Package ‘snQTL’

April 29, 2025

Title Spectral Network Quantitative Trait Loci (snQTL) Analysis

Version 0.2

Date 2025-04-05

Maintainer Jiaxin Hu <hjx_2025@outlook.com>

Author Jiaxin Hu [aut, cre, cph],
Miaoyan Wang [aut, cph]

Description A spectral framework to map quantitative trait loci (QTLs) affecting joint differential networks of gene co-Expression. Test the equivalence among multiple biological networks via spectral statistics. See reference Hu, J., Weber, J. N., Fuess, L. E., Steinel, N. C., Bolnick, D. I., & Wang, M. (2025) <doi:10.1371/journal.pcbi.1012953>.

License GPL (>= 2)

Encoding UTF-8

NeedsCompilation no

Imports rARPACK, MASS, methods

Depends R (>= 3.5.0)

RoxygenNote 7.2.3

Repository CRAN

Date/Publication 2025-04-29 08:10:01 UTC

Contents

`as.tensor` *Tensor Conversion*

Description

Create a `Tensor-class` object from an `array`, `matrix`, or `vector`.

Usage

```
as.tensor(x, drop = FALSE)
```

Arguments

`x`	an instance of `array`, `matrix`, or `vector`
`drop`	whether or not modes of 1 should be dropped

Value

a `Tensor-class` object

Examples

```
#From vector
vec <- runif(100); vecT <- as.tensor(vec); vecT
#From matrix
mat <- matrix(runif(1000),nrow=100,ncol=10)
matT <- as.tensor(mat); matT
#From array
indices <- c(10,20,30,40)
arr <- array(runif(prod(indices)), dim = indices)
arrT <- as.tensor(arr); arrT
```

`BinarySearch` *Search soft threshold*

Description

A binary search to find proper soft threshold `lamv` such that

$$sv = soft(argv, lamv)/||soft(argv, lamv)||_2, ||sv||_1 = sumabsv$$

Usage

```
BinarySearch(argv, sumabsv, maxiter = 150)
```

Arguments

`argv`	the vector to be soft thresholded
`sumabsv`	upperbound of the L_1 norm of sv
`maxiter`	max iteration to perform binary search

Value

the proper threshold level `lamv`.

See Also

`symmPMD()`.

`cs_unfold-methods` *Tensor Column Space Unfolding*

Description

Tensor Column Space Unfolding

Usage

```
cs_unfold(tnsr, m)

## S4 method for signature 'Tensor'
cs_unfold(tnsr, m = NULL)
```

Arguments

`tnsr`	Tensor instance
`m`	mode to be unfolded on

Details

```
cs_unfold(tnsr,m=NULL)
```

`diffnet_to_snQTL_stats` *Test statistics for snQTL*

Description

Generate snQTL test statistics from a given list of differential networks. This function takes a list of differential networks, the choice of test statistics, and other computational tuning parameters as inputs. Outputs include the calculated statistics, recall of the choice, and the decomposition components associated with the statistics.

Usage

```
diffnet_to_snQTL_stats(
  diffnet_list,
  method = c("sum", "sum_square", "max", "tensor"),
  rho = 1000,
  sumabs = 0.2,
  niter = 20,
  trace = FALSE,
  tensor_iter = 20,
  tensor_tol = 10^(-3),
  tensor_seed = NULL
)
```

Arguments

`diffnet_list`	list, a list of p-by-p differential networks
`method`	character, the choice of test statistics; see "details"
`rho`	number, a large positive constant adding to the diagonal elements to ensure positive definiteness in symmetric matrix spectral decomposition
`sumabs`	number, the number specify the sparsity level in the matrix/tensor eigenvector; `sumabs` takes value between $1/sqrt(p)$ and 1, where p is the dimension; `sumabs`$*sqrt(p)$ is the upperbound of the L1 norm of the leading matrix/tensor eigenvector (see `symmPMD()`)
`niter`	integer, the number of iterations to use in the PMD algorithm (see `symmPMD()`)
`trace`	logic variable, whether to trace the progress of PMD algorithm (see `symmPMD()`)
`tensor_iter`	integer, the maximal number of iteration in SSTD algorithm (see `max_iter` in `SSTD()`)
`tensor_tol`	number, a small positive constant for error difference to indicate the SSTD convergence (see `tol` in `SSTD()`)
`tensor_seed`	number, the seed to generate random initialization for SSTD algorithm

Details

The list `diffnet_list` records the pairwise differential networks D_{AB}, D_{AH}, D_{AB}. This package provides four options for test statistics:

