag = NULL, reference_dist = "normal",
  t_df = NULL, itermax = 100, holq_itermax = 100, holq_tol = 10^-9)
```

Arguments

p	A vector of integers. The dimensions of the array.
null_ident	A vector of integers. The modes that under the null have identity covariance.
alt_ident	A vector of integers. The modes that under the alternative have the identity covariance.
null_diag	A vector of integers. The modes that under the null have diagonal covariance.
alt_diag	A vector of integers. The modes that under the alternative have diagonal covariance.
reference_dist	Two options are supported, 'normal' and 't'. If 't' is specified, you have to specify t_df.
t_df	A numeric. If reference_dist is 't', then this is the degrees of freedom of the t_distribution that the array is distributed under.
itermax	An integer. The number of draws from the null distribution of the likelihood ratio test statistic that is to be performed.
holq_itermax	An integer. The maximum number of block coordinate ascent iterations to perform when calculating the MLE at each step.
holq_tol	A numeric. The stopping criterion when calculating the MLE.

Details

Let $vec(X)$ be $N(0, \Sigma)$. Given two nested hypotheses,

$$H_1 : \Sigma = \Psi_K \otimes \cdots \otimes \Psi_1$$

versus

$$H_0 : \Sigma = \Omega_K \otimes \cdots \otimes \Omega_1,$$

this function will draw from the null distribution of the likelihood ratio test statistic. The possible options are that Ψ_i or Ω_i are the identity matrix, a diagonal matrix, or any positive definite matrix. By default, it's assumed that these matrices are any positive definite matrix.

Unfortunately, this fuction does not support testing for the hypothesis of modeling the covariance between two modes with a single covariance matrix. I might code this up in later versions.

Value

A vector of draws from the null distribution of the likelihood ratio test statistic.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2016). A higher-order LQ decomposition for separable covariance models. *Linear Algebra and its Applications*, 505, 57-84. https://doi.org/10.1016/j.laa.2016.04.033 http://arxiv.org/pdf/1410.1094v1.pdf

See Also

lrt_stat for calculating the likelihood ratio test statistic.

Examples

```
#Test for all identity versus all unconstrained.
p = c(4,4,4)
null1 <- lrt_null_dist_dim_same(p,null_ident = 1:3)

#Generate Null Data
X <- array(stats::rnorm(prod(p)), dim = p)
sig_null <- holq(X, mode_rep = 1:3)$sig
sig_alt <- holq(X)$sig
lrt_x <- lrt_stat(sig_null, sig_alt, p = p)
p_value <- mean(null1 > lrt_x)

hist(null1,main = 'Null Distribution of LRT', xlab = 'LRT Statistic')
abline(v = lrt_x, lty = 2, col = 2, lwd = 2)
legend('topleft', 'Observed LRT Statistic', lty = 2, col = 2, lwd = 2)
mtext(side = 1, paste('P-value = ', round(p_value, digits = 2), sep = ''),
      line = 2)

#-----------------------------------------------------------------------

#Test for all identity versus all mode 1 identity,
#  mode 2 diagonal, mode 3 unconstrained.
p = c(4,4,4)
null2 <- lrt_null_dist_dim_same(p,null_ident = 1:3,
                                alt_ident = 1, alt_diag = 2)

#Generate Null Data
X <- array(stats::rnorm(prod(p)), dim = p)
sig_null <- holq(X, mode_rep = 1:3)$sig
sig_alt <- holq(X, mode_rep = 1, mode_diag = 2)$sig
lrt_x <- lrt_stat(sig_null, sig_alt, p = p)
p_value <- mean(null2 > lrt_x)

hist(null2,main = 'Null Distribution of LRT', xlab = 'LRT Statistic')
abline(v = lrt_x, lty = 2, col = 2, lwd = 2)
legend('topleft', 'Observed LRT Statistic', lty = 2, col = 2, lwd = 2)
mtext(side = 1, paste('P-value = ', round(p_value, digits = 2), sep = ''),
      line = 2)
```

---

lrt_stat *Calculate the likelihood ratio test statistic.*

Description

Calulate the likelihood ratio test statistic for Kronecker structured covariance models.

Usage

