Package 'multiness'

May 9, 2026

Title MULTIplex NEtworks with Shared Structure

Version 1.0.2

Description Model fitting and simulation for Gaussian and logistic inner product MultiNeSS models for multiplex networks. The package implements a convex fitting algorithm with fully adaptive parameter tuning, including options for edge cross-validation. For more details see MacDonald et al. (2020).

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Imports glmnet (>= 4.0.2), Matrix (>= 1.1.0), RSpectra (>= 0.16.0),

Depends R (>= 3.6.0)

URL https://github.com/peterwmacd/multiness/

BugReports https://github.com/peterwmacd/multiness/issues/

NeedsCompilation no

Author Peter W. MacDonald [aut, cre] (ORCID: <https://orcid.org/0000-0001-7024-7242>)

Maintainer Peter W. MacDonald <pwmacdon@umich.edu>

Repository CRAN

Date/Publication 2022-11-25 01:20:05 UTC

Contents

agri_trade *Agricultural trade multiplex network*

Description

An undirected multiplex network containing trade volumes for 13 highly traded agricultural products for the year 2010, collected by the Food and Agriculture Organization of the United Nations (FAO). The original data set can be downloaded from Manlio DeDomenico's website. Array entries are in units of tonnes (metric tons) of bilateral trade of a given agricultural product. For further documentation and product definitions see https://www.fao.org/faostat/en/#definitions/.

Usage

```
data(agri_trade)
```

Format

An array of dimension $145 \times 145 \times 13$.

Source

https://manliodedomenico.com/; https://www.fao.org/faostat/en/#data/

References

DeDomenico et al. (2015) Nature Communications

ase *Adjacency Spectral Embedding (ASE)*

Description

ase calculates the d-dimensional adjacency spectral embedding of a symmetric $n \times n$ matrix M.

Usage

```
ase(M,d)
```

Arguments

M	A symmetric matrix.
d	A non-negative integer embedding dimension.

Value

An $n \times d$ matrix X, defined as $U|S|^{1/2}$ where S is a diagonal matrix of the d leading (in absolute value) eigenvalues of M, and U is a matrix of the corresponding eigenvectors.

X has an additional attribute `"signs"` which gives the sign of the eigenvalue corresponding to each column.

If $d = 0$, `ase` returns an $n \times 1$ matrix of zeros.

`expit` *Inverse logistic link function*

Description

`expit` applies the inverse logistic link function $f(x) = e^x/(1 + e^x)$.

Usage

```
expit(x)
```

Arguments

`x`	A numeric vector.

`logit` *Logistic link function*

Description

`logit` applies the logistic link function $f(x) = log(x/(1 - x))$.

Usage

```
logit(x,tol=1e-6)
```

Arguments

`x`	A numeric vector with values in the interval [0,1].
`tol`	A positive scalar which bounds the entries of x away from 0 and 1 for numerical stability. Defaults to `tol=1e-6`

multiness_fit *Fit the MultiNeSS model*

Description

multiness_fit fits the Gaussian or logistic MultiNeSS model with various options for parameter tuning.

Usage

```
multiness_fit(A,model,self_loops,refit,tuning,tuning_opts,optim_opts)
```

Arguments

A	An $n \times n \times m$ array containing edge entries for an undirected multiplex network on n nodes and m layers.
model	A string which provides choice of model, either 'gaussian' or 'logistic'. Defaults to 'gaussian'.
self_loops	A Boolean, if FALSE, all diagonal entries are ignored in optimization. Defaults to TRUE.
refit	A Boolean, if TRUE, a refitting step is performed to debias the eigenvalues of the estimates. Defaults to TRUE.
tuning	A string which provides the tuning method, valid options are 'fixed', 'adaptive', or 'cv'. Defaults to 'adaptive'.
tuning_opts	A list, containing additional optional arguments controlling parameter tuning. The arguments used depends on the choice of tuning method. If tuning='fixed', multiness_fit will utilize the following arguments:

lambda A positive scalar, the λ parameter in the nuclear norm penalty, see Details. Defaults to 2.309 * sqrt(n*m).

alpha A positive scalar or numeric vector of length m, the parameters α_k in the nuclear norm penalty, see Details. If a scalar is provided all α_k parameters are set to that value. Defaults to 1/sqrt(m)

If tuning='adaptive', multiness_fit will utilize the following arguments:

layer_wise A Boolean, if TRUE, the entry-wise variance is estimated individually for each layer. Otherwise the estimates are pooled. Defaults to TRUE.

penalty_const A positive scalar C which scales the penalty parameters (see Details). Defaults to 2.309.

penalty_const_lambda A positive scalar c which scales only the λ penalty parameter (see Details). Defaults to 1.

If tuning='cv', multiness_fit will utilize the following arguments:

layer_wise A Boolean, if TRUE, the entry-wise variance is estimated individually for each layer. Otherwise the estimates are pooled. Defaults to TRUE.