1. sum, the sum of sparse leading matrix eigenvalues (sLMEs) of all pairwise differential networks:

   $$Stat_sum = \lambda(D_{AB}) + \lambda(D_{AH}) + \lambda(D_{BH}),$$

   where λ refers to the sLME operation with given sparsity level set up by `sumabs`.
2. sum_square, the sum of squared sLMEs:

   $$Stat_sumsquare = \lambda^2(D_{AB}) + \lambda^2(D_{AH}) + \lambda^2(D_{BH}).$$
3. max, the maximal of sLMEs:

   $$Stat_max = \max(\lambda(D_{AB}), \lambda(D_{AH}), \lambda(D_{BH})).$$
4. tensor, the sparse leading tensor eigenvalue (sLTE) of the differential tensor:

   $$Stat_tensor = \Lambda(\mathcal{D}),$$

   where Λ refers to the sLTE operation with given sparsity level set up by `sumabs`, and $\mathcal{D}$ is the differential tensor composed by stacking three pairwise differential networks.

The sparse symmetric matrix decomposition is implemented by `symmPMD()` with parameters `rho, sumabs, niter, trace`. The sparse symmetric tensor decomposition is implemented by `SSTD()`. Since `symmPMD()` is used in `SSTD()`, parameters for `symmPMD()` are used for `SSTD()`. While parameters `tensor_iter, tensor_tol, tensor_seed` should be uniquely defined for `tensor` method.

Value

a list containing the following:

`method`	character, recall of the choice of test statistics
`stats`	number, the calculated test statistics with given network list and choices
`decomp_result`	list, if `method = c("sum", "sum_square", "max")`, the matrix decomposition components for all pairwise differential networks are recorded; if `method = "tensor"`, the tensor decomposition components for the differential tensor are recorded

References

Hu, J., Weber, J. N., Fuess, L. E., Steinel, N. C., Bolnick, D. I., & Wang, M. (2025). A spectral framework to map QTLs affecting joint differential networks of gene co-expression. PLOS Computational Biology, 21(4), e1012953.

`dim-methods` *Mode Getter for Tensor*

Description

Return the vector of modes from a tensor

Usage

```
## S4 method for signature 'Tensor'
dim(x)
```

Arguments

`x`	the Tensor instance

Details

```
dim(x)
```

Value

an integer vector of the modes associated with `x`

Examples

```
tnsr <- rand_tensor()
dim(tnsr)
```

`fold` *General Folding of Matrix*

Description

General folding of a matrix into a Tensor. This is designed to be the inverse function to `unfold-methods`, with the same ordering of the indices. This amounts to following: if we were to unfold a Tensor using a set of `row_idx` and `col_idx`, then we can fold the resulting matrix back into the original Tensor using the same `row_idx` and `col_idx`.

Usage

```
fold(mat, row_idx = NULL, col_idx = NULL, modes = NULL)
```

Arguments

`mat`	matrix to be folded into a Tensor
`row_idx`	the indices of the modes that are mapped onto the row space
`col_idx`	the indices of the modes that are mapped onto the column space
`modes`	the modes of the output Tensor

Details

This function uses `aperm` as the primary workhorse.

Value

Tensor object with modes given by `modes`

References

T. Kolda, B. Bader, "Tensor decomposition and applications". SIAM Applied Mathematics and Applications 2009, Vol. 51, No. 3 (September 2009), pp. 455-500. URL: https://www.jstor.org/stable/25662308.

See Also

`unfold-methods`

Examples

```
tnsr <- new('Tensor',3L,c(3L,4L,5L),data=runif(60))
matT3<-unfold(tnsr,row_idx=2,col_idx=c(3,1))
identical(fold(matT3,row_idx=2,col_idx=c(3,1),modes=c(3,4,5)),tnsr)
```

`get_diffnet_from_exp` *The differential matrix*

Description

Given observations from two populations X and Y, compute the differential matrix

$$D = N(Y) - N(X)$$

where N() is the covariance matrix, or the weighted adjacency matrices defined as

$$N_{ij} = |corr(i,j)|^beta$$

for some constant beta > 0, 1 <= i, j <= p. Let N represent the regular correlation matrix when beta=0, and covariance matrix when beta<0.

Usage

```
get_diffnet_from_exp(X, Y, adj.beta = -1, trans = FALSE, location = NULL)
```

Arguments

`X`	n1-by-p matrix for samples from the first population. Rows are samples/observations, while columns are the features.
`Y`	n2-by-p matrix for samples from the second population. Rows are samples/observations, while columns are the features.
`adj.beta`	Power to transform correlation matrices to weighted adjacency matrices by $N_{ij} = \|r_i j\|^a dj.beta$ where $r_i j$ represents the Pearson's correlation. When `adj.beta=0`, the correlation marix is used. When `adj.beta<0`, the covariance matrix is used. The default value is `adj.beta=-1`.
`trans`	logic variable, whether to only consider the trans-correlation (between genes from two different chromosomes or regions); see "details"
`location`	vector, the (chromosome) locations for items

Value

The p-by-p differential matrix $D = N(Y) - N(X)$.