```
lrt_stat(sig_null, sig_alt, p)
```

Arguments

sig_null	A numeric. The MLE of the total variation parameter under the null (the standard deviation version).
sig_alt	A numeric. The MLE of the total variation parameter under the alternative (the standard deviation version).
p	A vector of integers. The dimension of the array.

Details

The LRT statistic is the exact same for all elliptically distributed Kronecker structured covariance models (not just the normal). The distribution of the likelihood ratio test statistic does change.

Value

A numeric. The likelihood ratio test statistic.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2016). A higher-order LQ decomposition for separable covariance models. *Linear Algebra and its Applications*, 505, 57-84. https://doi.org/10.1016/j.laa.2016.04.033 http://arxiv.org/pdf/1410.1094v1.pdf

See Also

holq for obtaining the MLE of the total variation parameter.

lrt_null_dist_dim_same for getting the null distribution of the likelihood ratio test statistic.

---

mat *Unfold a matrix.*

Description

mat returns a matrix version of a provided tensor.

Usage

```
mat(A, k)
```

Arguments

A	An array to be unfolded.
k	The mode, or dimension, along which the unfolding is to be applied.

Details

Applies the matrix unfolding operator (also called 'matricization' or 'matrix flattening' operator) on a provided tensor. There are multiple ways one could do this. This function performs the matrix unfolding described in Kolda and Bader (2009).

Value

A matrix whose rows index the kth mode and whose columns index every other mode. The ordering of the columns is in lexicographical order of the indices of the array A.

Author(s)

Peter Hoff.

References

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3), 455-500.

Examples

```
A <- array(1:8, dim = c(2,2,2))
mat(A, 1)
mat(A, 2)
mat(A, 3)
```

---

mhalf *The symmetric square root of a positive definite matrix.*

Description

Returns the unique symmetric positive definite square root matrix of a provided symmetric positive definite matrix.

Usage

```
mhalf(M)
```

Arguments

M	A symmetric positive definite matrix.

Value

The unique symmetric positive definite matrix X such that $XX = M$.

Author(s)

Peter Hoff.

Examples

```
Y <- matrix(stats::rnorm(4), nrow = 2)
M <- Y %*% t(Y)
X <- mhalf(M)
X
identical(M, X %*% X)
```

---

mle_from_holq *Get MLE from output of* holq.

Description

From the output of holq, this function will calculate the MLEs for the component covariance matrices and for the total variation parameter.

Usage

```
mle_from_holq(holq_obj)
```

Arguments

holq_obj	The output returned from holq.

Details

The function simply takes the A[[i]] output of holq and returs A[[i]] %*% t(A[[i]]). The estimate of the total variation parameter is sqrt(sig ^ 2 / prod{p}), whre p is the vector of dimensions of the data array and sig is the output from holq.

Value

cov_mle A list of positive definite matrices. These are the MLEs for the component covariance matrices.

sig_mle A numeric. This is an estimate of the "standard deviation" form of the total variation parameter.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2016). A higher-order LQ decomposition for separable covariance models. *Linear Algebra and its Applications*, 505, 57-84. https://doi.org/10.1016/j.laa.2016.04.033 http://arxiv.org/pdf/1410.1094v1.pdf

See Also

holq.

`multiway_takemura` *Calculate a truncated multiway Takemura estimator.*

Description

This function will 'average' Bayes rules given random rotations of the data array. This 'averaged' estimator has lower risk than the uniformly minimum risk equivariant estimator under a product group of lower triangular matrices. Truncated multiway Takemura's estimator is not equivariant with respect to this product group of lower triangular matrices, but it is an equivariant randomized estimator with respect to a product group of orthogonal matrices.

Usage