N_cv A positive integer, the number of repetitions of edge cross-validation performed for each parameter setting. Defaults to 3.

p_cv A positive scalar in the interval (0,1), the proportion of edge entries held out in edge cross-validation. Defaults to 0.1.

penalty_const_lambda A positive scalar c which scales only the λ penalty parameter (see Details). Defaults to `1`.

penalty_const_vec A numeric vector with positive entries, the candidate values of constant C to scale the penalty parameters (see Details). An optimal constant is chosen by edge cross-validation. Defaults to `c(1,1.5,...,3.5,4)`.

refit_cv A Boolean, if `TRUE`, a refitting step is performed when fitting the model for edge cross-validation. Defaults to `TRUE`

verbose_cv A Boolean, if `TRUE`, console output will provide updates on the progress of edge cross-validation. Defaults to `FALSE`.

`optim_opts` A list, containing additional optional arguments controlling the proximal gradient descent algorithm.

check_obj A Boolean, if `TRUE`, convergence is determined by checking the decrease in the objective. Otherwise it is determined by checking the average entry-wise difference in consecutive values of F. Defaults to `TRUE`.

eig_maxitr A positive integer, maximum iterations for internal eigenvalue solver. Defaults to `1000`.

eig_prec A positive scalar, estimated eigenvalues below this threshold are set to zero. Defaults to `1e-2`.

eps A positive scalar, convergence threshold for proximal gradient descent. Defaults to `1e-6`.

eta A positive scalar, step size for proximal gradient descent. Defaults to `1` for the Gaussian model, `5` for the logistic model.

init A string, initialization method. Valid options are `'fix'` (using initializers `optim_opts$V_init` and `optim_opts$U_init`), `'zero'` (initialize all parameters at zero), or `'svd'` (initialize with a truncated SVD with rank `optim_opts$init_rank`). Defaults to `'zero'`.

K_max A positive integer, maximum iterations for proximal gradient descent. Defaults to `100`.

max_rank A positive integer, maximum rank for internal eigenvalue solver. Defaults to `sqrt(n)`.

missing_pattern An $n \times n \times m$ Boolean array with `TRUE` for each observed entry and `FALSE` for missing entries. If unspecified, it is set to `!is.na(A)`.

positive A Boolean, if `TRUE`, singular value thresholding only retains positive eigenvalues. Defaults to `FALSE`.

return_posns A Boolean, if `TRUE`, returns estimates of the latent positions based on ASE. Defaults to `FALSE`.

verbose A Boolean, if `TRUE`, console output will provide updates on the progress of proximal gradient descent. Defaults to `FALSE`.

Details

A MultiNeSS model is fit to an $n \times n \times m$ array A of symmetric adjacency matrices on a common set of nodes. Fitting proceeds by convex proximal gradient descent on the entries of $F = VV^T$ and $G_k = U_k U_k^T$, see MacDonald et al., (2020), Section 3.2. Additional optional arguments for

the gradient descent routine can be provided in `optim_opts`. `refit` provides an option to perform an additional refitting step to debias the eigenvalues of the estimates, see MacDonald et al., (2020), Section 3.3.

By default, `multiness_fit` will return estimates of the matrices F and G_k. `optim_opts$return_posns` provides an option to instead return estimates of latent positions V and U_k based on the adjacency spectral embedding (if such a factorization exists).

Tuning parameters λ and α_k in the nuclear norm penalty

$$\lambda||F||_* + \sum_k \lambda\alpha_k||G_k||_*$$

are either set by the user (`tuning='fixed'`), selected adaptively using a robust estimator of the entry-wise variance (`tuning='adaptive'`), or selected using edge cross-validation (`tuning='cv'`). For more details see MacDonald et al., (2020), Section 3.4. Additional optional arguments for parameter tuning can be provided in `tuning_opts`.

Value

A list is returned with the MultiNeSS model estimates, dimensions of the common and individual latent spaces, and some additional optimization output:

`F_hat`	An $n \times n$ matrix estimating the common part of the expected adjacency matrix, $F = VV^T$. If `optim_opts$return_posns` is `TRUE`, this is not returned.
`G_hat`	A list of length m, the collection of $n \times n$ matrices estimating the individual part of each adjacency matrix, $G_k = U_k U_k^T$. If `optim_opts$return_posns` is `TRUE`, this is not returned.
`V_hat`	A matrix estimating the common latent positions. Returned if `optim_opts$return_posns` is `TRUE`.
`U_hat`	A list of length m, the collection of matrices estimating the individual latent positions. Returned if `optim_opts$return_posns` is `TRUE`.
`d1`	A non-negative integer, the estimated common dimension of the latent space.
`d2`	An integer vector of length m, the estimated individual dimension of the latent space for each layer.
`K`	A positive integer, the number of iterations run in proximal gradient descent.
`convergence`	An integer convergence code, `0` if proximal gradient descent converged in fewer than `optim_opts$K_max` iterations, `1` otherwise.
`lambda`	A positive scalar, the tuned λ penalty parameter (see Details).
`alpha`	A numeric vector of length m, the tuned α penalty parameters (see Details).

