Package 'ragt2ridges'

January 28, 2020

Type Package

Title Ridge Estimation of Vector Auto-Regressive (VAR) Processes

Version 0.3.4

Date 2020-01-28

Author Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

Maintainer Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

Description The ragt2ridges-package provides ridge maximum likelihood estimation of vector autoregressive processes: the VAR(1), VAR(2) and VARX(1) model (more to be added). Prior knowledge may be incorporated in the estimation through a) specification of the edges believed to be absent in the time series chain graph, and b) a shrinkage target towards which the parameter estimate is shrunken for large penalty parameter values. Estimation functionality is accompanied by methodology for penalty parameter selection. In addition, the package offers supporting functionality for the exploitation of estimated models. Among others, i) a procedure to infer the support of the non-sparse ridge estimate (and thereby of the time series chain graph) is implemented, ii) a table of node-wise network summary statistics, iii) mutual information analysis, and iv) impulse response analysis. Cf. Miok et al. (2017) <DOI:10.1002/bimj.201500269> and Miok et al. (2019) <DOI:10.1002/bimj.201700195> for details on the implemented methods.

License GPL (>= 2)

LazyLoad yes

URL https://github.com/wvanwie/ragt2ridges

Depends R (>= 2.15.1), igraph, rags2ridges,

Imports abind, expm, fdrtool, graphics, grDevices, MASS, Matrix, methods, mvtnorm, stats, Biobase, CGHbase

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-01-28 15:30:02 UTC

R topics documented:

`ragt2ridges-package` *Ridge Estimation of Vector Auto-Regressive (VAR) Processes*

Description

Ridge maximum likelihood estimation of vector auto-regressive processes and supporting functions for their exploitation. Currently, it includes:

- Ridge estimation of the parameters of Vector Auto-Regressive models, commonly referred to as VAR models, through the functions `ridgeVAR1`, `ridgeVAR2`, `ridgeVARX1` and `ridgeVAR1fused`. These functions are complemented by `optPenaltyVAR1`, `optPenaltyVAR2`, `optPenaltyVARX1` and `optPenaltyVAR1fused`, functions for penalty parameters selection through (leave-one-out) cross-validation (with supporting functions `loglikVAR1`, `loglikLOOCVVAR1`, `loglikLOOCVVAR2`, `loglikLOOCVVARX1` and `loglikLOOCVVAR1fused`).
- Functions for simulating VAR-type data (`createA`, `dataVAR1`, `dataVAR2` and `dataVARX1`), data visualization (`plotVAR1data`), and some simple data manipulations (`centerVAR1data`, `array2longitudinal`, and `longitudinal2array`).
- Some diagnostics provided through `evaluateVAR1fit`, `loglikLOOCVcontourVAR1`, and `ridgePathVAR1`.
- Several post-estimation analyses to exploit the fitted model. Among others: support determination of the various VAR model parameters (`sparsifyVAR1`, `sparsifyVAR2`, `sparsifyVARX1`), visualization of the (aspects of the) time-series chain graph (`graphVAR1`, `graphVAR2`, `graphVARX1`, `CIGofVAR1` and `CIGofVAR2`), and summary statistics per variate in terms of the VAR(1) model and its associated time-series chain graph (`nodeStatsVAR1`, `motifStatsVAR1`, `impulseResponseVAR1`, and `mutualInfoVAR1`). The latter are also available for the VAR(2) and VARX(1) models: `impulseResponseVAR2`, `impulseResponseVARX1` and `mutualInfoVAR2`).
- Time-series omics data (`hpvP53`).

Future versions aim to include more functionality for time-series models.

The `ragt2ridges`-package is a sister-package to the `rags2ridges`-package, augmenting the latter 'base' package with functionality for time-course studies. Being its sibling `ragt2ridges` mimicks `rags2ridges` in the function names (compare e.g. `ridgeP` to `ridgeVAR1`).

Details

Package:	ragt2ridges
Type:	Package
Version:	0.3.3
Date:	2019-12-05
License:	GPL (>= 2)

Note

The (R)cpp-code of ragt2ridges includes parts of the Rcpp-module of rags2ridges, for it is currently impossible to import this directly.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.ml>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2017), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

The rags2ridges-package.

---

array2longitudinal *Convert a time-series array to a longitudinal-object.*

Description

Converts a 3-dim array (containing time-series data of multiple individuals) to an object of that can directly be converted to the longitudinal class through the as.longitudinal function.

Usage

```
array2longitudinal(Y, keepMissings=TRUE)
```

Arguments

Y	Three-dimensional array containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively.
keepMissings	The array-format assumes a balanced layout of the time-course experiment. The experiment may have failed for some design points and no data is available. In the longitudinal-format these design points may be left out. This logical indicates whether they should be kept in (or not).

Value

A longitudinal-object containing time-series data.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

See Also

dataVAR1, longitudinal, longitudinal2array, as.longitudinal.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- createA(p, "chain")

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

# convert to longitudinal
Ylongitudinal <- array2longitudinal(Y)
```

---

centerVAR1data *Zero-centering of time-course data*

Description

Per individual, covariate-wise zero centering of the time-series data.

Usage

```
centerVAR1data(Y)
```

Arguments

Y	Three-dimensional array containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively.

Value

An array with dimensions as the input.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

See Also

dataVAR1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- createA(p, "chain")

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

# plot data sampled from the VAR(1) model.
Ycentered <- centerVAR1data(Y)
```

`CIGofVAR1` *Conditional independence graphs of the VAR(1) model*

Description

Constructs the global or contemporaneous conditional independence graph (CIG) of the VAR(1) model, as implied by the partial correlations.

Usage

```
CIGofVAR1(sparseA, sparseP, type="global")
```

Arguments

`sparseA`	A `matrix` $\mathbf{A}$ of autoregression parameters, which is assumed to be sparse.
`sparseP`	Precision `matrix` $\mathbf{\Omega}_{\varepsilon}$ the error, which is assumed to be sparse.
`type`	A `character` indicating whether the `global` or `contemp` (contemporaneous) CIG should be plotted.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Dahlhaus (2000), "Graphical interaction models for multivariate time series", *Metrika*, 51, 157-172.

Dahlhaus, Eichler (2003), "Causality and graphical models in time series analysis", *Oxford Statistical Science Series*, 115-137.

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2017), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

See Also

graphVAR1, sparsify, sparsifyVAR1.

Examples

```
# specify VAR(1) model parameters
A <- matrix(c(-0.1, -0.3,    0,
               0.5,    0,    0,
                 0,    0, -0.4), byrow=TRUE, ncol=3)
P <- matrix(c(   1,  0.5,    0,
               0.5,    1,    0,
                 0,    0,    1), byrow=TRUE, ncol=3)

# adjacency matrix of (global) conditional independencies.
CIGofVAR1(A, P, type="global")
```

CIGofVAR2 *Conditional independence graphs of the VAR(2) model*

Description

Constructs the global or contemporaneous conditional independence graph (CIG) of the VAR(2) model, as implied by the partial correlations.

Usage

```
CIGofVAR2(sparseA1, sparseA2, sparseP, type="global")
```

Arguments

sparseA1	A matrix $\mathbf{A}_1$ of lag one autoregression parameters, which is assumed to be sparse.
sparseA2	A matrix $\mathbf{A}_2$ of lag-two autoregression parameters, which is assumed to be sparse.
sparseP	Precision matrix $\mathbf{\Omega}_\varepsilon$ the error, which is assumed to be sparse.
type	A character indicating whether the global or contemp (contemporaneous) CIG should be plotted.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Dahlhaus (2000), "Graphical interaction models for multivariate time series", *Metrika*, 51, 157-172.

Dahlhaus, Eichler (2003), "Causality and graphical models in time series analysis", *Oxford Statistical Science Series*, 115-137.

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

CIGofVAR1, graphVAR2, sparsify, sparsifyVAR2.

Examples

```
# specify VAR(2) model parameters
A1 <- matrix(c(-0.1, -0.3,    0,
               0.5,    0,    0,
                 0,    0, -0.4), byrow=TRUE, ncol=3)
A2 <- matrix(c(   0,    0,    0,
                  0,    0,  0.2,
                  0, -0.4,    0), byrow=TRUE, ncol=3)
P  <- matrix(c(   1,  0.5,    0,
               0.5,    1,    0,
                 0,    0,    1), byrow=TRUE, ncol=3)

# adjacency matrix of (global) conditional independencies.
CIGofVAR2(A1, A2, P, type="global")
```

---

createA *Generation of the VAR(1) autoregression coefficient matrix.*

Description

Generates autoregression coefficient matrices of the VAR(1) with various type of topologies

Usage

```
createA(p, topology, nonzeroA=0, nCliques=1, nHubs=1,
        nBands=1, percZeros=0.9, stationary=TRUE)
```

Arguments

p	A positive integer specifying the dimension of the square matrix **A**.
topology	Topology to impose on **A**: a character equalling either clique, hub, chain, or random.
nonzeroA	Numeric, value that nonzero elements of **A** will assume. If equal to zero, a random value from the interval [-1,1] is sampled.
nCliques	When topology="clique", this positive integer specifies number of cliques.
nHubs	When topology="hub", this positive integer specifies number of hubs.
nBands	When topology="chain", this positive integer specifies number of bands.
percZeros	When topology="random", the probability with which zero elements of **A** are to be sampled.
stationary	A logical: should the generated **A** be stationary?

Value

A `matrix` with autoregression coefficient matrix **A** of the VAR(1) model.

Author(s)

Viktorian Miok, Wessel N. van Wieringen <w.vanwieringen@vumc.nl>.

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2016), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

See Also

`createS`, `dataVAR1`.

Examples

```
# create the VAR(1) parameters
A  <- createA(10, topology="clique", nonzeroA=0.1, nClique=4)
Se <- createS(1000, 10, "star")

# sample data from the VAR(1) model with above parameters
Y <- dataVAR1(4, 8, A, Se)
```

`dataVAR1` *Sample data from a VAR(1) model*

Description

Sample data from a VAR(1) model.

Usage

```
dataVAR1(n, T, A, SigmaE, TburnIn=1000)
```

Arguments

`n`	Positive `numeric` of length one: number of individuals to be sampled.
`T`	Positive `numeric` of length one: number of time points (per individual) to be sampled.
`A`	`Matrix` **A** of autoregression parameters.
`SigmaE`	Covariance `matrix` of the errors (innovations).
`TburnIn`	Positive `numeric` of length one: number of time points used to burn in the process.

Value

A three dimensional `array` containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

See Also

`ridgeVAR1`.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- matrix(c(-0.1, -0.3,  0.6,
                    0.5, -0.4,    0,
                    0.3, -0.5, -0.2), byrow=TRUE, ncol=3)

# generate data
Y <- dataVAR1(n, T, A, SigmaE)
```

`dataVAR2` *Sample data from a VAR(2) model*

Description

Sample data from a VAR(2) model.

Usage

```
dataVAR2(n, T, A1, A2, SigmaE, TburnIn=1000)
```

Arguments

`n`	Positive `numeric` of length one: number of individuals to be sampled.
`T`	Positive `numeric` of length one: number of time points (per individual) to be sampled.
`A1`	A `matrix` $\mathbf{A}_1$ of lag one autoregression parameters.
`A2`	A `matrix` $\mathbf{A}_2$ of lag two autoregression parameters.
`SigmaE`	Covariance `matrix` of the errors (innovations).
`TburnIn`	Positive `numeric` of length one: number of time points used to burn in the process.

Value

A three dimensional `array` containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

See Also

`dataVAR1`, `dataVAR2`, `ridgeVARX1`, .

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 12; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A1     <- -createA(p, "clique", nCliques=1, nonzeroA=0.1)
A2     <- t(createA(p, "chain", nBands=1, nonzeroA=0.1))

# generate data
Y <- dataVAR2(n, T, A1, A2, SigmaE)
```

`dataVARX1` *Sample data from a VARX(1) model*

Description

Sample data from a VARX(1) model.

Usage

```
dataVARX1(X, A, B, SigmaE, lagX)
```

Arguments

`X`	Three-dimensional `array` containing the time-varying covariates. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`A`	A `matrix` $\mathbf{A}$ of autoregression parameters.
`B`	A `matrix` $\mathbf{B}$ of regression parameters of the time-varying covariates stored in the array `X`.
`SigmaE`	Covariance `matrix` of the errors (innovations).
`lagX`	An `integer`, either `0` or 1, specifying whether $\mathbf{X}_t$ or $\mathbf{X}_{t-1}$ affects $\mathbf{Y}_t$, respectively.