```
multiway_takemura(X, ortho_max = 2, mcmc_itermax = 1000,
  start_identity = FALSE, print_mcmc = FALSE, mode_rep = NULL)
```

Arguments

`X`	An array. This is the data array.
`ortho_max`	An integer. The number of 'averagings' to perform.
`mcmc_itermax`	An integer. The number of iterations each MCMC should perform using `equi_mcmc`.
`start_identity`	Should each MCMC start their covariance matrices at the identity (TRUE) or at the sample covariance matrices (FALSE)?
`print_mcmc`	Should the output of the MCMC be printed to the screen (TRUE) or not (FALSE)?
`mode_rep`	A vector of integers. Which mode(s) are considered iid observations? Default is none.

Details

This function will (1) randomly rotate `X` along every mode, then (2) it will calculate the uniformly minimum risk equivariant estimator using `equi_mcmc`, then (3) it will 'average' these estimates.

Value

`B` A list of the truncated multiway Takemura's estimators for each component covariance matrix. Not their Cholesky square roots.

`b` Truncated multiway Takemura's estimator for the total variation parameter. The 'variance' form, not the 'standard devation' form.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

See Also

equi_mcmc, random_ortho.

Examples

```
# Simulate data.
p <- c(5, 5, 5)
X <- array(stats::rnorm(prod(p)), dim = p)
multi_out <- multiway_takemura(X, mode_rep = 3)
multi_out$b
trim(multi_out$B[[1]])
trim(multi_out$B[[2]])
trim(multi_out$B[[3]])
```

---

multi_stein_loss *Calculate multiway Stein's loss from square root matrices.*

Description

Given a list of estimates of the lower-triangular Cholesky square roots of component covariance matrices, a list of true lower-triangular Cholesky square roots of component covariance matrices, an estimate of the total variation, and the true total variation, `multi_stein_loss` will calculate multiway Stein's loss between the estimates and the truth.

Usage

```
multi_stein_loss(B, Psi, b, psi)
```

Arguments

`B`	A list of lower triangular matrices. These are the 'estimates' of the lower-triangular Cholesky square roots of the component covariance matrices.
`Psi`	A list of lower triangular matrices. These are the 'true' lower-triangular Cholesky square roots of the component covariance matrices.
`b`	A numeric. This is an 'estimate' of the total variation parameter, the 'standard devation' version of it.
`psi`	A numeric. This is the 'true' total variation parameter, the 'standard devation' version of it.

Details

Multiway Stein's loss is a generalization of Stein's loss. More details on multiway Stein's loss and the Bayes rules under it can be found in Gerard and Hoff (2015).

The function `multi_stien_loss_cov` also calculates multiway Stein's loss, but uses the component covariance matrices (not the Cholesky roots) as input.

Value

A numeric, the multiway Stein's loss between the 'truth' and the 'estimates'.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

See Also

`multi_stein_loss_cov`, `get_equi_bayes`.

`multi_stein_loss_cov` *Calculate multiway Stein's loss from component covariance matrices.*

Description

Given a list of estimated component covariance matrices, a list of true component covariance matrices, an estimate of the total variation, and the true total variation, `multi_stein_loss_cov` will calculate multiway Stein's loss between the estimates and the truth.

Usage

```
multi_stein_loss_cov(B, Sigma, b, sigma)
```

Arguments

`B`	A list of positive definite matrices. These are the 'estimates' of the component covariance matrices.
`Sigma`	A list of positive definite matrices. These are the 'true' component covariance matrices.
`b`	A numeric. This is an 'estimate' of the total variation parameter, the 'standard devation' version of it.
`sigma`	A numeric. This is the 'true' total variation parameter, the 'standard devation' version of it.

Details

Multiway Stein's loss is a generalization of Stein's loss. More details on multiway Stein's loss and the Bayes rules under it can be found in Gerard and Hoff (2015).

The function `multi_stien_loss` also calculates multiway Stein's loss, but uses the lower-triangular Cholesky square roots of the component covariance matrices as input.

Value

A numeric, the multiway Stein's loss between the 'truth' and the 'estimates'.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

See Also

`multi_stein_loss`, `get_equi_bayes`.

---

polar *The left polar decomposition.*

Description

`polar` calculates the left polar decomposition of a matrix.

Usage