`get_diffnet_list_from_exp`

Get the list of differential matrix from a list of expression data

---

Description

Given a list of expression data, $X_1, ..., X_K$, compute the list of differential matrix

$$D^{(k,l)} = N(X_l) - N(X_k), k < l,$$

where N() is the covariance matrix, or the weighted adjacency matrices defined as

$$N_{ij} = |corr(i,j)|^b eta$$

for some constant beta > 0, 1 <= i, j <= p. Let N represent the regular correlation matrix when beta=0, and covariance matrix when beta<0. In total, we will have K*(K-1)/2 pairwise differential networks in the list.

If `trans = TRUE`, we let $N_{ij} = 0$ if i, j are from the same region based on `location`. Under gene co-expression context, trans-correlation usually refer to the correlation between two genes i, j from two chromosomes.

Usage

```
get_diffnet_list_from_exp(
  exp_list,
  adj.beta = -1,
  trans = FALSE,
  location = NULL
)
```

Arguments

`exp_list`	a list of nk-by-p matrices from the K populations. Rows are samples/observations, while columns are the features.
`adj.beta`	Power to transform correlation matrices to weighted adjacency matrices by $N_{ij} = \|r_i j\|^a dj.beta$ where $r_i j$ represents the Pearson's correlation. When `adj.beta=0`, the correlation marix is used. When `adj.beta<0`, the covariance matrix is used. The default value is `adj.beta=-1`.
`trans`	logic variable, whether to only consider the trans-correlation (between genes from two different chromosomes or regions)
`location`	vector, the (chromosome) locations for items

Value

A list of p-by-p differential matrix $D^{(k,l)}, k < l$.

`kronecker_list` *List Kronecker Product*

Description

Returns the Kronecker product from a list of matrices or vectors. Commonly used for n-mode products and various Tensor decompositions.

Usage

```
kronecker_list(L)
```

Arguments

`L`	list of matrices or vectors

Value

matrix that is the Kronecker product

Examples

```
smalllizt <- list('mat1' = matrix(runif(12),ncol=4),
'mat2' = matrix(runif(12),ncol=4),
'mat3' = matrix(runif(12),ncol=4))
dim(kronecker_list(smalllizt))
```

`l2n` *L2 norm for vector*

Description

L2 norm for vector

Usage

```
l2n(vec)
```

Arguments

`vec`	a numeric vector

Value

the L2 norm of `vec`.

`Ops-methods` *Conformable elementwise operators for Tensor*

Description

Conformable elementwise operators for Tensor

Usage

```
## S4 method for signature 'Tensor,Tensor'
Ops(e1, e2)
```

Arguments

`e1`	left-hand object
`e2`	right-hand object

Examples

```
tnsr <- rand_tensor(c(3,4,5))
tnsr2 <- rand_tensor(c(3,4,5))
tnsrsum <- tnsr + tnsr2
tnsrdiff <- tnsr - tnsr2
tnsrelemprod <- tnsr * tnsr2
tnsrelemquot <- tnsr / tnsr2
for (i in 1:3L){
for (j in 1:4L){
```

```
for (k in 1:5L){
stopifnot(tnsrsum@data[i,j,k]==tnsr@data[i,j,k]+tnsr2@data[i,j,k])
stopifnot(tnsrdiff@data[i,j,k]==(tnsr@data[i,j,k]-tnsr2@data[i,j,k]))
stopifnot(tnsrelemprod@data[i,j,k]==tnsr@data[i,j,k]*tnsr2@data[i,j,k])
stopifnot(tnsrelemquot@data[i,j,k]==tnsr@data[i,j,k]/tnsr2@data[i,j,k])
}
}
}
```

rand_tensor *Tensor with Random Entries*

Description

Generate a Tensor with specified modes with iid normal(0,1) entries.

Usage

```
rand_tensor(modes = c(3, 4, 5), drop = FALSE)
```

Arguments

modes	the modes of the output Tensor
drop	whether or not modes equal to 1 should be dropped

Value

a Tensor object with modes given by modes

Note

Default rand_tensor() generates a 3-Tensor with modes c(3,4,5).

