Package 'LOCUS'

July 21, 2025

Type Package

Title Low-Rank Decomposition of Brain Connectivity Matrices with Uniform Sparsity

Version 1.0

Date 2022-08-28

Maintainer Jialu Ran <jialuran422@gmail.com>

Depends R (>= 3.1.0), ica, MASS, far

Description To decompose symmetric matrices such as brain connectivity matrices so that one can extract sparse latent component matrices and also estimate mixing coefficients, a blind source separation (BSS) method named LOCUS was proposed in Wang and Guo (2023) <doi:10.48550/arXiv.2008.08915>. For brain connectivity matrices, the outputs correspond to sparse latent connectivity traits and individual-level trait loadings.

License GPL-2

NeedsCompilation no

Author Yikai Wang [aut, cph],
Jialu Ran [aut, cre],
Ying Guo [aut, ths]

Repository CRAN

Date/Publication 2022-10-04 07:20:05 UTC

Contents

LOCUS *LOCUS: Low-rank decomposition of brain connectivity matrices with uniform sparsity*

Description

This is the main function in the package. It conducts the LOCUS approach for decomposing brain connectivity data into subnetworks.

Usage

```
LOCUS(Y, q, V, MaxIteration=100, penalty="SCAD", phi = 0.9, approximation=TRUE,
preprocess=TRUE, espli1=0.001, espli2=0.001, rho=0.95, silent=FALSE)
```

Arguments

Y	Group-level connectivity data from N subjects, which is of dimension N x p, where p is number of edges. Each row of Y represents a subject's vectorized connectivity matrix by Ltrans function.
q	Number of ICs/subnetworks to extract.
V	Number of nodes in the network. Note: p should be equal to V(V-1)/2.
MaxIteration	Maximum number of iteractions.
penalty	The penalization approach for uniform sparsity, which can be NULL, SCAD, and L1.
phi	ϕ: tuning parameter for uniform sparse penalty.
approximation	Whether to use an approximated algorithm to speed up the algorithm.
preprocess	Whether to preprocess the data, which reduces the data dimension to q and whiten the data.
espli1	Toleration for convergence on mixing coefficient matrix, i.e. A.
espli2	Toleration for convergence on latent sources, i.e. S.
rho	ρ: tuning parameter for selecting number of ranks in each subnetwork's decomposition.
silent	Whether to print intermediate steps.

Details

This is the main function for LOCUS decomposition of brain connectivity matrices, which is to minimize the following objective function:

$$\sum_{i=1}^{N} \|y_i - \sum_{l=1}^{q} a_{il} s_l\|_2^2 + \phi \sum_{l=1}^{q} \|s_l\|_*,$$

where y_i is the transpose of ith row in Y, $s_l = L(X_l D_l X_l')$ represents the lth vectorized latent source/subnetwork with low-rank decomposition, L is Ltrans function, $\|\cdot\|_*$ represents the penalty which can either be NULL, L1, or SCAD (Fan & Li, 2001).

If user want to do BIC parameter selection of ϕ, ρ before calling LOCUS main function, one can use `LOCUS_BIC_selection` to find the best parameter set. Further details can be found in the LOCUS paper.

Value

An R list from Locus containing the following terms:

`Conver`	Whether the algorithm is converaged.
`A`	Mixing matrix $\{a_{il}\}$ of dimension N by q.
`S`	Subnetworks of dimension q by p, where each row represents a vectorized subnetwork based on `Ltrans` function.
`theta`	A list of length q, where `theta[[i]]` contains the symmetric low-rank decomposition of `i`th subnetwork.

References

Wang, Y. and Guo, Y. (2023). *LOCUS: A novel signal decomposition method for brain network connectivity matrices using low-rank structure with uniform sparsity.* Annals of Applied Statistics.

Fan, J., & Li, R. (2001). *Variable selection via nonconcave penalized likelihood and its oracle properties.* Journal of the American statistical Association, 96(456), 1348-1360.

Examples

```
## Simulated the data to use.
V = 50
S1 = S2 = S3 = matrix(0,ncol = V,nrow = V)
S1[5:20,5:20] = 4;S1[23:37,23:37] = 3;S1[40:48,40:48] = 3
S2[15:20,] = -3;S2[,15:20] = -3
S3[15:25,36:45] = 3; S3[36:45,15:25] = 3
Struth = rbind(Ltrans(S1,FALSE) , Ltrans(S2,FALSE), Ltrans(S3,FALSE))
set.seed(100)
Atruth = matrix(rnorm(100*3),nrow=100,ncol=3)
Residual = matrix(rnorm(100*dim(Struth)[2]),nrow=100)
Yraw = Atruth%*%Struth + Residual

##### Run Locus on the data #####
Locus_result = LOCUS(Yraw,3,V)

oldpar = par(mfrow=c(2,3))
for(i in 1:dim(Struth)[1]){image(Ltrinv(Struth[i,],V,FALSE))}
for(i in 1:dim(Locus_result$S)[1]){image(Ltrinv(Locus_result$S[i,],V,FALSE))}
par(oldpar)
```

`LOCUS_BIC_selection` *BIC-based Hyper-parameters selection for LOCUS*

Description

This function is to conduct the BIC-based hyper-parameters selection for LOCUS.