Value

A three dimensional `array` containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

See Also

`createA`, `createS`, `dataVAR1`, `dataVAR2`, `ridgeVARX1`.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 12; T <- 10

# set model parameters
SigmaE <- diag(p)/4
Ax     <- createA(3, "chain")

# generate time-varying covariate data
X <- dataVAR1(n, T, Ax, SigmaE)

# autoregression parameter matrices of VARX(1) model
A <- createA(p, topology="clique", nonzeroA=0.1, nClique=1)
B <- createA(p, topology="hub",    nonzeroA=0.1, nHubs=1)

# generate data
Y <- dataVARX1(X, A, B, SigmaE, lagX=0)
```

---

evaluateVAR1fit *Visualize the fit of a VAR(1) model*

Description

Simple plots for assessment of the fit of an estimated VAR(1) model.

Usage

```
evaluateVAR1fit(Y, A, SigmaE, unbalanced=NULL, diag=FALSE,
                fileType="eps", dir=getwd())
```

Arguments

Y	Three-dimensional array containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
A	A matrix $\mathbf{A}$ of autoregression parameters.
SigmaE	Covariance matrix of the errors (innovations).
unbalanced	A matrix with two columns, indicating the unbalances in the design. Each row represents a missing design point in the (time x individual)-layout. The first and second column indicate the time and individual (respectively) specifics of the missing design point.
diag	A logical, should the diagonal be included in the evaluation of the fit.
fileType	A character specifying the format in which figures should be save. Either 'pdf' or 'eps'.
dir	A character specifying the directory where plots should be saved.

Value

Plots are saved in the specified directory.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

See Also

evaluateSfit, ridgeVAR1, dataVAR1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- createA(p, "chain")

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

# center data
Y <- centerVAR1data(Y)

# fit VAR(1) model
VAR1hat <- ridgeVAR1(Y, 1, 1)

# evaluate fit VAR(1) model
## Not run:  evaluateVAR1fit(Y, VAR1hat$A, symm(VAR1hat$P))
```

graphVAR1 *Graphs of the temporal (or contemporaneous) relations implied by the VAR(1) model*

Description

Graphs of the (conditional independence, temporal, or contemporaneous) relations among the variates as implied by the parameters of the VAR(1) model.

Usage

```
graphVAR1(sparseA, sparseP, type="TSCG", side="left", prune=TRUE,
          nNames=NULL, main=NULL, vertex.color.T0="lightcyan2",
          vertex.color.T1="lightcyan2", vertex.frame.color="steelblue",
          vertex.label.cex=-1, vertex.label.color.T0="black",
          vertex.label.color.T1="black", vertex.label.font=1.5,
          vertex.size=-1, edge.arrow.size=-1, edge.width=-1, ...)
```

Arguments

sparseA	A matrix $\mathbf{A}$ of autoregression parameters, which is assumed to be sparse.
sparseP	Precision matrix $\mathbf{\Omega}_\varepsilon$ the error, which is assumed to be sparse.
type	A character indicating what should be plotted. If type="TSCG". the time series chain graph is plotted, while if type="Aonly" limits this graph to the temporal relations. If type="globalPC" or type="contempPC", the global or contemporaneous (respectively) partial correlation graph is plotted.
side	A character indicating whether the contemporaneous dependencies should be plotted on the left-hand (time t) or the right-hand (time t+1) side. Only active when type="TSCG".
prune	A logical indicating whether to remove covariates without any temporal (or contemporaneous) relations (as implied by sparseA and sparseP).
nNames	A character containing the covariate names to be written inside the nodes.
main	The character to be plotted as title above the graph.
vertex.color.T0	Color of nodes at time point t.
vertex.color.T1	Color of nodes at time point t+1. This is ignored when type="globalPC" or type="contempPC".
vertex.frame.color	Refer to plot.igraph.
vertex.label.cex	Refer to plot.igraph.
vertex.label.color.T0	Color of the node label at time point t.

`vertex.label.color.T1`
: Color of the node label at time point t+1. Ignored when `type="globalPC"` or `type="contempPC"`.

`vertex.label.font`
: Refer to `plot.igraph`.

`vertex.size`
: Refer to `plot.igraph`.

`edge.arrow.size`
: Refer to `plot.igraph`.

`edge.width`
: Refer to `plot.igraph`.

`...`
: Other arguments to be passed on to `plot.igraph`.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2017), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

See Also

`CIGofVAR1`, `sparsify`, `sparsifyVAR1`, `plot.igraph`.

Examples

```
# specify VAR(1) model parameters
A <- matrix(c(-0.1, -0.3,    0,  0.5,    0,    0,
                 0,    0, -0.4, -0.1, -0.3,    0,
               0.5,    0,    0,    0,    0, -0.4,
              -0.1, -0.3,    0,  0.5,    0,    0,
                 0,    0, -0.4, -0.1, -0.3,    0,
               0.5,    0,    0,    0,    0, -0.4), byrow=TRUE, ncol=6)
P <- matrix(c(   2,    0, -0.5,  0.5,    0,  0.5,
                 0,    1,  0.5,  0.5,  0.5,    0,
              -0.5,  0.5,    1,    0,    0,  0.5,
               0.5,  0.5,    0,    1,    0,    0,
                 0,  0.5,    0,    0,    1,    0,
               0.5,    0,  0.5,    0,    0,    1), byrow=TRUE, ncol=6)

# adjacency matrix of (global) conditional independencies.
graphVAR1(A, P, type="TSCG")
```

graphVAR2 *Graphs of the temporal (or contemporaneous) relations implied by the VAR(2) model*

Description

Graphs of the (conditional independence, temporal, or contemporaneous) relations among the variates as implied by the parameters of the VAR(2) model.

Usage

```
graphVAR2(sparseA1, sparseA2, sparseP, type="TSCG", side="right",
          prune=TRUE, nNames=NULL, main=NULL, vertex.color.T0="lightcyan2",
          vertex.color.T1="lightcyan2", vertex.color.T2="lightcyan2",
          vertex.frame.color="steelblue", vertex.label.cex=-1,
          vertex.label.color.T0="black", vertex.label.color.T1="black",
          vertex.label.color.T2="black", vertex.label.font=1.5,
          vertex.size=-1, edge.arrow.size=-1, edge.width=-1, ...)
```

Arguments

sparseA1	A matrix $\mathbf{A}_1$ of lag one autoregression parameters, which is assumed to be sparse.
sparseA2	A matrix $\mathbf{A}_2$ of lag two autoregression parameters, which is assumed to be sparse.
sparseP	Precision matrix $\mathbf{\Omega}_\varepsilon$ the error, which is assumed to be sparse.
type	A character indicating what should be plotted. If type="TSCG". the time series chain graph is plotted, while if type="Aonly" limits this graph to the temporal relations. If type="globalPC" or type="contempPC", the global or contemporaneous (respectively) partial correlation graph is plotted.
side	A character indicating whether the contemporaneous dependencies should be plotted on the left-hand (time t) or the right-hand (time t+1) side. Only active when type="TSCG".
prune	A logical indicating whether to remove covariates without any temporal (or contemporaneous) relations (as implied by sparseA and sparseP).
nNames	A character containing the covariate names to be written inside the nodes.
main	The character to be plotted as title above the graph.
vertex.color.T0	Color of nodes at time point t.
vertex.color.T1	Color of nodes at time point t+1. This is ignored when type="globalPC" or type="contempPC".
vertex.color.T2	Color of nodes at time point t+2. This is ignored when type="globalPC" or type="contempPC".

`vertex.frame.color`
: Refer to `plot.igraph`.

`vertex.label.cex`
: Refer to `plot.igraph`.

`vertex.label.color.T0`
: Color of the node label at time point t.

`vertex.label.color.T1`
: Color of the node label at time point t+1. Ignored when `type="globalPC"` or `type="contempPC"`.

`vertex.label.color.T2`
: Color of the node label at time point t+2. Ignored when `type="globalPC"` or `type="contempPC"`.

`vertex.label.font`
: Refer to `plot.igraph`.

`vertex.size`
: Refer to `plot.igraph`.

`edge.arrow.size`
: Refer to `plot.igraph`.

`edge.width`
: Refer to `plot.igraph`.

`...`
: Other arguments to be passed on to `plot.igraph`.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

`CIGofVAR1`, `sparsify`, `sparsifyVAR1`, `plot.igraph`.

Examples

```
# specify VAR(2) model parameters
A1 <- matrix(c(-0.1, -0.3,    0,  0.5,    0,    0,
                 0,    0, -0.4, -0.1, -0.3,    0,
               0.5,    0,    0,    0,    0, -0.4,
              -0.1, -0.3,    0,  0.5,    0,    0,
                 0,    0, -0.4, -0.1, -0.3,    0,
               0.5,    0,    0,    0,    0, -0.4), byrow=TRUE, ncol=6)
A2 <- matrix(c(   0,    0,  0.2,    0,  0.3, -0.7,
                  0,    0,    0,  0.4,    0,    0,
                  0, -0.3,    0,    0,    0,    0,
                  0,    0,  0.5,    0,  0.1,    0,
                  0,    0,    0,    0,  0.4,    0,
                  0,    0,    0,  0.4,    0,    0), byrow=TRUE, ncol=6)
P <- matrix(c(    2,    0, -0.5,  0.5,    0,  0.5,
```

```
   0,    1,  0.5,  0.5,  0.5,    0,
-0.5,  0.5,    1,    0,    0,  0.5,
 0.5,  0.5,    0,    1,    0,    0,
   0,  0.5,    0,    0,    1,    0,
 0.5,    0,  0.5,    0,    0,    1), byrow=TRUE, ncol=6)

# adjacency matrix of (global) conditional independencies.
graphVAR2(A1, A2, P, type="TSCG")
```

---

graphVARX1 *Graphs of the temporal (or contemporaneous) relations implied by the VARX(1) model*

Description

Graphs of the (temporal and/or contemporaneous) relations among the variates as implied by the parameters of the VARX(1) model.

Usage

```
graphVARX1(sparseA, sparseB, sparseP, type="TSCG", side="right",
           prune=TRUE, nNamesY=NULL, nNamesX=NULL, main=NULL,
           vertex.color.X="lightcyan2", vertex.color.T0="lightcyan2",
           vertex.color.T1="lightcyan2", vertex.frame.color="steelblue",
           vertex.label.cex=-1, vertex.label.color.X="black",
           vertex.label.color.T0="black", vertex.label.color.T1="black",
           vertex.label.font=1.5, vertex.size=-1, edge.arrow.size=-1,
           edge.width=-1, ...)
```

Arguments

sparseA	A matrix $\mathbf{A}$ of lag one autoregression parameters, which is assumed to be sparse.
sparseB	A matrix $\mathbf{B}$ of regression parameters of the time-varying covariates, which is assumed to be sparse.
sparseP	Precision matrix $\mathbf{\Omega}_\varepsilon$ the error, which is assumed to be sparse.
type	A character indicating what should be plotted. If type="TSCG". the time series chain graph is plotted, while if type="ABonly" limits this graph to the temporal relations. If type="contempPC", the global or contemporaneous (respectively) partial correlation graph is plotted.
side	A character indicating whether the contemporaneous dependencies should be plotted on the bottomleft (time t) or the right-hand (time t+1) side. Only active when type="TSCG".
prune	A logical indicating whether to remove covariates without any temporal (or contemporaneous) relations (as implied by sparseA, sparseB and sparseP).
nNamesY	A character containing the variate names to be written inside the nodes.

`nNamesX` A `character` containing the covariate names to be written inside the nodes.

`main` The `character` to be plotted as title above the graph.

`vertex.color.X` Color of covariate nodes.

`vertex.color.T0`
Color of nodes at time point t. This is ignored when `type="contempPC"`.

`vertex.color.T1`
Color of nodes at time point t+1. This is ignored when `type="contempPC"`.

`vertex.frame.color`
Refer to `plot.igraph`.

`vertex.label.cex`
Refer to `plot.igraph`.

`vertex.label.color.X`
Color of the covariate node label.

`vertex.label.color.T0`
Color of the node label at time point t. Ignored when `type="contempPC"`.

`vertex.label.color.T1`
Color of the node label at time point t+1. Ignored when `type="contempPC"`.

`vertex.label.font`
Refer to `plot.igraph`.

`vertex.size` Refer to `plot.igraph`.

`edge.arrow.size`
Refer to `plot.igraph`.

`edge.width` Refer to `plot.igraph`.

`...` Other arguments to be passed on to `plot.igraph`.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

`ridgeVARX1`, `graphVAR1`, `graphVAR2`, `sparsifyVARX1`, `plot.igraph`.

Examples

```
# specify VARX(1) model parameters
A <- matrix(c(-0.1, -0.3,    0,  0.5,    0,    0,
                0,    0, -0.4, -0.1, -0.3,    0,
              0.5,    0,    0,    0,    0, -0.4,
             -0.1, -0.3,    0,  0.5,    0,    0,
                0,    0, -0.4, -0.1, -0.3,    0,
              0.5,    0,    0,    0,    0, -0.4), byrow=TRUE, ncol=6)
```

```
B <- matrix(c(   0,    0,  0.2,    0,  0.3, -0.7,
                 0,    0,    0,  0.4,    0,    0,
                 0, -0.3,    0,    0,    0,    0,
                 0,    0,  0.5,    0,  0.1,    0,
                 0,    0,    0,    0,  0.4,    0,
                 0,    0,    0,  0.4,    0,    0), byrow=TRUE, ncol=6)
P <- matrix(c(   2,    0, -0.5,  0.5,    0,  0.5,
                 0,    1,  0.5,  0.5,  0.5,    0,
              -0.5,  0.5,    1,    0,    0,  0.5,
               0.5,  0.5,    0,    1,    0,    0,
                 0,  0.5,    0,    0,    1,    0,
               0.5,    0,  0.5,    0,    0,    1), byrow=TRUE, ncol=6)

# time-series chain graph  of the VARX(1) model
graphVARX1(A, B, P, type="TSCG")
```

---

hpvP53 *Time-course P53 pathway data*

Description

Time-course omics data of the P53-signalling pathway during HPV-induced transformation. Four cell lines transfected with the human papilloma virus (HPV) are cultured in petri dishes. At eight time points the cells from each cell line are molecularly interrogated genomically and transcriptomically (both mRNA and microRNA expression). From each molecular levels data DNA segments/mRNAs/microRNAs mapping to the P53-signalling pathway (as defined by KEGG, Ogata et al., 1999) are included. The included microRNAs are predicted to target the mRNAs of the pathway. Finally, two adjacency matrices are included. One describes how the DNA copy number features link to that of the mRNA expression data. The map is based on the feature's genomic location, while the other specifies which microRNAs (columns) target which mRNA (rows) according target predicton data bases. Confer Wilting et al. (2017) for preprocessing details.