Examples

```
rand_tensor()
rand_tensor(c(4,4,4))
rand_tensor(c(10,2,1),TRUE)
```

`rs_unfold-methods` *Tensor Row Space Unfolding*

Description

Tensor Row Space Unfolding

Usage

```
rs_unfold(tnsr, m)

## S4 method for signature 'Tensor'
rs_unfold(tnsr, m = NULL)
```

Arguments

`tnsr`	Tensor instance
`m`	mode to be unfolded on

Details

```
rs_unfold(tnsr,m=NULL)
```

`single_exp_to_snQTL_stats` *Generate one single snQTL test statistics from expression data*

Description

Generate one single snQTL test statistics from a given list of expression data. This function takes a list of expression data, the choice of test statistics, the choice to permute or not, the choice of considering trans-correlation or not, and other computational tuning parameters as inputs. Outputs include the calculated statistics, recall of the choices, and the decomposition components associated with the statistics.

Usage

```
single_exp_to_snQTL_stats(
  seed = NULL,
  permute = FALSE,
  exp_list,
  method = c("sum", "sum_square", "max", "tensor"),
  rho = 1000,
  sumabs = 0.2,
  niter = 20,
  trace = FALSE,
  adj.beta = -1,
  tensor_iter = 20,
  tensor_tol = 10^(-3),
  trans = FALSE,
  location = NULL
)
```

Arguments

`seed`	number, the random seed to shuffle the expression data if `permute = TRUE` and for `SSTD()` initialization if `method = "tensor"`
`permute`	logic variable, whether to shuffle the samples in expression data; see "details"
`exp_list`	list, a list of expression data from samples with different genotypes; see "details"
`method`	character, the choice of test statistics (see `net_to_stats()`)
`rho`	number, a large positive constant adding to the diagonal elements to ensure positive definiteness in symmetric matrix spectral decomposition
`sumabs`	number, the number specify the sparsity level in the matrix/tensor eigenvector; `sumabs` takes value between $1/sqrt(p)$ and 1, where p is the dimension; `sumabs`$*sqrt(p)$ is the upperbound of the L1 norm of the leading matrix/tensor eigenvector (see `symmPMD()`)
`niter`	integer, the number of iterations to use in the PMD algorithm (see `symmPMD()`)
`trace`	logic variable, whether to trace the progress of PMD algorithm (see `symmPMD()`)
`adj.beta`	number, the power transformation to the correlation matrices (see `getDiffMatrix()`); particularly, when `adj.beta=0`, the correlation matrix is used, when `adj.beta<0`, the covariance matrix is used.
`tensor_iter`	integer, the maximal number of iteration in SSTD algorithm (see `max_iter` in `SSTD()`)
`tensor_tol`	number, a small positive constant for error difference to indicate the SSTD convergence (see `tol` in `SSTD()`)
`trans`	logic variable, whether to only consider the trans-correlation (between genes from two different chromosomes or regions); see "details"
`location`	vector, the (chromosome) locations for genes if `trans = TRUE`

Details

In `exp_list`, the dimensions for data matrices are n1-by-p, n2-by-p, and n3-by-p, respectively. The expression data is usually normalized. We use expression data to generate the Pearson's correlation co-expression networks.

If `permute = TRUE`, we shuffle the samples in three expression matrices while keeping the same dimensions. The test statistics from randomly shuffled data are considered as the statistics from null distribution.

If `trans = TRUE`, we only consider the trans-correlation between the genes from two different chromosomes or regions in co-expression networks. The entries in correlation matrices $N_{ij} = 0$ if gene i and gene j are from the same chromosome or region.

Value

a list containing the following:

`method`	character, recall of the choice of test statistics
`permute`	logic variable, recall of the choice of permutation
`stats`	number, the calculated test statistics with given expression list and choices
`decomp_result`	list, if `method = c("sum", "sum_square", "max")`, the matrix decomposition components for all pairwise differential networks are recorded; if `method = "tensor"`, the tensor decomposition components for the differential tensor are recorded

References

Hu, J., Weber, J. N., Fuess, L. E., Steinel, N. C., Bolnick, D. I., & Wang, M. (2025). A spectral framework to map QTLs affecting joint differential networks of gene co-expression. PLOS Computational Biology, 21(4), e1012953.

`sLME` *Calculate of sLME for matrices*

Description

Calculate the sLME given a matrix D. For any symmetric matrix D, sLME test statistic is defined as

$$maxsEig(D), sEig(-D)$$

where `sEig()` is the sparse leading eigenvalue, defined as

$$max_v v^T Av$$

subject to $||v||_2 \leq 1, ||v||_1 \leq s$.