Usage

```
LOCUS_BIC_selection(Y, q, V, MaxIteration=50, penalty="SCAD",
phi_grid_search=seq(0.2, 1, 0.2), rho_grid_search=c(0.95),
espli1=0.001, espli2=0.001, save_LOCUS_output=TRUE,
preprocess=TRUE)
```

Arguments

`Y`	Group-level connectivity data from N subjects, which is of dimension N x p, where p is number of edges. Each row of Y represents a subject's vectorized connectivity matrix by `Ltrans` function.
`q`	Number of ICs/subnetworks to extract.
`V`	Number of nodes in the network. Note: p should be equal to V(V-1)/2.
`MaxIteration`	Maximum number of iteractions.
`penalty`	The penalization approach for uniform sparsity, which can be `NULL`, `SCAD`, `L1`, `Hardthreshold`.
`phi_grid_search`	Grid search candidates for tuning parameter of uniform sparse penalty.
`espli1`	Toleration for convergence on mixing coefficient matrix, i.e. A.
`espli2`	Toleration for convergence on latent sources, i.e. S.
`rho_grid_search`	Grid search candidates for tuning parameter for selecting number of ranks in each subnetwork's decomposition.
`save_LOCUS_output`	Whether to save LOCUS output from each grid search.
`preprocess`	Whether to preprocess the data, which reduces the data dimension to `q` and whiten the data.

Details

In Wang, Y. and Guo, Y. (2023), the tuning parameters for learning the LOCUS model include ϕ, ρ. The BIC-type criterion is proposed to select those parameters.

$$BIC = -2\sum_{i=1}^{N} log\{g(y_i; \sum_{l=1}^{q} \hat{a}_{il}\hat{s}_l, \hat{\sigma}^2 I_p)\} + log(N)\sum_{l=1}^{q} \|\hat{s}_l\|_0$$

where g denotes the pdf of a multivariate Gaussian distribution, $\hat{\sigma}^2 = \frac{1}{Np}\sum_i \|y_i - \sum_{l=1}^{q} \hat{a}_{il}\hat{s}_l\|_2^2$, $\|\cdot\|_0$ denotes the L_0 norm . This criterion balances between model fitting and model sparsity.

Value

bic_tab	BIC values per phi and rho.
LOCUS_results	LOCUS output, if save_LOCUS_output is TRUE.

Examples

```
## Simulated the data to use.
V = 50
S1 = S2 = S3 = matrix(0,ncol = V,nrow = V)
S1[5:20,5:20] = 4;S1[23:37,23:37] = 3;S1[40:48,40:48] = 3
S2[15:20,] = -3;S2[,15:20] = -3
S3[15:25,36:45] = 3; S3[36:45,15:25] = 3
Struth = rbind(Ltrans(S1,FALSE) , Ltrans(S2,FALSE), Ltrans(S3,FALSE))
set.seed(100)
Atruth = matrix(rnorm(100*3),nrow=100,ncol=3)
Residual = matrix(rnorm(100*dim(Struth)[2]),nrow=100)
Yraw = Atruth%*%Struth + Residual

##### Run Locus on the data #####
Locus_bic_result = LOCUS_BIC_selection(Yraw,3,V)
print(Locus_bic_result$bic_tab)
# line plot
plot(Locus_bic_result$bic_tab[,2], Locus_bic_result$bic_tab[,3], type = "b",
     xlab = "phi", ylab = "BIC")

# visualize the best result based on BIC
idx = which.min(Locus_bic_result$bic_tab[,3])
oldpar = par(mfrow=c(2,3))
for(i in 1:3){image(Ltrinv(Struth[i,], V, FALSE))}
for(i in 1:3){image(Ltrinv(Locus_bic_result$LOCUS_results[[idx]]$LOCUS$S[i,],
V, FALSE))}
par(oldpar)
```

Ltrans *Map a symmetric matrix into its upper triangle part.*

Description

This function is to map the upper triganle part of a symmetric matrix into a vector.

Usage

```
Ltrans(X, d = TRUE)
```

Arguments

X	A symmetric matrix of dimentional V by V.
d	Whether to include the diagonal part of X.

Value

A vector containing the upper triganle part of X.

`Ltrinv` *Inverse function of* `Ltrans` *to map back a vector into a symmetric matrix.*

Description

This function is the inverse function of `Ltrans`, which is to map a vector back to a symmetric matrix.

Usage

```
Ltrinv(x, V, d = TRUE)
```

Arguments

`x`	A vector to convert to a matrix, which is of length p.
`V`	Dimension of the matrix which `x` is converted to.
`d`	Whether diagonal is kept in `x` or not.

Value

A symmetric matrix whose upper triangle part is `x`.

Index