```
polar(X)
```

Arguments

`X`	A matrix.

Details

`polar` Takes a matrix X, of dimensions n by p, and returns two matrices P and Z such that $X = PZ$. P is a symmetric positive definite matrix of dimension n by n and Z is an n by p matrix with orthonormal rows.

Value

P A n by n symmetric positive definite matrix.

Z A n by p matrix with orthonormal rows.

Note that `X == P %*% Z`, up to numerical precision.

Author(s)

David Gerard.

Examples

```
X <- matrix(1:6, nrow = 2)
polar_x <- polar(X)
P <- polar_x$P
Z <- polar_x$Z
P
Z
trim(Z %*% t(Z))
trim(X - P %*% Z)
```

`qr2` *QR Decomposition.*

Description

QR decomposition, constraining the R matrix to have non-negative diagonal entries.

Usage

```
qr2(X)
```

Arguments

`X`	A matrix of dimension n by p where $n \geq p$

Details

This function is almost a wrapper for `qr()`, `qr.R()`, and `qr.Q()`, except it constrains the diagonal elements of `R` to be non-negative. If `X` is full rank with fewer columns than rows, then this is sufficient to gaurantee uniqueness of the QR decomposition (Proposition 5.2 of Eaton (1983)).

Value

Q An n by p matrix with orthonormal columns.

R A p by p upper-triangular matrix with non-negative diagonal elements.

Author(s)

David Gerard.

See Also

qr, qr.Q, and qr.R for the base methods on the obtaining the QR decomposition. lq for the related LQ decomposition.

---

random_ortho *Generate a list of orthogonal matrices drawn from Haar distribution.*

Description

Given a vector p, random_ortho will generate a list ortho_list such that ortho_list[[i]] is a matrix with row and column dimensions p[[i]] and is drawn from the uniform (Haar) distribution over the space of orthogonal matrices.

Usage

```
random_ortho(p)
```

Arguments

p	A vector of dimensions for the matrices.

Details

This function is primarily used by multiway_takemura in its averaging over uniformly minimum risk equivariant estimators under rotations of the data array.

Value

ortho_list A list of orthogonal matrices whose dimensions are given in p.

Author(s)

David Gerard.

See Also

multiway_takemura.

`rmirror_wishart` *Sample from the mirror-Wishart distribution.*

Description

Given scale matrix `Phi` and degrees of freedom `nu`, `rmirror_wishart` will sample from the mirror-Wishart distribution.

Usage

```
rmirror_wishart(nu, Phi)
```

Arguments

`nu`	An integer. The degrees of freedom in the mirror-Wishart.
`Phi`	A matrix. The scale matrix of the mirror-Wishart.

Details

S is mirror-Wishart(ν, Φ) if

$$S = UV'VU',$$

where VV' is the lower triangular Cholesky decomposition of a Wishart(ν, I)-distributed random matrix and UU' is the upper triangular Cholesky decomposition of Φ. That is, V is lower triangular and U is upper triangular. For details on its applications, see Gerard and Hoff (2015).

Value

A matrix drawn from the mirror-Wishart distribution with `nu` degrees of freedom and scale matrix `Phi`.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

See Also

`sample_right_wishart`

rmvnorm *Multivariate normal simulation.*

Description

Simulate a multivariate normal random matrix.

Usage