Usage

```
sLME(Dmat, rho = 1000, sumabs.seq = 0.2, niter = 20, trace = FALSE)
```

Arguments

`Dmat`	p-by-p numeric matrix, the differential matrix
`rho`	a large positive constant such that $D+diag(rep(rho,p))$ and $-D+diag(rep(rho,p))$ are positive definite.
`sumabs.seq`	a numeric vector specifing the sequence of sparsity parameters, each between $1/sqrt(p)$ and 1. Each sumabs*$\sqrt{p}$ is the upperbound of the L_1 norm of leading sparse eigenvector v.
`niter`	the number of iterations to use in the PMD algorithm (see `symmPMD()`)
`trace`	whether to trace the progress of PMD algorithm (see `symmPMD()`)

Value

A list containing the following components:

`sumabs.seq`	the sequence of sparsity parameters
`rho`	a positive constant to augment the diagonal of the differential matrix such that $D + rho * I$ becomes positive definite.
`stats`	a numeric vector of test statistics when using different sparsity parameters (corresponding to `sumabs.seq`).
`sign`	a vector of signs when using different sparsity parameters (corresponding to `sumabs.seq`). Sign is "pos" if the test statistic is given by sEig(D), and "neg" if is given by sEig(-D), where `sEig` denotes the sparse leading eigenvalue.
`v`	the sequence of sparse leading eigenvectors, each row corresponds to one sparsity parameter given by `sumabs.seq`.
`leverage`	the leverage score for genes (defined as v^2 element-wise) using different sparsity parameters. Each row corresponds to one sparsity parameter given by `sumabs.seq`.

References

Zhu, Lingxue, et al. "Testing high-dimensional covariance matrices, with application to detecting schizophrenia risk genes." The annals of applied statistics 11.3 (2017): 1810.

`snQTL_test_corrnet` *Spectral network quantitative trait loci (snQTL) test*

Description

Spectral framework to detect network QTLs affecting the co-expression networks. This is the main function for snQTL test.

Given a list of expression data matrices from samples with different gentoypes, we test whether there are significant difference among three co-expression networks. Statistically, we consider the hypothesis testing task:

$$H_0 : N_A = N_B = N_H,$$

where A, B, H refer to different genotypes, N refers to the adjacency matrices corresponding to the co-expression network.

We provide four options for the test statistics, composed by sparse matrix/tensor eigenvalues. We perform permutation test to obtain the empirical p-values for the hypothesis testing.

NOTE: This function is also applicable for generalized cases to compare multiple (K > 3) biological networks. Instead of separating the samples by genotypes, people can separate the samples into K groups based on other interested metrics, e.g., locations, treatments. The generalized hypothesis testing problem becomes

$$H_0 : N_1 = ... = N_K,$$

where N_k refers to the correlation-based network corresponding to the group k. For consistency, we stick with the original genotype-based setting in this help document. See details and examples for the generalization on the Github manual https://github.com/Marchhu36/snQTL.

Usage

```
snQTL_test_corrnet(
  exp_list,
  method = c("sum", "sum_square", "max", "tensor"),
  npermute = 100,
  seeds = 1:100,
  stats_seed = NULL,
  rho = 1000,
  sumabs = 0.2,
  niter = 20,
  trace = FALSE,
  adj.beta = -1,
  tensor_iter = 20,
  tensor_tol = 10^(-3),
  trans = FALSE,
  location = NULL
)
```

Arguments

`exp_list`	list, a list of expression data from samples with different genotypes; the dimensions for data matrices are n1-by-p, n2-by-p, and n3-by-p, respectively; see "details"
`method`	character, the choice of test statistics; see "details"
`npermute`	number, the number of permutations to obtain empirical p-values
`seeds`	vector, the random seeds for permutation; length of the vector is equal to the `npermute`
`stats_seed`	number, the random seed for test statistics calculation with non-permuted data
`rho`	number, a large positive constant adding to the diagonal elements to ensure positive definiteness in symmetric matrix spectral decomposition
`sumabs`	number, the number specify the sparsity level in the matrix/tensor eigenvector; `sumabs` takes value between $1/sqrt(p)$ and 1, where p is the dimension; `sumabs`$*sqrt(p)$ is the upperbound of the L1 norm of the leading matrix/tensor eigenvector (see `symmPMD()`)
`niter`	integer, the number of iterations to use in the PMD algorithm (see `symmPMD()`)
`trace`	logic variable, whether to trace the progress of PMD algorithm (see `symmPMD()`)
`adj.beta`	number, the power transformation to the correlation matrices (see `getDiffMatrix()`); particularly, when `adj.beta=0`, the correlation matrix is used, when `adj.beta<0`, the covariance matrix is used.
`tensor_iter`	integer, the maximal number of iteration in SSTD algorithm (see `max_iter` in `SSTD()`)
`tensor_tol`	number, a small positive constant for error difference to indicate the SSTD convergence (see `tol` in `SSTD()`)
`trans`	logic variable, whether to only consider the trans-correlation (between genes from two different chromosomes or regions); see "details"
`location`	vector, the (chromosome) locations for genes if `trans = TRUE`

Details

In `exp_list`, the data matrices are usually ordered with marker's genotypes AA, BB, and AB. The expression data is usually normalized. We use expression data to generate the Pearson's correlation co-expression networks.

