Package ‘pdglasso’

March 27, 2023

Type Package

Title What the Package Does (Title Case)

Version 0.1.0

Description More about what it does (maybe more than one line)
Use four spaces when indenting paragraphs within the Description.

License What license is it under?

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Url https://github.com/savranciati/pdglasso

Depends R (>= 2.10)

R topics documented:

`admm.pdglasso` *Estimate a concentration matrix under the pdColG model using (adaptive) ADMM graphical lasso algorithm.*

Description

By providing a covariance matrix S and values for lambda_1 and lambda_2, this function estimates a concentration matrix X under the coloured graphical model for paired data, using the (adaptive) ADMM algorithm. The output is the matrix and a list of internal parameters used by the function, together with the specific call in terms of symmetries and penalties required by the user.

Usage

```
admm.pdglasso(
  S,
  lambda1 = 1,
  lambda2 = 1e-04,
  type = c("vertex", "inside.block.edge", "across.block.edge"),
  force.symm = NULL,
  X.init = NULL,
  rho1 = 1,
  rho2 = 1,
  varying.rho1 = TRUE,
  varying.rho2 = TRUE,
  max_iter = 1000,
  eps.abs = 1e-12,
  eps.rel = 1e-12,
  verbose = FALSE,
  print.type = TRUE
)
```

Arguments

`S`	A $p \times p$ covariance (or correlation) matrix.
`lambda1`	A non-negative scalar (or vector) penalty that encourages sparsity in the concentration matrix. If a vector is provided, it should match the appropriate length, i.e.
`lambda2`	A non-negative scalar (or vector) penalty that encourages equality constraints in the concentration matrix. If a vector is provided, it should match the appropriate length, i.e.
`type`	A string or vector of strings for the type of equality constraints to be imposed; zero, one or more available options can be selected among: * "vertex", symmetries are imposed on the diagonal entries of the concentration matrix. * "inside.block.edge", symmetries are imposed between elements of the LL and RR block the concentration matrix. * "across.block.edge", symmetries are imposed between elements of the LR and RL block the concentration matrix. Shortened forms are accepted too, i.e. "V" or "vert" for "vertex".
`force.symm`	A string or vector of strings to impose forced symmetry on the corresponding block of the concentration matrix. Same options as "type".
`X.init`	(optional) A $p \times p$ initial guess for the concentration matrix and/or starting solution for the ADMM algorithm.
`rho1`	A scalar; tuning parameter of the ADMM algorithm to be used for the outer loop. It must be strictly positive.
`rho2`	A scalar; tuning parameter of the ADMM algorithm to be used for the inner loop. It must be strictly positive.
`varying.rho1`	A boolean value; if `TRUE` the parameter rho1 is updated iteratively to speed-up convergence.
`varying.rho2`	A boolean value; if `TRUE` the parameter rho2 is updated iteratively to speed-up convergence.
`max_iter`	An integer; maximum number of iterations to be run in case the algorithm does not converge.

`eps.abs`	A scalar; the absolute precision required for the computation of primal and dual residuals of the ADMM algorithm.
`eps.rel`	A scalar; the relative precision required for the computation of primal and dual residuals of the ADMM algorithm.
`verbose`	A boolean value; if `TRUE` the progress (and internal convergence of inner loop) is shown in the console while the algorithm is running.
`print.type`	A boolean value; if `TRUE` the acronym used for the model - which penalties - is returned as printed output in the console.

Value

A list, whose element are:

- `X`, the estimated concentration matrix under the pdglasso model; the model is identified by the values of lambda1 and lambda 2, together with the type of penalization imposed.
- `acronyms`, a vector of strings for the type of penalties and forced symmetries imposed when calling the function.
- `internal.par`, a list of internal parameters passed to the function at the call, as well as convergence information.

Examples

```
S <- cov(toy_data$sample.data)
admm.pdglasso(S)
```

`fit.pdColG` *Fit and select a coloured graphical models for paired data according to eBIC criterion.*

Description

Performs a sequence of calls to `admm.pdglasso()` providing two grids of values for lambda_1 and lambda_2. First, a grid search conditional on lambda_2=0 is run to select the best lambda_1 value among the candidates (according to eBIC); conditional on the best lambda_1, a similar search is performed for lambda_2. The output is the select model, given by the estimated concenration matrix and corresponding graph.

Usage

```
fit.pdColG(
  S,
  n,
  n.l1 = 15,
  n.l2 = 15,
  gamma.eBIC = 0.5,
  type = c("vertex", "inside.block.edge", "across.block.edge"),
  force.symm = NULL,
  X.init = NULL,
  rho1 = 1,
  rho2 = 1,
  varying.rho1 = TRUE,
  varying.rho2 = TRUE,
  max_iter = 1000,
  eps.abs = 1e-12,
  eps.rel = 1e-12,
  verbose = FALSE,
  print.type = TRUE,
  ...
)
```