Usage

```
data(hpvP53)
```

Value

hpvP53rna	An object of class ExpressionSet with the time-course mRNA expression data of $32 (= 4$ cell lines $\times 8$ time points$)$ observations.
hpvP53mir	An object of class ExpressionSet with the time-course microRNA expression data of $32 (= 4$ cell lines $\times 8$ time points$)$ observations.
hpvP53cn	An object of class cghRaw with the time-course DNA copy number data of $32 (= 4$ cell lines $\times 8$ time points$)$ observations.
mir2rna	An object of class matrix. Adjancency matrix indicating which microRNAs (features in hpvP53mir) target which mRNAs (features in hpvP53rna). Rows = mRNAs; columns = microRNAs.
cn2rna	An object of class matrix. Adjancency matrix indicating which DNA copy number features (in hpvP53cn) map to which mRNAs (in hpvP53rna).

Source

Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., Kanehisa, M. (1999), "KEGG: Kyoto Encyclopedia of Genes and Genomes". *Nucleic Acids Research*, 27(1), 29-34.

Wilting, S.M., Miok, V., Jaspers, A., Boon, D., Hanne Sorgard, H., Lando, M., Snoek, B.C., Van Wieringen, W.N., Meijer, C.J.L.M., Lyng, H., Snijders, P.J.F., Steenbergen, R.D.M. (2016), "Aberrant methylation-mediated silencing of microRNAs contributes to HPV-induced anchorage independence", *Oncotarget*, Vol. 7, No. 28, 43805-43819, DOI:10.18632/oncotarget.9698 .

Examples

```
data(hpvP53)
```

---

impulseResponseVAR1 *Impulse response analysis of the VAR(1) model*

Description

Evaluate the impulse responses of the VAR(1) model. It assesses the effect of an innovation (error) at one time point on the variates at future time points. In the VAR(1) model this amounts to studying powers of $\mathbf{A}$, the matrix of autoregression coefficients.

Usage

```
impulseResponseVAR1(A, T)
```

Arguments

A	A `matrix` $\mathbf{A}$ of autoregression parameters.
T	Positive `numeric` of length one specifying the time points for which the impulse response is to be evaluated.

Value

Object of class `matrix`. Rows and columns correspond to covariates, elements to the impulse response of 'row variate' on the 'columns variate' on `T` time points from the current.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Hamilton, J. D. (1994), *Time Series Analysis*. Princeton: Princeton university press.

Lutkepohl, H. (2005), *New Introduction to Multiple Time Series Analysis*. Springer, Berlin.

See Also

ridgeVAR1.

Examples

```
# set dimensions
p <- 3
n <- 4
T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- matrix(c(-0.1, -0.3, 0.6, 0.5, -0.4, 0, 0.3, -0.5, -0.2),
                 byrow=TRUE, ncol=3)

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

# fit VAR(1) model
VAR1hat <- ridgeVAR1(Y, 1, 1)

# impulse response analysis
impulseResponseVAR1(VAR1hat$A, 10)
```

---

impulseResponseVAR2 *Impulse response analysis of the VAR(2) model*

Description

Evaluate the impulse responses of the VAR(2) model. It assesses the effect of an innovation (error) at one time point on the variates at future time points. In the VAR(2) model this amounts evaluating a recursive relationship in $\mathbf{A}_1$ and $\mathbf{A}_2$, the matrices of lag 1 and lag 2 autoregression coefficients.

Usage

```
impulseResponseVAR2(A1, A2, T)
```

Arguments

A1	A matrix $\mathbf{A}_1$ with lag 1 autoregression parameters.
A2	A matrix $\mathbf{A}_2$ with lag 2 autoregression parameters.
T	Non-negative integer of length one specifying the time point at which the impulse responses is to be evaluated.

Value

A matrix with the impulse response of the innovation vector.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Hamilton, J. D. (1994). *Time series analysis*. Princeton: Princeton university press.

Lutkepohl, H. (2005). *New Introduction to Multiple Time Series Analysis*. Springer, Berlin.

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

`impulseResponseVAR1`, `impulseResponseVARX1`, `ridgeVAR2`.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 12; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A1     <- createA(p, "clique", nCliques=1)
A2     <- createA(p, "hub", nHubs=1)

# generate time-varying covariates in accordance with VAR(2) process
Y <- dataVAR2(n, T, A1, A2, SigmaE)

# fit VAR(2) model
VAR2hat <- ridgeVAR2(Y, 1, 1, 1)

# impulse response analysis
impulseResponseVAR2(VAR2hat$A1, VAR2hat$A2, 10)
```

---

impulseResponseVARX1 *Impulse response analysis of the VARX(1) model*

Description

Evaluate the impulse responses of the VARX(1) model. It assesses the effect of an innovation (error) at one time point on the variates at future time points. In the VARX(1) model this amounts to studying powers of $\mathbf{A}$, the matrix of autoregression coefficients, for the innovation. For the impulse response of the time-varying covariates these powers are post-multiplied by $\mathbf{B}$.

Usage

```
impulseResponseVARX1(A, B, T)
```

Arguments

A	A matrix $\mathbf{A}$ with autoregression parameters.
B	A matrix $\mathbf{B}$ with regression parameters time-varying covariates.
T	Non-negative integer of length one specifying the time point at which the impulse responses is to be evaluated.

Value

A list-object with slots:

impResponse	A matrix with the impulse response of the innovation vector.
impResponseX	A matrix with the impulse response of the time-varying covariate vector.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Hamilton, J. D. (1994). *Time series analysis*. Princeton: Princeton university press.

Lutkepohl, H. (2005). *New Introduction to Multiple Time Series Analysis*. Springer, Berlin.

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

ridgeVAR1, ridgeVAR2, ridgeVARX1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 12; T <- 10

# set model parameters
SigmaE <- diag(p)/4
Ax     <- createA(p, "chain", nBands=1)

# generate time-varying covariates in accordance with VAR(1) process
X <- dataVAR1(n, T, Ax, SigmaE)

# set model parameters
B <- createA(p, "clique", nCliques=1)
A <- createA(p, "hub", nHubs=1)

# generate time-varying covariates in accordance with VAR(1) process
Y <- dataVARX1(X, A, B, SigmaE, lagX=0)

# fit VARX(1) model
VARX1hat <- ridgeVARX1(Y, X, 1, 1, 1, lagX=0)
```

```
# impulse response analysis
impulseResponseVARX1(VARX1hat$A, VARX1hat$B, 10)
```

---

loglikLOOCVcontourVAR1 *Contourplot of LOOCV log-likelihood of VAR(1) model*

Description

Evaluates the leave-one-out cross-validated log-likelihood of the VAR(1) model for a given grid of the ridge penalty parameters (λ_a and λ_ω) for the autoregression coefficient matrix $\mathbf{A}$ and the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$, respectively). The result is plotted as a contour plot, which facilitates the choice of optimal penalty parameters. The function also works with a (possibly) unbalanced experimental set-up. The VAR(1)-process is assumed to have mean zero.

Usage

```
loglikLOOCVcontourVAR1(lambdaAgrid, lambdaPgrid, Y, figure=TRUE,
                       verbose=TRUE, ...)
```

Arguments

lambdaAgrid	A numeric of length larger than one, comprising positive numbers only. It contains the grid points corresponding to the λ_a (the penalty parameter for the autoregression coefficient matrix $\mathbf{A}$).
lambdaPgrid	A numeric of length larger than one, comprising positive numbers only. It contains the grid points corresponding to the λ_ω (the penalty parameters for the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$).
Y	Three-dimensional array containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to centered covariate-wise.
figure	A logical, indicating whether the contour plot should be generated.
verbose	A logical indicator: should intermediate output be printed on the screen?
...	Other arguments to be passed on (indirectly) to ridgeVAR1.

Value

A list-object with slots:

lambdaA	A numeric with the grid points corresponding to λ_a (the penalty parameter for the autoregression coefficient matrix $\mathbf{A}$).
lambdaP	A numeric with the grid points corresponding to λ_ω (the penalty parameter for the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$).
llLOOCV	A matrix of leave-one-out cross-validated log-likelihoods. Rows and columns correspond to λ_a and λ_ω values, respectively.

Note

Internally, this function calls the loglikLOOCVVAR1-function, which evaluates the minus (!) LOOCV log-likelihood (for practical reasons). For interpretation purposes loglikLOOCVcontourVAR1 provides the regular LOOCV log-likelihood (that is, without the minus).

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2017), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

See Also

loglikLOOCVVAR1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A <- createA(p, "chain")

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

## plot contour of cross-validated likelihood
## Not run:  lambdaAgrid <- seq(0.01, 1, length.out=20)
## Not run:  lambdaPgrid <- seq(0.01, 1000, length.out=20)
## Not run:  loglikLOOCVcontourVAR1(lambdaAgrid, lambdaPgrid, Y)

## determine optimal values of the penalty parameters
## Not run: optLambdas <- constrOptim(c(1,1), loglikLOOCVVAR1, gr=NULL,
## Not run:               ui=diag(2), ci=c(0,0), Y=Y,
## Not run:               control=list(reltol=0.01))$par

## add point of optimum
## Not run:  points(optLambdas[1], optLambdas[2], pch=20, cex=2,
## Not run:  col="red")
```

loglikLOOCVcontourVAR1fused
Contourplot of LOOCV log-likelihood of multiple VAR(1) models

Description

Evaluates the leave-one-out cross-validated log-likelihood of multiple jointly estimated VAR(1) models over a grid of the (fused) ridge penalty parameters (λ_a and λ_f) for the autoregression coefficient matrices $\mathbf{A}_g$, while keeping λ_ω, the penalty parameter of the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$, fixed at a user-specified value. The result is plotted as a contour plot, which facilitates the choice of optimal penalty parameters. The function also works with a (possibly) unbalanced experimental set-up. The VAR(1)-processes are assumed to have mean zero.

Usage

```
loglikLOOCVcontourVAR1fused(lambdaAgrid, lambdaFgrid, Y, id,
                            lambdaP, figure=TRUE, verbose=TRUE, ...)
```

Arguments

lambdaAgrid	A numeric of length larger than one, comprising positive numbers only. It contains the grid points corresponding to λ_a, the ridge penalty parameter for the autoregression coefficient matrices $\mathbf{A}_g$.
lambdaFgrid	A numeric of length larger than one, comprising positive numbers only. It contains the grid points corresponding to λ_f, the fused ridge penalty parameter for the autoregression coefficient matrices $\mathbf{A}_g$.
Y	Three-dimensional array containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
id	A vector with group indices comprising of integers only. First group is represented by '0', the next by '1', and so on until the last.
lambdaP	A numeric specifying the value at which λ_ω should be fixed.
figure	A logical, indicating whether the contour plot should be generated.
verbose	A logical indicator: should intermediate output be printed on the screen?
...	Other arguments to be passed on (indirectly) to ridgeVAR1fused.

Value

A list-object with slots:

lambdaA	A numeric with the grid points corresponding to λ_a (the ridge penalty parameter for the autoregression coefficient matrices $\mathbf{A}_g$).
lambdaF	A numeric with the grid points corresponding to λ_f (the fused ridge penalty parameter for the autoregression coefficient matrices $\mathbf{A}_g$.
llLOOCV	A matrix of leave-one-out cross-validated log-likelihoods. Rows and columns correspond to λ_a and λ_ω values, respectively.

Note

Internally, this function calls the loglikLOOCVVAR1-function, which evaluates the minus (!) LOOCV log-likelihood (for practical reasons). For interpretation purposes loglikLOOCVcontourVAR1 provides the regular LOOCV log-likelihood (that is, without the minus).

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>, Viktorian Miok.

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

loglikLOOCVcontourVAR1, loglikLOOCVcontourVARX1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A <- createA(p, "chain")

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

## plot contour of cross-validated likelihood
## Not run:  lambdaAgrid <- seq(0.01, 1, length.out=20)
## Not run:  lambdaPgrid <- seq(0.01, 1000, length.out=20)
## Not run:  loglikLOOCVcontourVAR1(lambdaAgrid, lambdaPgrid, Y)

## determine optimal values of the penalty parameters
## Not run: optLambdas <- constrOptim(c(1,1), loglikLOOCVVAR1, gr=NULL,
## Not run:               ui=diag(2), ci=c(0,0), Y=Y,
## Not run:               control=list(reltol=0.01))$par

## add point of optimum
## Not run:  points(optLambdas[1], optLambdas[2], pch=20, cex=2,
## Not run:  col="red")
```

loglikLOOCVcontourVAR2 *Contourplot of LOOCV log-likelihood of the VAR(2) model*

Description

Evaluates the leave-one-out cross-validated log-likelihood of the estimated VAR(2) model over a grid of the ridge penalty parameters (λ_{a1} and λ_{a2}) for the lag one and lag two autoregression coefficient matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ (respectively), while keeping λ_ω, the penalty parameter of the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$, fixed at a user-specified value. The result is plotted as a contour plot, which facilitates the choice of optimal penalty parameters. The function also works with a (possibly) unbalanced experimental set-up. The VAR(2)-process are assumed to have mean zero.