```
rmvnorm(n, mu, Sigma, Sigma.chol = chol(Sigma))
```

Arguments

n	number of mvnormal vectors to simulate.
mu	mean vector.
Sigma	covariance matrix.
Sigma.chol	Cholesky decomposition of Sigma.

Details

This function simulates multivariate normal random vectors.

Author(s)

Peter Hoff.

Examples

```
# Simulate several matrices and compute the mean.
Y <- tensr:::rmvnorm(100, c(1, 2, 3), matrix(c(3, 0, 1, 0, 1, -1, 1, -1, 2), 3, 3))
colMeans(Y)
cov(Y)
```

rsan *Standard normal array.*

Description

Generate an array of iid standard normal variables.

Usage

```
rsan(dim)
```

Arguments

dim	a vector of positive integers.

Details

This functions generates an array of dimension dim filled with iid standard normal variables.

Author(s)

Peter Hoff.

Examples

```
tensr:::rsan(c(5,4,3))
```

---

rwish *Wishart simulation.*

Description

Simulate a Wishart-distributed random matrix.

Usage

```
rwish(S0, nu = dim(as.matrix(S0))[1] + 1)
```

Arguments

S0	a positive definite matrix.
nu	a positive scalar.

Details

This function simulates a Wishart random matrix using Bartletts decomposition, as described in Everson and Morris (2000).

Author(s)

Peter Hoff.

Examples

```
# simulate several matrices and compute the mean.
SS <- matrix(0, 5, 5)
for(s in 1:1000) { SS <- SS + tensr:::rwish(diag(5), 3) }
SS / s
```

sample_right_wishart *Gibbs update of* Phi_inv.

Description

Samples an upper triangular Cholesky square root of a mirror-Wishart distributed random variable.

Usage

```
sample_right_wishart(nu, V)
```

Arguments

nu	A numeric. The degrees of freedom in the mirror-Wishart.
V	A matrix. The inverse of the scale matrix in the mirror-Wishart.

Details

Let X be mirror-Wishart(ν, $V^{-}1$). Then This code returns an upper triangular C where $X = CC'$. This function is used primarily during the Gibbs updates of the inverse of the lower triangular Cholesky square root of the component covariance matrices in equi_mcmc.

Value

C An upper triangular matrix such that C %*% t(C) is a sample from the mirror-Wishart(nu, V ^ -1) distribution.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

See Also

equi_mcmc, rmirror_wishart.

`sample_sig` *Update for total variation parameter in* `equi_mcmc`.

Description

Samples from the square root of an inverse-gamma.

Usage

```
sample_sig(X, phi_inv)
```

Arguments

`X`	An array. The tensor data.
`phi_inv`	A list of the current values of inverse of the lower-triangular Cholesky square root of the the component covariance matrices. This is equivalent to the transpose of the upper-triangular Cholesky square root of the inverse component covariance matrices. `phi_inv[[i]]` is a lower triangluar matrix where `solve(phi_inv[[i]]) %*% t(solve(phi_inv[[i]])` is the current estimate of the ith component covariance matrix.

Details

This function provides a Gibbs update for the total variation parameter from the MCMC implemented in `equi_mcmc`. This corresponds to the square root of an inverse-gamma distributed random variable whose parameters depend on the data and the component covariance matrices. Roughly, this is the update for the standard deviation, not the variance.

Value

A numeric. The update for the total variation parameter in the MCMC implemented in `equi_bayes`.

Author(s)

David Gerard.

References

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

See Also

`equi_mcmc` for a Gibbs sampler where this function is used.

`start_ident` *Get list of identity matrices.*

Description

Will provide a list of identity matrices for the specified modes.

Usage