Arguments

`S`	A $p \times p$ covariance (or correlation) matrix.
`n`	the sample size of the data used to compute the sample covariance matrix S.
`n.l1`	the number of values in the grid of candidates for lambda_1.
`n.l2`	the number of values in the grid of candidates for lambda_2.
`type`	A string or vector of strings for the type of equality constraints to be imposed; zero, one or more available options can be selected among: * "vertex", symmetries are imposed on the diagonal entries of the concentration matrix. * "inside.block.edge", symmetries are imposed between elements of the LL and RR block the concentration matrix. * "across.block.edge", symmetries are imposed between elements of the LR and RL block the concentration matrix. Shortened forms are accepted too, i.e. "V" or "vert" for "vertex".
`force.symm`	A string or vector of strings to impose forced symmetry on the corresponding block of the concentration matrix. Same options as "type".
`X.init`	(optional) A $p \times p$ initial guess for the concentration matrix and/or starting solution for the ADMM algorithm.
`rho1`	A scalar; tuning parameter of the ADMM algorithm to be used for the outer loop. It must be strictly positive.
`rho2`	A scalar; tuning parameter of the ADMM algorithm to be used for the inner loop. It must be strictly positive.
`varying.rho1`	A boolean value; if `TRUE` the parameter rho1 is updated iteratively to speed-up convergence.
`varying.rho2`	A boolean value; if `TRUE` the parameter rho2 is updated iteratively to speed-up convergence.
`max_iter`	An integer; maximum number of iterations to be run in case the algorithm does not converge.
`eps.abs`	A scalar; the absolute precision required for the computation of primal and dual residuals of the ADMM algorithm.
`eps.rel`	A scalar; the relative precision required for the computation of primal and dual residuals of the ADMM algorithm.
`verbose`	A boolean value; if `TRUE` the progress (and internal convergence of inner loop) is shown in the console while the algorithm is running.
`print.type`	A boolean value; if `TRUE` the acronym used for the model - which penalties - is returned as printed output in the console.
`gamma`	the parameter for the eBIC computation. gamma=0 is equivalent to BIC.

Value

a list:

- model, the final model;
- pdColG, the associated coloured graph;
- l1.path, a matrix containing the grid values for lambda_1 as well as quantities used in eBIC computation;
- l2.path, a matrix containing the grid values for lambda_2 as well as quantities used in eBIC computation.

Examples

```
S <- cov(toy_data$sample.data)
fit.pdColG(S)
```

`get.pdColG` *Build a graph from the output of a call to* `admm.pdglasso`.

Description

Description here.

Usage

```
get.pdColG(admm.out, th1 = NULL, th2 = NULL, verbose = FALSE)
```

Arguments

`admm.out`	An object of list type, that is the output of a call to the `admm-pdglasso` function.
`th1`	(optional) A scalar, the threshold to identify edges in the graph; it must be non-negative.
`th2`	(optional) A scalar, the threshold to identify coloured edges in the graph; it must be non-negative.
`verbose`	(optional) if `TRUE` provides summary statistics of the graph.

Value

a list, containing:

- g, the graph in matrix form.
- dof, the degrees of freedom corresponding to the graph build under the pdglasso model provided.

Examples

```
S <- cov(toy_data)
mod.out <- admm.pdglasso(S)
get.pdColG(mod.out)
```

pdColG.mle — *Maximum likelihood estimate*

Description

Computes the m.l.e. of the concentration matrix of a colured graphical model for paired data.

Usage

```
pdColG.mle(S, pdColG)
```

Arguments

S	a sample variance and covariance matrix.
pdColG	a coloured graph for paired data.

Value

the m.l.e. of the concentration matrix Σ^{-1}.

Examples

```
#
```

pdColG.summarize — *Summary statistics for coloured graphs for paired data*

Description

This function

Usage

```
pdColG.summarize(pdColG, print.summary = TRUE)
```

Arguments

pdColG	a coloured graph for paired data.
print.summary	logical (default TRUE) indicating whether a summary should be printed.

Value

a list

Examples

```
#
pdColG.summarize(toy_data$pdColG)
```

simul.pdColG *Title*

Description

Title

Usage

```
simul.pdColG(
  p,
  concent.mat = TRUE,
  sample = TRUE,
  Sigma = NULL,
  sample.size = NULL,
  type = c("vertex", "inside.block.edge", "across.block.edge"),
  force.symm = NULL,
  dens = 0.1,
  dens.vertex = NULL,
  dens.inside = NULL,
  dens.across = NULL
)
```

Arguments

`p`	number of variables.
`concent.mat`	a logical (default `TRUE`) indicating whether a concentration matrix should be generated.
`sample`	a logical (default `TRUE`) indicating whether a sample form a normal distribution with zero mean vector and the generated concentration matrix should be generated.
`Sigma`	x
`sample.size`	sample size with default value equal to $3p$.
`type`	x
`force.symm`	x
`dens`	x
`dens.vertex`	x
`dens.inside`	x
`dens.across`	x

Value

this function

Examples

```
#
```

toy_data *Toy dataset generated through* `simul.pdColG()` *function*

Description

Data simulated by using the function simul.pdColG with the following arguments:

- `p <- 20`
- `q <- p/2`
- `dens=0.3, type=c("v","i","a"), force=NULL`

Usage

```
toy_data
```

Format

`toy_data`**:**

A list containing three elements:

- pdColG, a $p \times p$ matrix describing the coloured graphical model,
- K, the concentration matrix associated to the model,
- sample.data, a data frame with 60 rows and 20 columns.

Source

Generated by the package functions.

Index