Given the list of co-expression networks, we generate pairwise differential networks

$$D_{AB} = N_A - N_B, D_{AH} = N_H - N_A, D_{BH} = N_H - N_B.$$

We use pairwise differential networks to generate the snQTL test statistics.

We provide four options of test statistics with different choices of `method`:

1. sum, the sum of sparse leading matrix eigenvalues (sLMEs) of all pairwise differential networks:

   $$Stat_sum = \lambda(D_{AB}) + \lambda(D_{AH}) + \lambda(D_{BH}),$$

   where λ refers to the sLME operation with given sparsity level set up by `sumabs`.
2. sum_square, the sum of squared sLMEs:

   $$Stat_sumsquare = \lambda^2(D_{AB}) + \lambda^2(D_{AH}) + \lambda^2(D_{BH}).$$

3. max, the maximal of sLMEs:

   $$Stat_max = \max(\lambda(D_{AB}), \lambda(D_{AH}), \lambda(D_{BH})).$$

4. tensor, the sparse leading tensor eigenvalue (sLTE) of the differential tensor:

   $$Stat_tensor = \Lambda(\mathcal{D}),$$

   where Λ refers to the sLTE operation with given sparsity level set up by `sumabs`, and $\mathcal{D}$ is the differential tensor composed by stacking three pairwise differential networks.

Additionally, if `trans = TRUE`, we only consider the trans-correlation between the genes from two different chromosomes or regions in co-expression networks. The entries in correlation matrices $N_{ij} = 0$ if gene i and gene j are from the same chromosome or region. The gene location information is required if `trans = TRUE`.

Value

a list containing the following:

`method`	character, recall of the choice of test statistics
`res_original`	list, test result for non-permuted data, including the recall of method choices, test statistics, and decomposition components
`res_permute`	list, test results for each permuted data, including the recall of method choices, test statistics, and decomposition components
`emp_p_value`	number, the empirical p-value from permutation test

References

Hu, J., Weber, J. N., Fuess, L. E., Steinel, N. C., Bolnick, D. I., & Wang, M. (2025). A spectral framework to map QTLs affecting joint differential networks of gene co-expression. PLOS Computational Biology, 21(4), e1012953.

Examples

```
### artificial example
n1 = 50
n2 = 60
n3 = 100

p = 200

location = c(rep(1,20), rep(2, 50), rep(3, 100), rep(4, 30))

## expression data from null
set.seed(0416) # random seeds for example data
exp1 = matrix(rnorm(n1*p, mean = 0, sd = 1), nrow = n1)
exp2 = matrix(rnorm(n2*p, mean = 0, sd = 1), nrow = n2)
exp3 = matrix(rnorm(n3*p, mean = 0, sd = 1), nrow = n3)

exp_list = list(exp1, exp2, exp3)

result = snQTL_test_corrnet(exp_list = exp_list, method = 'tensor',
                          npermute = 30, seeds = 1:30, stats_seed = 0416,
                          trans = TRUE, location = location)

result$emp_p_value
```

`soft` *Soft threshold*

Description

Soft threshold

Usage

```
soft(x, d)
```

Arguments

x	a numeric vector
d	the soft threshold level

Value

the vector after soft thresholding `x` at level `d`.

See Also

`symmPMD()`.

`solvePMD` *Solving symmetric Penalized Matrix Decomposition*

Description

An iterative algorithm that solves the Sparse Principal Component Analysis problem: given a positive definite matrix A:

$$max_v v^T A v$$

subject to

$$||v||_2 \leq 1, ||v||_1 \leq s$$

The solution v is the sparse leading eigenvector, and the corresponding objective $v^T A v$ is the sparse leading eigenvalue.

Usage

```
solvePMD(x, sumabsv, v, niter = 50, trace = TRUE)
```

Arguments

`x`	p-by-p matrix, symmetric and positive definite
`sumabsv`	the upperbound of the L_1 norm of v, controlling the sparsity of solution. Must be between 1 and $sqrt(p)$.
`v`	the starting value of the algorithm.
`niter`	number of iterations to perform the iterative optimizations
`trace`	whether to print tracing info during optimization

Value

A list containing the following components:

`v`	the sparse leading eigenvector v
`d`	the sparse leading eigenvalue $d = v^T A v$
`v.init`	the initial value of v

See Also

`symmPMD()`.