```
start_ident(p, modes = NULL)
```

Arguments

`p`	A vector of integers. This is the dimension of the array and the length of the list to be created.
`modes`	A vector of integers. These are the indices in the list to be given an identity matrix.

Details

Given a vector of dimensions `p` and a vector indicating which modes will get an identity matrix `modes`, this function will return a list `start_vals` where `start_vals[[i]]` is the identity matrix of dimensions `p[i]` if `i` is in `modes` and will be `NULL` otherwise.

This is primarily used when getting starting values in `equi_mcmc`.

Value

`start_vals` A list of identity matrices and NULL values.

Author(s)

David Gerard.

See Also

`equi_mcmc`.

`start_resids` *Sample covariance matrices for each mode.*

Description

Scaled Cholesky square roots of the sample covariance matrix and its inverse.

Usage

```
start_resids(Y, mode_rep = NULL)
```

Arguments

`Y`	An array of numeric data.
`mode_rep`	A vector of integers. The modes specified by `mode_rep` will be given an identity matrix instead of a sample-based matrix.

Details

This function will take the sample covariance matrix of the ith matricization of an input array Y and will return (1) its lower-triangular Cholesky square root scaled down to have determinant 1 and (2) the inverse of its lower-triangular Cholesky square root scaled down to have determinant 1. This function is primarily used to obtain starting values for the Gibbs sampler implemented in `equi_mcmc`.

Value

`Sig` A list where `Sig[[i]]` is the lower-triangular Cholesky square root of the sample covariance matrix of the ith mode, scaled down to have determinant 1.

`Sig_inv` A list where `Sig_inv[[i]]` is the inverse of the lower-triangular Cholesky square root of the sample covariance matrix of the ith mode, scaled down to have determinant 1.

If `mode_rep` is not `NULL`, then the list elements in `Sig` and `Sig_inv` specified in `mode_rep` will be the identity matrix instead of sample-based matrices.

Author(s)

David Gerard.

See Also

`equi_mcmc`.

tensr *tensr: A package for Kronecker structured covariance inference.*

Description

This package provides a collection of functions for likelihood and equivariant inference for covariance matrices under the array normal model. Also included are functions for calculating tensor decompositions that are related to likelihood inference in the array normal model.

Introduction

Let X be a multidimensional array (also called a tensor) of K dimensions. This package provides a series of functions to perform statistical inference when

$$vec(X) \sim N(0, \Sigma),$$

where Σ is assumed to be Kronecker structured. That is, Σ is the Kronecker product of K covariance matrices, each of which has the interpretation of being the covariance of X along its kth mode, or dimension.

Pay particular attention to the zero mean assumption. That is, you need to de-mean your data prior to applying these functions. If you have more than one sample, X_i for $i = 1, \ldots, n$, then you can concatenate these tensors along a $(K+1)$th mode to form a new tensor Y and apply the `demean_tensor()` function to Y which will return a tensor that satisfies the mean-zero assumption.

The details of the methods in this package can be found in Gerard and Hoff (2015) and Gerard and Hoff (2016).

Tensr functions

`amprod` k-mode product.

`anorm_cd` Array normal conditional distributions.

`array_bic_aic` Calculate the AIC and BIC.

`arrIndices` Array indices.

`atrans` Tucker product.

`collapse_mode` Collapse multiple modes into one mode.

`convert_cov` Convert the output from `equi_mcmc` to component covariance matrices.

`demean_tensor` Demeans array data.

`equi_mcmc` Gibbs sampler using an invariant prior.

`fnorm` Frobenius norm of an array.

`get_equi_bayes` Get the Bayes rule under multiway Stein's loss.

`get_isvd` Calculate the incredible SVD (ISVD).

`holq` Calculate the incredible higher-order LQ decomposition (HOLQ).

`hooi` Calculate the higher-order orthogonal iteration (HOOI).

hosvd Calculate the (truncated) higher-order SVD (HOSVD).

Kom Commutation matrix.

ihop The incredible higher-order polar decomposition (IHOP).

ldan Log-likelihood of array normal model.

listprod Element-wise matrix products between two lists.

lq LQ decomposition.

lrt_null_dist_dim_same Draw from null distribution of likelihood ratio test statistic.

lrt_stat Calculate the likelihood ratio test statistic.

mat Unfold a matrix.

mhalf The symmetric square root of a positive definite matrix.

mle_from_holq Get MLE from output of holq.

multi_stein_loss Calculate multiway Stein's loss from square root matrices.

multi_stein_loss_cov Calculate multiway Stein's loss from component covariance matrices.

multiway_takemura Calculate a truncated multiway Takemura estimator.

polar The left polar decomposition.

qr2 QR Decomposition.

random_ortho Generate a list of orthogonal matrices drawn from Haar distribution.

rmirror_wishart Sample from the mirror-Wishart distribution.

sample_sig Update for total variation parameter in equi_mcmc.

sample_right_wishart Gibbs update of Phi_inv.

start_ident Get list of identity matrices.

start_resids Sample covariance matrices for each mode.

tsum Tucker sum.

tr Trace of a matrix.

trim Truncates small numbers to 0.

References

Gerard, D., & Hoff, P. (2016). A higher-order LQ decomposition for separable covariance models. *Linear Algebra and its Applications*, 505, 57-84. https://doi.org/10.1016/j.laa.2016.04.033 http://arxiv.org/pdf/1410.1094v1.pdf

Gerard, D., & Hoff, P. (2015). Equivariant minimax dominators of the MLE in the array normal model. *Journal of Multivariate Analysis*, 137, 32-49. https://doi.org/10.1016/j.jmva.2015.01.020 http://arxiv.org/pdf/1408.0424.pdf

topK *Top K elements of a vector.*

Description

Identify top K elements of a vector.

Usage