Examples

```
# gaussian model data
data1 <- multiness_sim(n=100,m=4,d1=2,d2=2,
                       model="gaussian")

# multiness_fit with fixed tuning
fit1 <- multiness_fit(A=data1$A,
```

```
                        model="gaussian",
                        self_loops=TRUE,
                        refit=FALSE,
                        tuning="fixed",
                        tuning_opts=list(lambda=40,alpha=1/2),
                        optim_opts=list(max_rank=20,verbose=TRUE))

# multiness_fit with adaptive tuning
fit2 <- multiness_fit(A=data1$A,
                        refit=TRUE,
                        tuning="adaptive",
                        tuning_opts=list(layer_wise=FALSE),
                        optim_opts=list(return_posns=TRUE))

# logistic model data
data2 <- multiness_sim(n=100,m=4,d1=2,d2=2,
                         model="logistic",
                         self_loops=FALSE)

# multiness_fit with cv tuning
fit3 <- multiness_fit(A=data2$A,
                        model="logistic",
                        self_loops=FALSE,
                        tuning="cv",
                        tuning_opts=list(N_cv=2,
                                         penalty_const_vec=c(1,2,2.309,3),
                                         verbose_cv=TRUE))
```

`multiness_sim` *Simulate from the MultiNeSS model*

Description

`multiness_sim` simulates a realization of the Gaussian or logistic MultiNeSS model with Gaussian latent positions.

Usage

```
multiness_sim(n,m,d1,d2,model,sigma,self_loops,opts)
```

Arguments

`n`	A positive integer, the number of nodes.
`m`	A positive integer, the number of layers.
`d1`	A non-negative integer, the number of common latent dimensions.
`d2`	A non-negative integer, the number of individual latent dimensions.
`model`	A string which provides choice of model, either `'gaussian'` or `'logistic'`. Defaults to `'gaussian'`.

`sigma` A positive scalar or numeric vector of length `m`, the entry-wise standard deviation for the Gaussian noise for all layers (if a scalar) or for each layer (if a vector). Ignored under the logistic model. Defaults to `1`.

`self_loops` A Boolean, if `FALSE`, all diagonal entries are set to zero. Defaults to `TRUE`.

`opts` A list, containing additional optional arguments:

density_shift A positive scalar, for the logistic model only, a shift subtracted from the log-odds of each edge to control overall edge density. Defaults to `0`.

dependence_type A string, valid choices are `'all'` or `'U_only'` for the Gaussian model; `'all'` for the logistic model. If `'all'`, V and U_k; and U_k and U_l (for $k \neq l$) have expected canonical correlation approximately equal to |rho| (see `rho`). If `'U_only'`, U_k and U_l (for $k \neq l$) have expected canonical correlation approximately equal to |rho| (see `rho`). Defaults to `'all'`.

gamma A positive scalar, the standard deviation of the entries of the latent position matrices V and U_k. Defaults to `1`.

return_density A Boolean, if `TRUE` and `model='logistic'`, the function will return an array containing the overall edge density. Defaults to `FALSE`.

return_P A Boolean, if `TRUE`, the function will return an array containing the expected adjacency matrices. Defaults to `FALSE`.

rho A positive scalar in the interval (-1,1), controls the expected canonical correlation between latent position matrices (see `dependence_type`). Defaults to `0`.

Details

The common and individual latent positions, V and U_k respectively, are generated as Gaussian random variables with standard deviation `opts$gamma`, and dependence controlled by the optional arguments `opts$dependence_type` and `opts$rho`.

Under the Gaussian model, the $n \times n$ adjacency matrix for layer $k = 1, ..., m$ has independent Gaussian entries with standard deviation `sigma` and mean given by

$$E(A_k) = VV^T + U_k U_k^T.$$

Under the logistic model, the $n \times n$ adjacency matrix for layer $k = 1, ..., m$ has independent Bernoulli entries with mean given by

$$E(A_k) = g(VV^T + U_k U_k^T),$$

where g denotes the element-wise application of the inverse logistic link (`expit`) function. Under both models, `self_loops` provides an option to set the diagonal entries of the adjacency matrices to zero.

Value

A list is returned with the realizations of the latent dimensions and the multiplex network:

`A` An array of dimension $n \times n \times m$, the realized multiplex network.

V	A matrix of dimension $n \times d1$, the realized common latent positions. If d1=0, returns NULL.
U	An array of dimension $n \times d2 \times m$, the realized individual latent positions. If d2=0, returns NULL.
P	If specified, an array of dimension $n \times n \times m$, the expected multiplex network.
density	If specified and model='logistic', the overall edge density.

Examples

```
# gaussian model, uncorrelated latent positions
data1 <- multiness_sim(n=100,m=4,d1=2,d2=2,
                       model="gaussian")

# logistic model, correlated latent positions
data2 <- multiness_sim(n=100,m=4,d1=2,d2=2,
                        model="logistic",
                        self_loops=FALSE,
                        opts=list(dependence_type="all",rho=.3,return_density=TRUE))
```

Index