SSTD_R1 *Sparse Symmetric Tensor Decomposition (SSTD)*

Description

SSTD solves the rank-1 approximation to the a p-by-p-by-q sparse symmetric tensor $\mathcal{D}$:

$$\min_{\Lambda,v,u} ||\mathcal{D} - \Lambda v \circ v \circ u||_F^2$$

subject to

$$\Lambda > 0, v \in R^p, u \in R^q, ||v||_2 = ||u||_2 = 1, ||v||_0 <= R$$

The solution Λ is the sparse leading tensor eigenvalue (sLTE), v is the sparse leading tensor eigenvector, and u is the loading vector.

The Symmetric Penalized Matrix Decomposition `symmPMD()` is used in the iterative algorithm.

Usage

```
SSTD_R1(
  T_obs,
  u_ini,
  v_ini,
  max_iter = 20,
  sumabs = 0.5,
  niter = 20,
  rho = 1000,
  tol = 10^(-3),
  verbose = FALSE
)
```

Arguments

`T_obs`	array, a p-by-p-by-q tensor; each p-by-p layer in `T_obs` should be symmetric
`u_ini`	vector, with length q; the random initialization for loading vector
`v_ini`	vector, with length p; the random initialization for tensor eigenvector
`max_iter`	integer, the maximal iteration number
`sumabs`	number, the number specify the sparsity level in the matrix/tensor eigenvector; `sumabs` takes value between $1/sqrt(p)$ and 1, where p is the dimension; `sumabs`$*sqrt(p)$ is the upperbound of the L1 norm of the leading matrix/tensor eigenvector (see `symmPMD()`)
`niter`	integer, the number of iterations to use in the PMD algorithm (see `symmPMD()`)
`rho`	number, a large positive constant adding to the diagonal elements to ensure positive definiteness in symmetric matrix spectral decomposition
`tol`	number, the tolerance threshold for SSTD convergence; if the error difference between two iterations is smaller than `tol`, then we stop the iteration and consider the algorithm converges
`verbose`	logic variable, whether to print the progress during permutation tests

Value

a list containing the following:

`u_hat`	vector, with length q; the estimated loading vector
`v_hat`	vector, with length p; the estimated tensor eigenvector
`gamma_hat`	number, the estimated sLTE Λ

References

Hu, J., Weber, J. N., Fuess, L. E., Steinel, N. C., Bolnick, D. I., & Wang, M. (2025). A spectral framework to map QTLs affecting joint differential networks of gene co-expression. PLOS Computational Biology, 21(4), e1012953.

Sun, W. W., Lu, J., Liu, H., & Cheng, G. (2017). "Provable sparse tensor decomposition." Journal of the Royal Statistical Society Series B: Statistical Methodology, 79(3), 899-916.

See Also

`symmPMD()`

`symmPMD` *Symmetric Penalized Matrix Decomposition.*

Description

This function solves for the Sparse Principal Component Analysis given a positive definite matrix A:

$$max_v v^T Av$$

subject to

$$||v||_2 \leq 1, ||v||_1 \leq s$$

The solution v is the sparse leading eigenvector, and the corresponding objective $v^T Av$ is the sparse leading engenvalue.

The algorithm uses an iterative procedure similar to the R Package "PMA", but speeds up the computation using the extra constraint that the decomposition is symmetric.

Usage

```
symmPMD(x, sumabs = 0.3, niter = 50, v = NULL, trace = TRUE)
```

Arguments

`x`	p-by-p matrix, symmetric and positive definite
`sumabs`	sumabs*$sqrt(p)$ is the upperbound of the L_1 norm of v, controling the sparsity of solution. Must be between $1/sqrt(p)$ and 1.
`niter`	number of iterations to perform the iterative optimizations
`v`	the starting value of the algorithm, either a pre-calculated first singular vector of x, or NULL.
`trace`	whether to print tracing info during optimization

Value

A list containing the following components:

`v`	the sparse leading eigenvector v
`d`	the sparse leading eigenvalue $d = v^T A v$
`sumabs`	sumabs$*sqrt(p)$ is the upperbound of the L_1 norm of v

References

Zhu, Lingxue, et al. "Testing high-dimensional covariance matrices, with application to detecting schizophrenia risk genes." The annals of applied statistics 11.3 (2017): 1810.

Witten, Tibshirani and Hastie (2009), "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis", Biostatistics 10(3):515-534.

`Tensor-class` *S4 Class for a Tensor*

Description

An S4 class for a tensor with arbitrary number of modes. The Tensor class extends the base "array" class to include additional tensor manipulation (folding, unfolding, reshaping, subsetting) as well as a formal class definition that enables more explicit tensor algebra.