```
topK(x, K = 1, ignoreties = TRUE)
```

Arguments

x	The vector.
K	The number of indices to return.
ignoreties	If FALSE, will return a vector of the indices whose elements are greater than or equal to the Kth largest element, resulting in a vector possibly of length greater than K in the case of ties.

Details

This function returns the indices corresponding to the top elements of a vector.

Author(s)

Peter Hoff.

Examples

```
x <- c(3, 6, 2, 4, 1)
tensr:::topK(x, 3)
```

tr *Trace of a matrix.*

Description

Returns the sum of the diagonal elements of a matrix.

Usage

```
tr(X)
```

Arguments

X	A matrix whose diagonal elements will be added together.

Details

This returns the trace of a matrix, which is just the sum of its diagonal elements.

Value

The sum of the diagonal elements of X.

Author(s)

Peter Hoff.

Examples

```
X <- matrix(1:4, nrow = 2, ncol = 2)
X
tr(X)
```

`trim` *Truncates small numbers to 0.*

Description

Given an array, matrix, or vector, `trim` will truncate all elements smaller than `epsilon` (in absolute value) to zero.

Usage

```
trim(X, epsilon = 10^-6)
```

Arguments

`X`	An array, a matrix, or a vector.
`epsilon`	A numeric.

Details

All elements in `X` that are smaller than `epsilon` (in absolute value) will be set to zero then returned.

Author(s)

David Gerard.

Examples

```
X <- c(0, 1, 10^-7, -1, -10^-7)
X
trim(X)
```

tsum *Tucker sum.*

Description

Computes the Tucker sum of an array and a list of matrices.

Usage

```
tsum(X, A)
```

Arguments

X	A real array.
A	A list of real matrices.

zscores *Normal scores.*

Description

This function applies a quantile-quantile transformation to the data, resulting in a distribution that is approximately normal but has the same ranks as the original data.

Usage

```
zscores(y, ties.method = "average")
```

Arguments

y	A vector.
ties.method	The option ties.method in the rank function.

Value

A vector of the same length as y.

Author(s)

Peter Hoff.

Examples

```
y <- rexp(100)
z <- tensr:::zscores(y)
par(mfrow = c(1, 3))
hist(y)
hist(z)
plot(y,z)
```

Index