Usage

```
loglikLOOCVcontourVAR2(lambdaA1grid, lambdaA2grid,
                       lambdaPgrid, Y, figure=TRUE, verbose=TRUE, ...)
```

Arguments

lambdaA1grid	A numeric of length larger than one, comprising positive numbers only. It contains the grid points corresponding to λ_{a1}, the ridge penalty parameter for the lag one autoregression coefficient matrix $\mathbf{A}_1$.
lambdaA2grid	A numeric of length larger than one, comprising positive numbers only. It contains the grid points corresponding to λ_{a2}, the ridge penalty parameter for the lag two autoregression coefficient matrix $\mathbf{A}_2$.
lambdaPgrid	A numeric of length larger than one, comprising positive numbers only. It contains the grid points corresponding to the λ_ω (the penalty parameters for the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$).
Y	Three-dimensional array containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
figure	A logical, indicating whether the contour plot should be generated.
verbose	A logical indicator: should intermediate output be printed on the screen?
...	Other arguments to be passed on (indirectly) to ridgeVAR2.

Value

A list-object with slots:

lambdaA1	A numeric with the grid points corresponding to λ_{a1} (the ridge penalty parameter for the lag one autoregression coefficient matrix $\mathbf{A}_1$).
lambdaA2	A numeric with the grid points corresponding to λ_{a2} (the ridge penalty parameter for the lag one autoregression coefficient matrix $\mathbf{A}_2$).

`lambdaP`	A `numeric` with the grid points corresponding to λ_ω (the penalty parameter for the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$).
`llLOOCV`	A `matrix` of leave-one-out cross-validated log-likelihoods. Rows and columns correspond to λ_{a1} and λ_{a2} values, respectively.

Note

When `lambdaA1grid`, `lambdaA2grid` and `lambdaPgrid` are all vectors of length exceeding one, the contour is determined for the grid formed by the Cartesius product of `lambdaA1grid` and `lambdaA2grid` meanwhile restricting `lambdaPgrid` to its first element.

Internally, this function calls the `loglikLOOCVVAR2`-function, which evaluates the minus (!) LOOCV log-likelihood (for practical reasons). For interpretation purposes `loglikLOOCVcontourVAR2` provides the regular LOOCV log-likelihood (that is, without the minus).

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>, Viktorian Miok.

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

`loglikLOOCVcontourVAR1`, `loglikLOOCVcontourVARX1`.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 12; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A1 <- -createA(p, "clique", nCliques=1, nonzeroA=0.1)
A2 <- t(createA(p, "chain", nBands=1,   nonzeroA=0.1))

# generate data
Y <- dataVAR2(n, T, A1, A2, SigmaE)

## plot contour of cross-validated likelihood
## Not run:  lambdaA1grid <- seq(0.01, 1, length.out=20)
## Not run:  lambdaA2grid <- seq(0.01, 1000, length.out=20)
## Not run:  lambdaPgrid  <- seq(0.01, 1000, length.out=20)
## Not run:  loglikLOOCVcontourVAR2(lambdaA1grid, lambdaA2grid, lambdaPgrid, Y)

## determine optimal values of the penalty parameters
## Not run: optLambdas <- constrOptim(c(1,1,1), loglikLOOCVVAR2, gr=NULL,
## Not run:                ui=diag(3), ci=c(0,0,0), Y=Y,
## Not run:                control=list(reltol=0.01))$par
```

```
## add point of optimum
## Not run:  points(optLambdas[1], optLambdas[2], pch=20, cex=2,
## Not run:  col="red")
```

`loglikLOOCVcontourVARX1` *Contourplot of the LOOCV log-likelihood of VARX(1) model*

Description

Evaluates the leave-one-out cross-validated log-likelihood of the VARX(1) model over a grid of the ridge penalty parameters (λ_a and λ_b) for the autoregression and time-varying covariate regression coefficient matrices $\mathbf{A}$ and $\mathbf{B}$, respectively, while keeping λ_ω, the penalty parameter of the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$, fixed at a user-specified value. The result is plotted as a contour plot, which facilitates the choice of optimal penalty parameters. The function also works with a (possibly) unbalanced experimental set-up. The VARX(1)-process is assumed to have mean zero.

Usage

```
loglikLOOCVcontourVARX1(lambdaAgrid, lambdaBgrid, lambdaPgrid, Y, X,
                        lagX=0, figure=TRUE, verbose=TRUE, ...)
```

Arguments

`lambdaAgrid`	A `numeric` of length larger than one, comprising positive numbers only. It contains the grid points corresponding to the λ_a (the penalty parameter for the lag one autoregression coefficient matrix $\mathbf{A}$).
`lambdaBgrid`	A `numeric` of length larger than one, comprising positive numbers only. It contains the grid points corresponding to the λ_b (the penalty parameters for the regression coefficient matrix $\mathbf{B}$ of the time-varying covariates.
`lambdaPgrid`	A `numeric` of length larger than one, comprising positive numbers only. It contains the grid points corresponding to the λ_ω (the penalty parameters for the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$).
`Y`	Three-dimensional `array` containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered variate-wise.
`X`	Three-dimensional `array` containing the time-varying covariates. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`lagX`	An `integer`, either `0` or 1, specifying whether $\mathbf{X}_t$ or $\mathbf{X}_{t-1}$ affects $\mathbf{Y}_t$, respectively.
`figure`	A `logical` indicating whether the contour plot should be generated.
`verbose`	A `logical` indicator: should intermediate output be printed on the screen?
`...`	Other arguments to be passed on (indirectly) to `ridgeVARX1`.

Value

A `list`-object with slots:

`lambdaA`	A `numeric` with the grid points corresponding to λ_a (the penalty parameter for the autoregression coefficient matrix $\mathbf{A}$).
`lambdaB`	A `numeric` with the grid points corresponding to λ_b (the penalty parameter for time-varying covariate regression coefficient matrix $\mathbf{B}$.
`lambdaP`	A `numeric` with the grid points corresponding to λ_ω (the penalty parameter for the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$.
`llLOOCV`	A `matrix` of leave-one-out cross-validated log-likelihoods. Rows and columns correspond to λ_a and λ_b values, respectively.

Note

When `lambdaAgrid`, `lambdaBgrid` and `lambdaPgrid` are all vectors of length exceeding one, the contour is determined for the grid formed by the Cartesius product of `lambdaAgrid` and `lambdaBgrid` meanwhile restricting `lambdaPgrid` to its first element.

Internally, this function calls the `loglikLOOCVVARX1`-function, which evaluates the minus (!) LOOCV log-likelihood (for practical reasons). For interpretation purposes `loglikLOOCVcontourVARX1` provides the regular LOOCV log-likelihood (that is, without the minus).

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>, Viktorian Miok.

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

`loglikLOOCVcontourVAR1`, `loglikLOOCVcontourVAR1fused`.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 12; T <- 10

# set model parameters
SigmaE <- diag(p)/4
Ax     <- createA(3, "chain")

# generate time-varying covariate data
X <- dataVAR1(n, T, Ax, SigmaE)

# regression parameter matrices of VARX(1) model
A <- createA(p, topology="clique", nonzeroA=0.1, nClique=1)
B <- createA(p, topology="hub", nonzeroA=0.1, nHubs=1)
```

```
# generate data
Y <- dataVARX1(X, A, B, SigmaE, lagX=0)

## plot contour of cross-validated likelihood
## Not run:  lambdaAgrid <- seq(0.01, 1, length.out=20)
## Not run:  lambdaBgrid <- seq(0.01, 1000, length.out=20)
## Not run:  lambdaPgrid <- seq(0.01, 1000, length.out=20)
## Not run:  loglikLOOCVcontourVARX1(lambdaAgrid, lambdaBgrid, lambdaPgrid, Y, X)

## determine optimal values of the penalty parameters
## Not run: optLambdas <- constrOptim(c(1,1,1), loglikLOOCVVARX1, gr=NULL,
## Not run:               ui=diag(3), ci=c(0,0,0), Y=Y, X=X,
## Not run:               control=list(reltol=0.01))$par

## add point of optimum
## Not run:  points(optLambdas[1], optLambdas[2], pch=20, cex=2,
## Not run:  col="red")
```

`loglikLOOCVVAR1` *Leave-one-out (minus) cross-validated log-likelihood of VAR(1) model*

Description

Evaluation of the (minus) leave-one-out cross-validated log-likelihood of the VAR(1) model for given choices of the ridge penalty parameters (λ_a and λ_ω for the autoregression coefficient matrix $\mathbf{A}$ and the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$, respectively). The functions also works with a (possibly) unbalanced experimental set-up. The VAR(1)-process is assumed to have mean zero.

Usage

```
loglikLOOCVVAR1(lambdas, Y, unbalanced=matrix(nrow=0, ncol=2), ...)
```

Arguments

`lambdas`	A `numeric` of length two, comprising positive values only. It contains the ridge penalty parameters to be used in the estimation of $\mathbf{A}$ and the precision matrix of the errors, respectively.
`Y`	Three-dimensional `array` containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`unbalanced`	A `matrix` with two columns, indicating the unbalances in the design. Each row represents a missing design point in the (time x individual)-layout. The first and second column indicate the time and individual (respectively) specifics of the missing design point.
`...`	Other arguments to be passed to `ridgeVAR1`.

Value

A numeric of length one: the minus (!) LOOCV log-likelihood.

Note

The minus LOOCV log-likelihood is returned as standard optimization procedures in R like nlminb and constrOptim minimize (rather then maximize). Hence, by providing the minus LOOCV log-likelihood the function loglikLOOCVVAR1 can directly used by these optimization procedures.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2017), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

See Also

ridgeP and ridgeVAR1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- createA(p, "chain")

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

## determine optimal values of the penalty parameters
## Not run: optLambdas <- constrOptim(c(1,1), loglikLOOCVVAR1, gr=NULL,
## Not run:               ui=diag(2), ci=c(0,0), Y=Y,
## Not run:               control=list(reltol=0.01))$par

# ridge ML estimation of the VAR(1) parameter estimates with
# optimal penalty parameters
optLambdas <- c(0.1, 0.1)
ridgeVAR1(Y, optLambdas[1], optLambdas[2])$A
```

`loglikLOOCVVAR1fused` *Leave-one-out (minus) cross-validated log-likelihood of multiple VAR(1) models*

Description

Evaluation of the (minus) leave-one-out cross-validated log-likelihood of multiple VAR(1) models for given choices of the (fused) ridge penalty parameters (λ_a, λ_f and λ_ω for the autoregression coefficient matrice $\mathbf{A}_g$'s and the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$, respectively). The functions also works with a (possibly) unbalanced experimental set-up. The VAR(1)-processes are assumed to have mean zero.

Usage

```
loglikLOOCVVAR1fused(lambdas, Y, id, unbalanced=matrix(nrow=0, ncol=2), ...)
```

Arguments

`lambdas`	A `numeric` of length three, comprising positive values only. It contains the ridge penalty parameters to be used in the estimation of $\mathbf{A}$ and the precision matrix of the errors, respectively.
`Y`	Three-dimensional `array` containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`id`	A vector with group indices comprising of integers only. First group is represented by '0', the next by '1', and so on until the last.
`unbalanced`	A `matrix` with two columns, indicating the unbalances in the design. Each row represents a missing design point in the (time x individual)-layout. The first and second column indicate the time and individual (respectively) specifics of the missing design point.
`...`	Other arguments to be passed to `ridgeVAR1`.

Value

A `numeric` of length one: the minus (!) LOOCV log-likelihood.

Note

The minus LOOCV log-likelihood is returned as standard optimization procedures in R like `nlminb` and `constrOptim` minimize (rather then maximize). Hence, by providing the minus LOOCV log-likelihood the function `loglikLOOCVVAR1fused` can directly used by these optimization procedures.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

ridgeP and ridgeVAR1fused.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points, G=groups)
p <- 3; n <- 12; T <- 10; G <- 3

# set model parameters
SigmaE <- matrix(1/2, p, p)
diag(SigmaE) <- 1
A1 <- -createA(p, "clique", nCliques=1, nonzeroA=0.1)
A2 <- t(createA(p, "chain", nBands=1,   nonzeroA=0.1))
A3 <- (A1 + A2) / 2

# generate data
Y1 <- dataVAR1(n/G, T, A1, SigmaE)
Y2 <- dataVAR1(n/G, T, A2, SigmaE)
Y3 <- dataVAR1(n/G, T, A3, SigmaE)
Y  <- abind::abind(Y1, Y2, Y3, along=3)
id <- c(rep(1, n/G), rep(2, n/G), rep(3, n/G))-1

## determine optimal values of the penalty parameters
## Not run: optLambdas <- constrOptim(c(1,1,1), loglikLOOCVVAR1fused, gr=NULL,
## Not run:                ui=diag(3), ci=c(0,0,0), Y=Y, id=id
## Not run:                control=list(reltol=0.01))$par

# ridge ML estimation of the VAR(1) parameter estimates with
# optimal penalty parameters
optLambdas <- c(0.1, 0.1, 0.1)
VAR1hats <- ridgeVAR1fused(Y, id, optLambdas[1], optLambdas[2], optLambdas[3])
```

---

loglikLOOCVVAR2 *Leave-one-out (minus) cross-validated log-likelihood of VAR(2) model*

Description

Evaluation of the (minus) leave-one-out cross-validated log-likelihood of the VAR(2) model for given choices of the ridge penalty parameters (λ_{a1}, λ_{a2} and λ_{ω} for the lag one autoregression coefficient matrix $\mathbf{A}_1$, lag two autoregression coefficient matrix $\mathbf{A}_2$ of time-varying covariates, and the inverse error covariance matrix $\boldsymbol{\Omega}_{\varepsilon} (= \boldsymbol{\Sigma}_{\varepsilon}^{-1})$, respectively). The functions also works with a (possibly) unbalanced experimental set-up. The VAR(2)-process is assumed to have mean zero.