Slots

num_modes number of modes (integer)

modes vector of modes (integer), aka sizes/extents/dimensions

data actual data of the tensor, which can be 'array' or 'vector'

Note

All of the decompositions and regression models in this package require a Tensor input.

Author(s)

James Li <`jamesyili@gmail.com`>

References

James Li, Jacob Bien, Martin T. Wells (2018). rTensor: An R Package for Multidimensional Array (Tensor) Unfolding, Multiplication, and Decomposition. Journal of Statistical Software, Vol. 87, No. 10, 1-31. URL: http://www.jstatsoft.org/v087/i10/.

See Also

`as.tensor`

`ttl` *Tensor Times List*

Description

Contracted (m-Mode) product between a Tensor of arbitrary number of modes and a list of matrices. The result is folded back into Tensor.

Usage

```
ttl(tnsr, list_mat, ms = NULL)
```

Arguments

`tnsr`	Tensor object with K modes
`list_mat`	a list of matrices
`ms`	a vector of modes to contract on (order should match the order of `list_mat`)

Details

Performs `ttm` repeated for a single Tensor and a list of matrices on multiple modes. For instance, suppose we want to do multiply a Tensor object `tnsr` with three matrices `mat1`, `mat2`, `mat3` on modes 1, 2, and 3. We could do `ttm(ttm(ttm(tnsr,mat1,1),mat2,2),3)`, or we could do `ttl(tnsr,list(mat1,mat2,mat3),c(1,2,3))`. The order of the matrices in the list should obviously match the order of the modes. This is a common operation for various Tensor decompositions such as CP and Tucker. For the math on the m-Mode Product, see Kolda and Bader (2009).

Value

Tensor object with K modes

Note

The returned Tensor does not drop any modes equal to 1.

References

T. Kolda, B. Bader, "Tensor decomposition and applications". SIAM Applied Mathematics and Applications 2009, Vol. 51, No. 3 (September 2009), pp. 455-500. URL: https://www.jstor.org/stable/25662308

See Also

`ttm`

Examples

```
tnsr <- new('Tensor',3L,c(3L,4L,5L),data=runif(60))
lizt <- list('mat1' = matrix(runif(30),ncol=3),
'mat2' = matrix(runif(40),ncol=4),
'mat3' = matrix(runif(50),ncol=5))
ttl(tnsr,lizt,ms=c(1,2,3))
```

---

ttm *Tensor Matrix Product (m-Mode Product)*

Description

Contracted (m-Mode) product between a Tensor of arbitrary number of modes and a matrix. The result is folded back into Tensor.

Usage

```
ttm(tnsr, mat, m = NULL)
```

Arguments

`tnsr`	Tensor object with K modes
`mat`	input matrix with same number columns as the `m`th mode of `tnsr`
`m`	the mode to contract on

Details

By definition, the number of columns in `mat` must match the `m`th mode of `tnsr`. For the math on the m-Mode Product, see Kolda and Bader (2009).

Value

a Tensor object with K modes

Note

The `m`th mode of `tnsr` must match the number of columns in `mat`. By default, the returned Tensor does not drop any modes equal to 1.

References

T. Kolda, B. Bader, "Tensor decomposition and applications". SIAM Applied Mathematics and Applications 2009, Vol. 51, No. 3 (September 2009), pp. 455-500. URL: https://www.jstor.org/stable/25662308

See Also

`ttl`

Examples

```
tnsr <- new('Tensor',3L,c(3L,4L,5L),data=runif(60))
mat <- matrix(runif(50),ncol=5)
ttm(tnsr,mat,m=3)
```

`unfold-methods` *Tensor Unfolding*

Description

Unfolds the tensor into a matrix, with the modes in `rs` onto the rows and modes in `cs` onto the columns. Note that `c(rs,cs)` must have the same elements (order doesn't matter) as `x@modes`. Within the rows and columns, the order of the unfolding is determined by the order of the modes. This convention is consistent with Kolda and Bader (2009).

Usage

```
unfold(tnsr, row_idx, col_idx)
```

Arguments

`tnsr`	the Tensor instance
`row_idx`	the indices of the modes to map onto the row space
`col_idx`	the indices of the modes to map onto the column space

Details

```
unfold(tnsr,row_idx=NULL,col_idx=NULL)
```

Value

matrix with `prod(row_idx)` rows and `prod(col_idx)` columns

References

T. Kolda, B. Bader, "Tensor decomposition and applications". SIAM Applied Mathematics and Applications 2009, Vol. 51, No. 3 (September 2009), pp. 455-500. URL: https://www.jstor.org/stable/25662308.

Examples

```
tnsr <- rand_tensor()
matT3<-unfold(tnsr,row_idx=2,col_idx=c(3,1))
```

Index