Usage

```
loglikLOOCVVAR2(lambdas, Y, unbalanced=matrix(nrow=0, ncol=2), ...)
```

Arguments

lambdas	A numeric of length three, comprising positive values only. It contains the ridge penalty parameters to be used in the estimation of $\mathbf{A}_1$, $\mathbf{A}_2$ and the precision matrix of the errors.
Y	Three-dimensional array containing the response data. The first, second and third dimensions correspond to variates, time and samples, respectively. The data are assumed to be centered covariate-wise.
unbalanced	A matrix with two columns, indicating the unbalances in the design. Each row represents a missing design point in the (time x individual)-layout. The first and second column indicate the time and individual (respectively) specifics of the missing design point.
...	Other arguments to be passed to the ridgeVAR2-function.

Value

A numeric of length one: the minus (!) LOOCV log-likelihood.

Note

The minus LOOCV log-likelihood is returned as standard optimization procedures in R like nlminb and constrOptim minimize (rather then maximize). Hence, by providing the minus LOOCV log-likelihood the function loglikLOOCVVAR2 can directly used by these optimization procedures.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

ridgeP and ridgeVAR2.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 12; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A1 <- createA(p, topology="clique", nonzeroA=0.1, nClique=1)
A2 <- createA(p, topology="hub", nonzeroA=0.1, nHubs=1)
```

```
# generate data
Y <- dataVAR2(n, T, A1, A2, SigmaE)

## determine optimal values of the penalty parameters
## Not run: optLambdas <- constrOptim(c(1,1, 1), loglikLOOCVVAR2, gr=NULL,
## Not run:               ui=diag(3), ci=c(0,0,0), Y=Y,
## Not run:               control=list(reltol=0.01))$par

# ridge ML estimation of the VAR(2) parameter estimates with
# optimal penalty parameters
optLambdas <- c(0.1, 0.1, 0.1)
ridgeVAR2(Y, optLambdas[1], optLambdas[2], optLambdas[3])
```

`loglikLOOCVVARX1` *Leave-one-out (minus) cross-validated log-likelihood of VARX(1) model*

Description

Evaluation of the (minus) leave-one-out cross-validated log-likelihood of the VARX(1) model for given choices of the ridge penalty parameters (λ_a, λ_b and λ_ω for the autoregression coefficient matrix $\mathbf{A}$, regression coefficient matrix $\mathbf{B}$ of time-varying covariates, and the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$, respectively). The functions also works with a (possibly) unbalanced experimental set-up. The VARX(1)-process is assumed to have mean zero.

Usage

```
loglikLOOCVVARX1(lambdas, Y, X, unbalanced=matrix(nrow=0, ncol=2), lagX=0, ...)
```

Arguments

`lambdas`	A `numeric` of length three, comprising positive values only. It contains the ridge penalty parameters to be used in the estimation of $\mathbf{A}$, $\mathbf{B}$ and the precision matrix of the errors, respectively.
`Y`	Three-dimensional `array` containing the response data. The first, second and third dimensions correspond to variates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`X`	Three-dimensional `array` containing the time-varying covariate data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`unbalanced`	A `matrix` with two columns, indicating the unbalances in the design. Each row represents a missing design point in the (time x individual)-layout. The first and second column indicate the time and individual (respectively) specifics of the missing design point.
`lagX`	`Integer`, either `0` or 1, specifying whether $\mathbf{X}_t$ or $\mathbf{X}_{t-1}$ affects $\mathbf{Y}_t$, respectively.
`...`	Other arguments to be passed to `ridgeVARX1`.

Value

A numeric of length one: the minus (!) LOOCV log-likelihood.

Note

The minus LOOCV log-likelihood is returned as standard optimization procedures in R like nlminb and constrOptim minimize (rather then maximize). Hence, by providing the minus LOOCV log-likelihood the function loglikLOOCVVARX1 can directly used by these optimization procedures.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

ridgeP and ridgeVARX1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 12; T <- 10

# set model parameters
SigmaE <- diag(p)/4
Ax     <- createA(3, "chain")

# generate time-varying covariate data
X <- dataVAR1(n, T, Ax, SigmaE)

# regression parameter matrices of VARX(1) model
A <- createA(p, topology="clique", nonzeroA=0.1, nClique=1)
B <- createA(p, topology="hub", nonzeroA=0.1, nHubs=1)

# generate data
Y <- dataVARX1(X, A, B, SigmaE, lagX=0)

## determine optimal values of the penalty parameters
## Not run: optLambdas <- constrOptim(c(1,1, 1), loglikLOOCVVARX1, gr=NULL,
## Not run:               ui=diag(3), ci=c(0,0,0), Y=Y, X=X, lagX=0,
## Not run:               control=list(reltol=0.01))$par

# ridge ML estimation of the VAR(1) parameter estimates with
# optimal penalty parameters
optLambdas <- c(0.1, 0.1, 0.1)
ridgeVARX1(Y, X, optLambdas[1], optLambdas[2], optLambdas[3], lagX=0)$A
```

`loglikVAR1` *Log-likelihood of the VAR(1) model.*

Description

Log-likelihood of the VAR(1) model specified by the supplied parameters

Usage

```
loglikVAR1(Y, A, P, unbalanced=matrix(nrow=0, ncol=2))
```

Arguments

`Y`	Three-dimensional `array` containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`A`	A `matrix` $\mathbf{A}$ of autoregression parameters.
`P`	Inverse error covariance `matrix` $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$.
`unbalanced`	A `matrix` with two columns, indicating the unbalances in the design. Each row represents a missing design point in the (time x individual)-layout. The first and second column indicate the time and individual (respectively) specifics of the missing design point.

Value

The log-likelihood of the VAR(1) model with supplied parameters.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

See Also

`ridgeVAR1`.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- createA(p, "chain")

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

# center data
```

```
Y <- centerVAR1data(Y)

# fit VAR(1) model
VAR1hat <- ridgeVAR1(Y, 1, 1)

# evaluate the log-likelihood of this fit.
loglikVAR1(Y, VAR1hat$A, VAR1hat$P)
```

`longitudinal2array` *Convert a longitudinal object into an array.*

Description

Converts an object of the `longitudinal`-class into a 3-dim array (containing time-series data of multiple individuals).

Usage

```
longitudinal2array(Y)
```

Arguments

`Y`	Object of the `longitudinal` class. Essentially, this is a matrix with the time-point slices of the array stacked on top of each other.

Value

A `array`-object containing time-series data.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

See Also

`array2longitudinal`, `dataVAR1`, `longitudinal`, `as.longitudinal`.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- createA(p, "chain")

# generate data
Y <- dataVAR1(n, T, A, SigmaE)
```

```
# convert data to a longitudinal-object
Ylongitudinal <- array2longitudinal(Y)

# convert back to array-format
Yback <- longitudinal2array(Ylongitudinal)
```

`motifStatsVAR1` *Network motif detection for the VAR(1) model.*

Description

Function that detects standard network motifs from the lag-one autoregression relationships as implied by the VAR(1) model.

Usage

```
motifStatsVAR1(sparseA, verbose=TRUE)
```

Arguments

`sparseA`	A `matrix` $\mathbf{A}$ of autoregression parameters, which is assumed to be sparse.
`verbose`	A `logical` specifying whether summary output should be displayed.

Details

Six types of motifs are detected: self-regulators, feedback pairs, feedforward loops, feedback loops, bi-fans, and diamonds (cf. Alon, 2007 for details). A detected motif is reported by the paths that constitute them. In line with Alon (2007), who distinguishes subtypes of these motived based on the sign of the path's contribution, the latter is also reported. When `verbose=TRUE` the summary output is also visualized. See this plot for the definition of the motifs in terms of the VAR(1) time series chain graph.

Value

An object of class `list` with slots:

`selfregulators`	A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.
`feedbackpairs`	A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.
`feedforwardloops`	A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.
`feedbackloops`	A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.
`bifans`	A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.

diamonds	A list of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.

Future versions of this function may include additional slots reporting more motif types.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>.

References

Alon, U. (2007), "Network motifs: theory and experimental approaches", *Nature Reviews Genetics*, 8, 450-461.

See Also

ridgeVAR1, sparsifyVAR1, graphVAR1, nodeStatsVAR1

Examples

```
# specify lag one autoregression model
sparseA <- matrix(runif(2500), ncol=50)
sparseA[sparseA < 0.9] <- 0

# find motifs
motifList <- motifStatsVAR1(sparseA)
```

---

mutualInfoVAR1 *Mutual information analysis of the VAR(1) model*

Description

Evaluate, within the VAR(1) model, the mutual information between each variate at the current time point and those at a future time point.

Usage

```
mutualInfoVAR1(A, SigmaE, T)
```

Arguments

A	Matrix $\mathbf{A}$ of autoregression parameters.
SigmaE	Covariance matrix of the errors (innovations).
T	Positive numeric of length one specifying the future time point with which the mutual informations are to be evaluated.

Value

Object of class `numeric` with elements corresponding to the mutual informations. The j-th element represents the mutual information of the j-th variate at the current time point with all variates at the `T`-th time point from now.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2017), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

See Also

`ridgeVAR1`.

Examples

```
# set dimensions
p <- 3
n <- 4
T <- 10

# set model parameters
SigmaE <- diag(p)/4
A <- matrix(c(-0.1, -0.3,  0.6,
               0.5, -0.4,    0,
               0.3, -0.5, -0.2), byrow=TRUE, ncol=3)

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

# fit VAR(1) model
VAR1hat <- ridgeVAR1(Y, 1, 1)

# impulse response analysis
mutualInfoVAR1(VAR1hat$A, solve(symm(VAR1hat$P)), T=5)
```

`mutualInfoVAR2` *Mutual information analysis of the VAR(2) model*

Description

Evaluate, within the VAR(2) model, the mutual information between a variate at one time point and the variates at a future time point.

Usage

```
mutualInfoVAR2(A1, A2, SigmaE, T)
```

Arguments

A1	A matrix $\mathbf{A}_1$ of lag one autoregression parameters.
A2	A matrix $\mathbf{A}_2$ of lag two autoregression parameters.
SigmaE	Covariance matrix of the errors (innovations).
T	Positive integer of length one specifying the lag between time points for which the mutual informations are to be evaluated.

Value

Object of class numeric with elements corresponding to the mutual informations.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

mutualInfoVAR1, ridgeVAR2.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 12; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A1     <- -createA(p, "clique", nCliques=1, nonzeroA=0.1)
A2     <- t(createA(p, "chain", nBands=1, nonzeroA=0.1))

# generate data
Y <- dataVAR2(n, T, A1, A2, SigmaE)

# fit VAR(1) model
VAR2hat <- ridgeVAR2(Y, 1, 1, 1)

# impulse response analysis
mutualInfoVAR2(VAR2hat$A1, VAR2hat$A2, solve(symm(VAR2hat$P)), 10)
```

`nodeStatsVAR1` *VAR(1) model node statistics*

Description

Function that calculates for each variate various statistics from a sparse VAR(1) model

Usage

```
nodeStatsVAR1(sparseA, sparseP, as.table = FALSE)
```

Arguments

`sparseA`	A `matrix` $\mathbf{A}$ of autoregression parameters, which is assumed to be sparse.
`sparseP`	Precision `matrix` $\mathbf{\Omega}_{\varepsilon}$ the error, which is assumed to be sparse.
`as.table`	A `logical` indicating if the output should be in tabular format.

Details

The function calculates various node statistics from a sparse VAR(1) model. The input matrices $\mathbf{A}$ and $\mathbf{\Omega}_{\varepsilon}$ are assumed to be sparse autoregression coefficient and error precision matrices. From these matrices the global and contemporaneous conditional independence graphs are obtained (Dahlhaus, 2000; Dahlhaus, Eichler, 2003).

For both graph types the function calculates various measures of centrality: node degree, betweenness centrality, closeness centrality, and eigenvalue centrality. It also calculates the number of positive and the number of negative edges for each node. For more information on network measures, consult, e.g., Newman (2010).

In addition, for each variate the mutual information (with all other variates) (Miok et al., 2017), mean impulse response (Hamilton, 1994; Lutkepohl 2005), the (error) variance, and the partial error variance are presented.

Value

An object of class `list` (when `as.table = FALSE`) with slots:

`degreeAin`	A `numeric` vector with the number of (temporal) edges pointing to each node ('in'-degree).
`degreeAout`	A `numeric` vector with the number of (temporal) edges leaving each node ('out'-degree).
`nNegAin`	A `numeric` vector with the number of negative (temporal) edges pointing to each node.
`nPosAin`	A `numeric` vector with the number of positive (temporal) edges pointing to each node ('in'-degree)
`nNegAout`	A `numeric` vector with the number of negative (temporal) edges leaving each node ('out'-degree)

`nPosAout` A `numeric` vector with the number of positive (temporal) edges leaving each node ('out'-degree)

`degreePe` A `numeric` vector with the number of contemporaneous edges of each node (as implied by the error precision matrix)

`betweennessPe` A `numeric` vector representing the contemporaneous betweenness centrality for each node.

`closenessPe` A `numeric` vector representing the contemporaneous closeness centrality for each node.

`eigenCentralityPe` A `numeric` vector representing the contemporaneous eigen centrality for each node.

`nNegPe` A `numeric` vector representing the number of negative contemporaneous edges for each node.

`nPosPe` A `numeric` vector representing the number of positive contemporaneous edges for each node.

`variancePe` A `numeric` vector representing the error variance of each node.

`partialVarPe` A `numeric` vector representing the partial error variance of each node.

`varianceY` A `numeric` vector representing the variance of each node.

`degreePy` A `numeric` number of edges of each node in the global Markov graph.

`betweennessPy` A `numeric` vector representing the betweenness centrality for each node in the global Markov graph.

`closenessPy` A `numeric` vector representing the closeness centrality for each node in the global Markov graph.

`eigenCentralityPy` A `numeric` vector representing the eigen centrality for each node in the global Markov graph.

`mutualInfo_Tplus1` A `numeric` vector with for each node its mutual information with all other nodes at the next (t+1) time point.

`mutualInfo_Tplus2` A `numeric` vector with for each node its mutual information with all other nodes at the (t+2)-th time point.

`itemResponse_Tplus1` A `numeric` vector with for each node its mean absolute impulse response on all other nodes at the next (t+1) time point.

`itemResponse_Tplus2` A `numeric` vector with for each node its mean absolute impulse response on all other nodes at the (t+2)-th time point.

When `as.table = TRUE` the list items above are represented in tabular form as an object of class `matrix`.

Future versions of this function may include additional statistics

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>, Carel F.W. Peeters.

References

Dahlhaus (2000), "Graphical interaction models for multivariate time series", Metrika, 51, 157-172.

Dahlhaus, Eichler (2003), "Causality and graphical models in time series analysis", Oxford Statistical Science Series, 115-137.

Hamilton, J. D. (1994), *Time Series Analysis*. Princeton: Princeton university press.

Lutkepohl, H. (2005), *New Introduction to Multiple Time Series Analysis*. Springer, Berlin.

Newman, M.E.J. (2010). *Networks: An Introduction*, Oxford University Press.

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2017), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

See Also

`ridgeVAR1`, `sparsifyVAR1`, `graphVAR1`

Examples

```
# specify VAR(1) model parameters
A <- matrix(c(-0.1, -0.3,    0,
               0.5,    0,    0,
                 0,    0, -0.4), byrow=TRUE, ncol=3)
P <- matrix(c(   1,  0.5,    0,
               0.5,    1,    0,
                 0,    0,    1), byrow=TRUE, ncol=3)

# adjacency matrix of (global) conditional independencies.
# nodeStatsVAR1(A, P)
```

`nodeStatsVAR2` *VAR(2) model node statistics*

Description

Function that calculates for each variate various statistics from a sparse VAR(2) model

Usage

```
nodeStatsVAR2(sparseA1, sparseA2, sparseP, as.table = FALSE)
```

Arguments

sparseA1	A matrix $\mathbf{A}_1$ of autoregression parameters, which is assumed to be sparse.
sparseA2	A matrix $\mathbf{A}_2$ of autoregression parameters, which is assumed to be sparse.
sparseP	Precision matrix $\mathbf{\Omega}_\varepsilon$ the error, which is assumed to be sparse.
as.table	A logical indicating if the output should be in tabular format.

Details

The function calculates various node statistics from a sparse VAR(2) model. The input matrices $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{\Omega}_\varepsilon$ are assumed to be sparse autoregression coefficient and error precision matrices. From these matrices the global and contemporaneous conditional independence graphs are obtained (Dahlhaus, 2000; Dahlhaus, Eichler, 2003).

For both graph types the function calculates various measures of centrality: node degree, betweenness centrality, closeness centrality, and eigenvalue centrality. It also calculates the number of positive and the number of negative edges for each node. For more information on network measures, consult, e.g., Newman (2010).

In addition, for each variate the mutual information (with all other variates) (Miok et al., 2017), mean impulse response (Hamilton, 1994; Lutkepohl 2005), the (error) variance, and the partial error variance are presented.

Value

An object of class list (when as.table = FALSE) with slots:

degreeA1in	A numeric vector with the number of one-lag (temporal) edges pointing to each node ('in'-degree).
degreeA1out	A numeric vector with the number of one-lag (temporal) edges leaving each node ('out'-degree).
nNegA1in	A numeric vector with the number of negative one-lag (temporal) edges pointing to each node.
nPosA1in	A numeric vector with the number of positive one-lag (temporal) edges pointing to each node ('in'-degree)
nNegA1out	A numeric vector with the number of negative one-lag (temporal) edges leaving each node ('out'-degree)
nPosA1out	A numeric vector with the number of positive one-lag (temporal) edges leaving each node ('out'-degree)
degreeA2in	A numeric vector with the number of two-lag (temporal) edges pointing to each node ('in'-degree).
degreeA2out	A numeric vector with the number of two-lag (temporal) edges leaving each node ('out'-degree).
nNegA2in	A numeric vector with the number of negative two-lag (temporal) edges pointing to each node.
nPosA2in	A numeric vector with the number of positive two-lag (temporal) edges pointing to each node ('in'-degree)

`nNegA2out` A `numeric` vector with the number of negative two-lag (temporal) edges leaving each node ('out'-degree)

`nPosA2out` A `numeric` vector with the number of positive two-lag (temporal) edges leaving each node ('out'-degree)

`degreePe` A `numeric` vector with the number of contemporaneous edges of each node (as implied by the error precision matrix)

`betweennessPe` A `numeric` vector representing the contemporaneous betweenness centrality for each node.

`closenessPe` A `numeric` vector representing the contemporaneous closeness centrality for each node.

`eigenCentralityPe` A `numeric` vector representing the contemporaneous eigen centrality for each node.

`nNegPe` A `numeric` vector representing the number of negative contemporaneous edges for each node.

`nPosPe` A `numeric` vector representing the number of positive contemporaneous edges for each node.

`variancePe` A `numeric` vector representing the error variance of each node.

`partialVarPe` A `numeric` vector representing the partial error variance of each node.

`varianceY` A `numeric` vector representing the variance of each node.

`degreePy` A `numeric` number of edges of each node in the global Markov graph.

`betweennessPy` A `numeric` vector representing the betweenness centrality for each node in the global Markov graph.

`closenessPy` A `numeric` vector representing the closeness centrality for each node in the global Markov graph.

`eigenCentralityPy` A `numeric` vector representing the eigen centrality for each node in the global Markov graph.

`mutualInfo_Tplus1` A `numeric` vector with for each node its mutual information with all other nodes at the next (t+1) time point.

`mutualInfo_Tplus2` A `numeric` vector with for each node its mutual information with all other nodes at the (t+2)-th time point.

`itemResponse_Tplus1` A `numeric` vector with for each node its mean absolute impulse response on all other nodes at the next (t+1) time point.

`itemResponse_Tplus2` A `numeric` vector with for each node its mean absolute impulse response on all other nodes at the (t+2)-th time point.

When `as.table = TRUE` the list items above are represented in tabular form as an object of class `matrix`.

Future versions of this function may include additional statistics

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>, Carel F.W. Peeters.

References

Dahlhaus (2000), "Graphical interaction models for multivariate time series", Metrika, 51, 157-172.

Dahlhaus, Eichler (2003), "Causality and graphical models in time series analysis", Oxford Statistical Science Series, 115-137.

Hamilton, J. D. (1994), *Time series analysis*. Princeton: Princeton university press.

Lutkepohl, H. (2005), *New Introduction to Multiple Time Series Analysis*. Springer, Berlin.

Newman, M.E.J. (2010). *Networks: An Introduction*, Oxford University Press.

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

ridgeVAR2, sparsifyVAR2, graphVAR2

Examples

```
# specify dimension
p <- 3

# specify VAR(1) model parameters
A1 <- -createA(p, "clique", nCliques=1, nonzeroA=0.1)
A2 <- t(createA(p, "chain", nBands=1, nonzeroA=0.1))
P <- matrix(c(  1, 0.5,  0,
              0.5,   1,  0,
                0,   0,  1), byrow=TRUE, ncol=3)

# adjacency matrix of (global) conditional independencies.
# nodeStatsVAR2(A1, A2, P)
```

---

optPenaltyVAR1 *Automatic penalty parameter selection for the VAR(1) model.*

Description

Automatic penalty parameter selection for the VAR(1) model through maximization of the leave-one-out cross-validated (LOOCV) log-likelihood.

Usage

```
optPenaltyVAR1(Y, lambdaMin, lambdaMax,
               lambdaInit=(lambdaMin+lambdaMax)/2,
               optimizer="nlm", ...)
```

Arguments

Y	Three-dimensional array containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to centered covariate-wise.
lambdaMin	A numeric of length two, containing the minimum values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}$, the matrix with autoregression coefficients, while the second parameter relates to the penalty on ($\mathbf{\Omega}_\varepsilon$, the precision matrix of the errors.
lambdaMax	A numeric of length two, containing the maximum values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}$, the matrix with autoregression coefficients, while the second parameter relates to the penalty on ($\mathbf{\Omega}_\varepsilon$, the precision matrix of the errors.
lambdaInit	A numeric of length two, containing the initial values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}$, the matrix with autoregression coefficients, while the second parameter relates to the penalty on ($\mathbf{\Omega}_\varepsilon$, the precision matrix of the errors.
optimizer	A character (either nlm (default) or optim) specifying which optimization function should be used: nlminb (default) or constrOptim?
...	Additional arguments passed on to loglikLOOCVVAR1.

Value

A numeric with the LOOCV optimal choice for the ridge penalty parameter.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2017), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

See Also

loglikLOOCVVAR1, ridgeVAR1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- createA(3, "chain")
```

```
# generate data
Y <- dataVAR1(n, T, A, SigmaE)

# determine the optimal penalty parameter
optLambda <- optPenaltyVAR1(Y, rep(10^(-10), 2), rep(1000, 2))

# fit VAR(1) model
ridgeVAR1(Y, optLambda[1], optLambda[2])$A
```

`optPenaltyVAR1fused` *Automatic penalty parameter selection for multiple VAR(1) models.*

Description

Automatic penalty parameter selection for multiple VAR(1) models through maximization of the leave-one-out cross-validated (LOOCV) log-likelihood.

Usage

```
optPenaltyVAR1fused(Y, id, lambdaMin, lambdaMax,
                    lambdaInit=(lambdaMin+lambdaMax)/2,
                    optimizer="nlm", ...)
```

Arguments

`Y`	Three-dimensional `array` containing the response data. The first, second and third dimensions correspond to variates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`id`	A vector with group indices comprising of integers only. First group is represented by '0', the next by '1', and so on until the last.
`lambdaMin`	A `numeric` of length three, containing the minimum values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}_g$'s, the matrices with lag one autoregression coefficients, the second to the fused ridge parameter for these $\mathbf{A}_g$'s, while the third parameter relates to the penalty on $\boldsymbol{\Omega}_\varepsilon$, the precision matrix of the errors.
`lambdaMax`	A `numeric` of length three, containing the maximum values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}_g$'s, the matrices with lag one autoregression coefficients, the second to the fused ridge parameter for these $\mathbf{A}_g$'s, while the third parameter relates to the penalty on $\boldsymbol{\Omega}_\varepsilon$, the precision matrix of the errors.
`lambdaInit`	A `numeric` of length three, containing the initial values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}_g$'s, the matrices with lag one autoregression coefficients, the second to the fused ridge parameter for these $\mathbf{A}_g$'s, while the third parameter relates to the penalty on $\boldsymbol{\Omega}_\varepsilon$, the precision matrix of the errors.

optimizer	A character (either nlm (default) or optim) specifying which optimization function should be used: nlminb (default) or constrOptim?
...	Additional arguments passed on to loglikLOOCVVAR1fused.

Value

A numeric with the LOOCV optimal choice for the ridge penalty parameter.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

loglikLOOCVVAR1fused, ridgeVAR1fused.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points, G=groups)
p <- 3; n <- 12; T <- 10; G <- 3

# set model parameters
SigmaE       <- matrix(1/2, p, p)
diag(SigmaE) <- 1
A1           <- -createA(p, "clique", nCliques=1, nonzeroA=0.1)
A2           <- t(createA(p, "chain", nBands=1, nonzeroA=0.1))
A3           <- (A1 + A2) / 2

# generate data
Y1 <- dataVAR1(n/G, T, A1, SigmaE)
Y2 <- dataVAR1(n/G, T, A2, SigmaE)
Y3 <- dataVAR1(n/G, T, A3, SigmaE)
Y  <- abind::abind(Y1, Y2, Y3, along=3)
id <- c(rep(1, n/G), rep(2, n/G), rep(3, n/G))-1

# determine the optimal penalty parameter
## Not run: optLambdas <- optPenaltyVAR1fused(Y, rep(10^(-10), 3),
## Not run:                rep(1000, 3), optimizer="nlm")

# ridge ML estimation of the VAR(1) parameter estimates with
# optimal penalty parameters
optLambdas <- c(0.1, 0.1, 0.1)
VAR1hats <- ridgeVAR1fused(Y, id, optLambdas[1], optLambdas[2], optLambdas[3])
```

optPenaltyVAR2 *Automatic penalty parameter selection for the VAR(2) model.*

Description

Automatic penalty parameter selection for the VAR(2) model through maximization of the leave-one-out cross-validated (LOOCV) log-likelihood.

Usage

```
optPenaltyVAR2(Y, lambdaMin, lambdaMax,
               lambdaInit=(lambdaMin+lambdaMax)/2,
               optimizer="nlm", ...)
```

Arguments

Y	Three-dimensional array containing the response data. The first, second and third dimensions correspond to variates, time and samples, respectively. The data are assumed to be centered covariate-wise.
lambdaMin	A numeric of length three, containing the minimum values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}_1$, the matrix with lag one autoregression coefficients, the second to matrix $\mathbf{A}_2$ containing the lag two autoregression coefficients, while the third parameter relates to the penalty on $\mathbf{\Omega}_\varepsilon$, the precision matrix of the errors.
lambdaMax	A numeric of length three, containing the maximum values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}_1$, the matrix with lag one autoregression coefficients, the second to matrix $\mathbf{A}_2$ containing the lag two autoregression coefficients, while the third parameter relates to the penalty on $\mathbf{\Omega}_\varepsilon$, the precision matrix of the errors.
lambdaInit	A numeric of length three, containing the initial values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}_1$, the matrix with lag one autoregression coefficients, the second to matrix $\mathbf{A}_2$ containing the lag two autoregression coefficients, while the third parameter relates to the penalty on $\mathbf{\Omega}_\varepsilon$, the precision matrix of the errors.
optimizer	A character (either nlm (default) or optim) specifying which optimization function should be used: nlminb (default) or constrOptim?
...	Additional arguments passed on to loglikLOOCVVAR2.

Value

A numeric with the LOOCV optimal choice for the ridge penalty parameter.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

loglikLOOCVVAR2, ridgeVAR2.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A1 <- createA(p, topology="clique", nonzeroA=0.1, nClique=1)
A2 <- createA(p, topology="hub", nonzeroA=0.1, nHubs=1)

# generate data
Y <- dataVAR2(n, T, A1, A2, SigmaE)

# determine the optimal penalty parameter
optLambda <- optPenaltyVAR2(Y, rep(10^(-10), 3), rep(1000, 3),
                            optimizer="nlm")

# fit VAR(2) model
ridgeVAR2(Y, optLambda[1], optLambda[2], optLambda[3])
```

---

optPenaltyVARX1 *Automatic penalty parameter selection for the VARX(1) model.*

Description

Automatic penalty parameter selection for the VARX(1) model through maximization of the leave-one-out cross-validated (LOOCV) log-likelihood.

Usage

```
optPenaltyVARX1(Y, X, lambdaMin, lambdaMax,
                lambdaInit=(lambdaMin+lambdaMax)/2,
                optimizer="nlm", ...)
```

Arguments

Y	Three-dimensional `array` containing the response data. The first, second and third dimensions correspond to variates, time and samples, respectively. The data are assumed to be centered covariate-wise.

X	Three-dimensional array containing the time-varying covariate data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
lambdaMin	A numeric of length three, containing the minimum values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}$, the matrix with autoregression coefficients, the second to matrix $\mathbf{B}$ containing the regression coefficients of the time-varying covariates, while the third parameter relates to the penalty on $\mathbf{\Omega}_{\varepsilon}$, the precision matrix of the errors.
lambdaMax	A numeric of length three, containing the maximum values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}$, the matrix with autoregression coefficients, the second to matrix $\mathbf{B}$ containing the regression coefficients of the time-varying covariates, while the third parameter relates to the penalty on $\mathbf{\Omega}_{\varepsilon}$, the precision matrix of the errors.
lambdaInit	A numeric of length three, containing the initial values of ridge penalty parameters to be considered. The first element is the ridge parameter corresponding to the penalty on $\mathbf{A}$, the matrix with autoregression coefficients, the second to matrix $\mathbf{B}$ containing the regression coefficients of the time-varying covariates, while the third parameter relates to the penalty on $\mathbf{\Omega}_{\varepsilon}$, the precision matrix of the errors.
optimizer	A character (either nlm (default) or optim) specifying which optimization function should be used: nlminb (default) or constrOptim?
...	Additional arguments passed on to the loglikLOOCVVARX1-function.

Value

A numeric with the LOOCV optimal choice for the ridge penalty parameter.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

loglikLOOCVVARX1, ridgeVARX1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
```

```
SigmaE <- diag(p)/4
Ax     <- createA(3, "chain")

# generate time-varying covariate data
X <- dataVAR1(n, T, Ax, SigmaE)

# (auto)regression parameter matrices of VARX(1) model
A <- createA(p, topology="clique", nonzeroA=0.1, nClique=1)
B <- createA(p, topology="hub", nonzeroA=0.1, nHubs=1)

# generate data
Y <- dataVARX1(X, A, B, SigmaE, lagX=0)

# determine the optimal penalty parameter
optLambda <- optPenaltyVARX1(Y, X, rep(10^(-10), 3), rep(1000, 3),
                             optimizer="nlm", lagX=0)

# fit VAR(1) model
ridgeVARX1(Y, X, optLambda[1], optLambda[2], optLambda[3], lagX=0)$A
```

`plotVAR1data` *Time series plot*

Description

Plot of time series data. Per variate and individual a line connecting the observations at each time point is plotted.

Usage

```
plotVAR1data(Y, lwd=1)
```

Arguments

`Y`	Three-dimensional `array` containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be zero centered covariate-wise, per sample.
`lwd`	A `numeric` of length one, specifying the line width.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

See Also

`dataVAR1`.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- createA(p, "chain")

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

# plot data sampled from the VAR(1) model.
plotVAR1data(Y, lwd=2)
```

---

pruneMotifStats *Network motif list subsetting.*

Description

Limit a list of network motifs as returned by the `motifStatsVAR1`-function to those involving a particular node.

Usage

```
pruneMotifStats(motifList, id)
```

Arguments

`motifList`	A `list` with motifs as returned by the `motifStatsVAR1`-function.
`id`	A `integer` representing the node of interest.

Value

An object of class `list` with slots:

`selfregulators`	A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.
`feedbackpairs`	A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.
`feedforwardloops`	A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.
`feedbackloops`	A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.
`bifans`	A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.

`diamonds` A `list` of motifs specified as matrices with each row one of the motif's paths and in the last column the sign of the path's contribution.

Future versions of this function may include additional slots reporting more motif types.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>.

References

Alon, U. (2007), "Network motifs: theory and experimental approaches", Nature Reviews Genetics, 8, 450-461.

See Also

graphVAR1, motifStatsVAR1

Examples

```
# specify lag one autoregression model
sparseA <- matrix(runif(2500), ncol=50)
sparseA[sparseA < 0.9] <- 0

# find motifs
motifList <- motifStatsVAR1(sparseA)

# prune motif list
pruneMotifStats(motifList, 1)
```

`ridgePathVAR1` *Visualize the ridge regularization paths of the parameters of the VAR(1) model*

Description

Function that visualizes the regularization paths of the parameters of the VAR(1) model. The elements of the ridge ML estimate of either $\mathbf{A}$ or (possibly standardized, inverse of) $\mathbf{\Sigma}_\varepsilon$ are plotted against a specified range of their penalty parameter (keeping the other penalty parameter fixed).

Usage

```
ridgePathVAR1(Y, lambdaAgrid, lambdaPgrid, pathType="A",
              plotTypeSigmaE="pcor", diag=FALSE, verbose=TRUE, ...)
```

Arguments

Y	Three-dimensional array containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
lambdaAgrid	A numeric of length larger than one, comprising positive numbers only. It contains the grid points corresponding to the λ_a (the penalty parameter for the autoregression coefficient matrix $\mathbf{A}$).
lambdaPgrid	A numeric of length larger than one, comprising positive numbers only. It contains the grid points corresponding to the λ_ω (the penalty parameters for the inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$).
pathType	A character indicating of which parameter to plot its ridge regularization paths. Either "A" or "SigmaE".
plotTypeSigmaE	A character indicating the type of element for which a visualization of the regularization paths (of $\mathbf{\Sigma}_\varepsilon$) is desired. Must be one of: "pcor", "cor", "cov", "prec".
diag	A logical indicating if the diagonal elements should be retained for visualization of the regularization path of $\mathbf{\Sigma}_\varepsilon$.
verbose	A logical indicator: should intermediate output be printed on the screen?
...	Other arguments to be passed to ridgeVAR1.

Details

If pathType="A", the regularization paths of $\mathbf{A}$ will be evaluated for λ_ω equal to the minimum value of lambdaPgrid. If pathType="SigmaE", the regularization paths of (inverse of / possibly standardarized) $\mathbf{\Sigma}_\varepsilon$ will be evaluated for λ_a equal to the minimum value of lambdaAgrid.

Regularization paths may be visualized for (partial) correlations, covariances and precision elements. The type of element for which a visualization of the regularization paths is desired can be indicated by the argument plotType.

The arguments diag and plotTypeSigmaE are ignored when pathType="A".

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>.

See Also

ridgePathS, ridgeP, ridgeVAR1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- createA(p, "chain")
```

```
# generate data
Y <- dataVAR1(n, T, A, SigmaE)

## Visualize regularization paths
lambdaAgrid <- seq(0.01, 1, length.out=20)
lambdaPgrid <- seq(0.01, 1, length.out=20)
ridgePathVAR1(Y, lambdaAgrid, lambdaPgrid, pathType="A")
```

---

ridgeVAR1 *Ridge ML estimation of the VAR(1) model*

Description

Ridge penalized maximum likelihood estimation of the parameters of the VAR(1), first-order vector auto-regressive, model. The VAR(1) model explains the current vector of observations $\mathbf{Y}_{*,t+1}$ by a linear combination of the previous observation vector: $\mathbf{Y}_{*,t+1} = \mathbf{A}\mathbf{Y}_{*,t} + \varepsilon_{*,t+1}$, where $\mathbf{A}$ is the autoregression coefficient matrix and $\varepsilon_{*,t+1}$ the vector of errors (or innovations). The VAR(1)-process is assumed to have mean zero. The experimental design is allowed to be unbalanced.

Usage

```
ridgeVAR1(Y, lambdaA=0, lambdaP=0,
          targetA=matrix(0, dim(Y)[1], dim(Y)[1]),
          targetP=matrix(0, dim(Y)[1], dim(Y)[1]), targetPtype="none",
          fitA="ml", zerosA=matrix(nrow=0, ncol=2),
          zerosAfit="sparse", zerosP=matrix(nrow=0, ncol=2),
          cliquesP=list(), separatorsP=list(),
          unbalanced=matrix(nrow=0, ncol=2), diagP=FALSE,
          efficient=TRUE, nInit=100, minSuccDiff=0.001)
```

Arguments

Y	Three-dimensional `array` containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
lambdaA	Ridge penalty parameter (positive `numeric` of length 1) to be used in the estimation of $\mathbf{A}$, the matrix with autoregression coefficients.
lambdaP	Ridge penalty parameter (positive `numeric` of length 1) to be used in the estimation of the inverse error covariance matrix ($\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$): the precision matrix of the errors.
targetA	Target `matrix` to which the matrix $\mathbf{A}$ is to be shrunken.
targetP	Target `matrix` to which the in the inverse error covariance matrix, the precision matrix, is to be shrunken.
fitA	A `character`. If `fitA="ml"` the parameter $\mathbf{A}$ is estimate by (penalized) maximum likelihood. If `fitA="ss"` the parameter $\mathbf{A}$ is estimate by (penalized) sum of squares. The latter being much faster as it need not iterate.

`targetPtype` A `character` indicating the type of target to be used for the precision matrix. When specified it overrules the `targetP`-option. See the `default.target`-function for the options.

`zerosA` A `matrix` with indices of entries of $\mathbf{A}$ that are constrained to zero. The matrix comprises two columns, each row corresponding to an entry of $\mathbf{A}$. The first column contains the row indices and the second the column indices.

`zerosAfit` A `character`, either "sparse" or "dense". With "sparse", the matrix $\mathbf{A}$ is assumed to contain many zeros and a computational efficient implementation of its estimation is employed. If "dense", it is assumed that $\mathbf{A}$ contains only few zeros and the estimation method is optimized computationally accordingly.

`zerosP` A `matrix`-object with indices of entries of the precision matrix that are constrained to zero. The matrix comprises two columns, each row corresponding to an entry of the adjacency matrix. The first column contains the row indices and the second the column indices. The specified graph should be undirected and decomposable. If not, it is symmetrized and triangulated (unless `cliquesP` and `seperatorsP` are supplied). Hence, the employed zero structure may differ from the input `zerosP`.

`cliquesP` A `list`-object containing the node indices per clique as object from the `rip`-function.

`separatorsP` A `list`-object containing the node indices per clique as object from the `rip`-function.

`unbalanced` A `matrix` with two columns, indicating the unbalances in the design. Each row represents a missing design point in the (time x individual)-layout. The first and second column indicate the time and individual (respectively) specifics of the missing design point.

`diagP` A `logical`, indicates whether the inverse error covariance matrix is assumed to be diagonal.

`efficient` A `logical`, affects estimation of $\mathbf{A}$. Details below.

`nInit` Maximum number of iterations (positive `numeric` of length 1) to be used in maximum likelihood estimation.

`minSuccDiff` Minimum distance (positive `numeric` of length 1) between estimates of two successive iterations to be achieved.

Details

The ridge ML estimator employs the following estimator of the variance of the VAR(1) process:

$$\frac{1}{n(\mathcal{T}-1)}\sum_{i=1}^{n}\sum_{t=2}^{\mathcal{T}}\mathbf{Y}_{*,i,t}\mathbf{Y}_{*,i,t}^{\mathrm{T}}.$$

This is used when `efficient=FALSE`. However, a more efficient estimator of this variance can be used

$$\frac{1}{n\mathcal{T}}\sum_{i=1}^{n}\sum_{t=1}^{\mathcal{T}}\mathbf{Y}_{*,i,t}\mathbf{Y}_{*,i,t}^{\mathrm{T}},$$

which is achieved by setting when `efficient=TRUE`. Both estimators are adjusted accordingly when dealing with an unbalanced design.

Value

A list-object with slots:

A	Ridge ML estimate of the matrix $\mathbf{A}$, the matrix with lag one auto-regressive coefficients.
P	Ridge ML estimate of the inverse error covariance matrix $\mathbf{\Omega}_{\varepsilon}(=\mathbf{\Sigma}_{\varepsilon}^{-1})$.
lambdaA	Positive numeric of length one: ridge penalty used in the estimation of $\mathbf{A}$.
lambdaP	Positive numeric of length one: ridge penalty used in the estimation of inverse error covariance matrix $\mathbf{\Omega}_{\varepsilon}(=\mathbf{\Sigma}_{\varepsilon}^{-1})$.

Note

When the target of the precision matrix is specified through the targetPtype-argument, the target is data-driven (for both fitA="ss" and fitA="ml"). In particular, it is updated at each iteration when fitA="ml".

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2017), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

See Also

loglikLOOCVVAR1, ridgeP, default.target, ridgePchordal.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A      <- createA(p, "chain")

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

# fit VAR(1) model
ridgeVAR1(Y, 1, 1)$A
```

`ridgeVAR1fused` *Fused ridge ML estimation of multiple VAR(1) model*

Description

Ridge penalized maximum likelihood estimation of the parameters of the first-order Vector Auto-Regressive model, with a (possibly) unbalanced experimental set-up. The VAR(1)-process is assumed to have mean zero.

Usage

```
ridgeVAR1fused(Y, id, lambdaA=0, lambdaF=0, lambdaP=0,
               targetA=matrix(0, dim(Y)[1], dim(Y)[1]),
               targetP=matrix(0, dim(Y)[1], dim(Y)[1]),
               targetPtype="none", fitA="ml",
               zerosA=matrix(nrow=0, ncol=2), zerosAfit="sparse",
               zerosP=matrix(nrow=0, ncol=2), cliquesP=list(),
               separatorsP=list(), unbalanced=matrix(nrow=0, ncol=2),
               diagP=FALSE, efficient=TRUE, nInit=100, nInitA=5,
               minSuccDiff=0.001, minSuccDiffA=0.001)
```

Arguments

`Y`	Three-dimensional `array` containing the data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`id`	A vector with group indices comprising of integers only. First group is represented by '0', the next by '1', and so on until the last.
`lambdaA`	Ridge penalty parameter (positive `numeric` of length 1) to be used in the estimation of the $\mathbf{A}_g$, the matrices with autoregression coefficients.
`lambdaF`	Fused ridge penalty parameter (positive `numeric` of length 1) to be used in the estimation of $\mathbf{A}_g$, the matrices with autoregression coefficients.
`lambdaP`	Ridge penalty parameter (positive `numeric` of length 1) to be used in the estimation of the inverse error covariance matrix ($\boldsymbol{\Omega}_\varepsilon (= \boldsymbol{\Sigma}_\varepsilon^{-\mathbf{1}})$): the precision matrix of the errors.
`targetA`	Target `matrix` to which the matrix $\mathbf{A}$ is to be shrunken. This target is shared among the groups (otherwise why fuse?).
`targetP`	Target `matrix` to which the in the inverse error covariance matrix, the precision matrix, is to be shrunken.
`fitA`	A `character`. If `fitA="ml"` the parameters $\mathbf{A}_g$ ared estimated by (penalized) maximum likelihood. If `fitA="ss"` the parameter $\mathbf{A}$ is estimate by (penalized) sum of squares. The latter is much faster.
`targetPtype`	A `character` indicating the type of target to be used for the precision matrix. When specified it overrules the `targetP`-option. See the `default.target`-function for the options.

`zerosA` A `matrix` with indices of entries of $\mathbf{A}_g$'s that are constrained to zero. The matrix comprises two columns, each row corresponding to an entry of the $\mathbf{A}_s$. The first column contains the row indices and the second the column indices. The support is shared among the groups (otherwise why fuse?).

`zerosAfit` A `character`, either "sparse" or "dense". With "sparse", the matrix $\mathbf{A}$ is assumed to contain many zeros and a computational efficient implementation of its estimation is employed. If "dense", it is assumed that $\mathbf{A}_g$'s contain only few zeros and the estimation method is optimized computationally accordingly.

`zerosP` A `matrix`-object with indices of entries of the precision matrix that are constrained to zero. The matrix comprises two columns, each row corresponding to an entry of the adjacency matrix. The first column contains the row indices and the second the column indices. The specified graph should be undirected and decomposable. If not, it is symmetrized and triangulated (unless `cliquesP` and `seperatorsP` are supplied). Hence, the employed zero structure may differ from the input `zerosP`.

`cliquesP` A `list`-object containing the node indices per clique as object from the `rip`-function.

`separatorsP` A `list`-object containing the node indices per clique as object from the `rip`-function.

`unbalanced` A `matrix` with two columns, indicating the unbalances in the design. Each row represents a missing design point in the (time x individual)-layout. The first and second column indicate the time and individual (respectively) specifics of the missing design point.

`diagP` A `logical`, indicates whether the inverse error covariance matrix is assumed to be diagonal.

`efficient` A `logical`, affects estimation of the $\mathbf{A}_g$. Details below.

`nInit` Maximum number of iterations (positive `numeric` of length 1) to be used in maximum likelihood estimation.

`nInitA` Maximum number of iterations (positive `numeric` of length 1) to be used in fused estimation of the autoregression matrices $\mathbf{A}_g$, given the current estimate of $\mathbf{\Omega}_\varepsilon$.

`minSuccDiff` Minimum distance (positive `numeric` of length 1) between estimates of two successive iterations to be achieved.

`minSuccDiffA` Minimum distance (positive `numeric` of length 1) between the $\mathbf{A}_g$ estimates of two successive fused estimation iterations to be achieved.

Details

If `diagP=TRUE`, no penalization to estimation of the covariance matrix is applied. Consequently, the arguments `lambdaP` and `targetP` are ignored (if supplied).

The ridge ML estimator employs the following estimator of the variance of the VAR(1) process:

$$\frac{1}{n(\mathcal{T}-1)}\sum_{i=1}^{n}\sum_{t=2}^{\mathcal{T}}\mathbf{Y}_{*,i,t}\mathbf{Y}_{*,i,t}^{\mathrm{T}}.$$

This is used when `efficient=FALSE`. However, a more efficient estimator of this variance can be used

$$\frac{1}{n\mathcal{T}}\sum_{i=1}^{n}\sum_{t=1}^{\mathcal{T}}\mathbf{Y}_{*,i,t}\mathbf{Y}_{*,i,t}^{\mathrm{T}},$$

which is achieved by setting when `efficient=TRUE`. Both estimators are adjusted accordingly when dealing with an unbalanced design.

Value

A list-object with slots:

`As`	Ridge ML estimates of the matrices $\mathbf{A}_g$, stacked and stored as a single rectangular `matrices`.
`P`	Ridge ML estimate of the inverse error covariance `matrix` $\boldsymbol{\Omega}_{\varepsilon}(=\boldsymbol{\Sigma}_{\varepsilon}^{-1})$.
`lambdaA`	Positive `numeric` of length one: ridge penalty used in the estimation of the $\mathbf{A}_g$.
`lambdaF`	Positive `numeric` of length one: fused ridge penalty used in the estimation of the $\mathbf{A}_g$.
`lambdaP`	Positive `numeric` of length one: ridge penalty used in the estimation of inverse error covariance matrix $\boldsymbol{\Omega}_{\varepsilon}(=\boldsymbol{\Sigma}_{\varepsilon}^{-1})$.

Note

When the target of the precision matrix is specified through the `targetPtype`-argument, the target is data-driven and updated at each iteration when `fitA="ml"`.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

`loglikLOOCVVAR1`, `ridgeVAR1`, `ridgePchordal`.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points, G=groups)
p <- 3; n <- 12; T <- 10; G <- 3

# set model parameters
SigmaE <- matrix(1/2, p, p)
diag(SigmaE) <- 1
A1 <- -createA(p, "clique", nCliques=1, nonzeroA=0.1)
A2 <- t(createA(p, "chain", nBands=1, nonzeroA=0.1))
A3 <- (A1 + A2) / 2
```

```
# generate data
Y1 <- dataVAR1(n/G, T, A1, SigmaE)
Y2 <- dataVAR1(n/G, T, A2, SigmaE)
Y3 <- dataVAR1(n/G, T, A3, SigmaE)
Y  <- abind::abind(Y1, Y2, Y3, along=3)
id <- c(rep(1, n/G), rep(2, n/G), rep(3, n/G))-1

VAR1hats <- ridgeVAR1fused(Y, id, lambdaA=1, lambdaF=1, lambdaP=1)
```

`ridgeVAR2` *Ridge ML estimation of the VAR(2) model*

Description

Ridge penalized maximum likelihood estimation of the parameters of the VAR(2), second-order vector auto-regressive, model. The VAR(2) model explains the current vector of observations $\mathbf{Y}_{*,t+2}$ by a linear combination of the previous two observation vectors: $\mathbf{Y}_{*,t+1} = \mathbf{A}_1\mathbf{Y}_{*,t+1} + \mathbf{A}_2\mathbf{Y}_{*,t} + \varepsilon_{*,t+2}$, where $\mathbf{A}_1$ and $\mathbf{A}_2$ are the lag one and two autoregression coefficient matrices, respectively, and $\varepsilon_{*,t+2}$ the vector of errors (or innovations). The VAR(2)-process is assumed to have mean zero. The experimental design is allowed to be unbalanced.

Usage

```
ridgeVAR2(Y, lambdaA1=-1, lambdaA2=-1, lambdaP=-1,
          targetA1=matrix(0, dim(Y)[1], dim(Y)[1]),
          targetA2=matrix(0, dim(Y)[1], dim(Y)[1]),
          targetP=matrix(0, dim(Y)[1], dim(Y)[1]),
          targetPtype="none", fitA12="ml",
          zerosA1=matrix(nrow=0, ncol=2),
          zerosA2=matrix(nrow=0, ncol=2), zerosA1fit="sparse",
          zerosA2fit="sparse", zerosP=matrix(nrow=0, ncol=2),
          cliquesP=list(), separatorsP=list(),
          unbalanced=matrix(nrow=0, ncol=2), diagP=FALSE,
          efficient=TRUE, nInit=100, minSuccDiff=0.001)
```

Arguments

`Y`	Three-dimensional `array` containing the response data. The first, second and third dimensions correspond to variates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`lambdaA1`	Ridge penalty parameter (positive `numeric` of length 1) to be used in the estimation of $\mathbf{A}_1$, the matrix with the lag one autoregression coefficients.
`lambdaA2`	Ridge penalty parameter (positive `numeric` of length 1) to be used in the estimation of $\mathbf{A}_2$, the matrix with the lag two autoregression coefficients.
`lambdaP`	Ridge penalty parameter (positive `numeric` of length 1) to be used in the estimation of the inverse error covariance matrix ($\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$): the precision matrix of the errors.

targetA1	Target matrix to which the matrix $\mathbf{A}_1$ is to be shrunken.
targetA2	Target matrix to which the matrix $\mathbf{A}_2$ is to be shrunken.
targetP	Target matrix to which the in the inverse error covariance matrix, the precision matrix, is to be shrunken.
fitA12	A character. If fitAB="ml" the parameters $\mathbf{A}_1$ and $\mathbf{A}_2$ are estimated by (penalized) maximum likelihood. If fitAB="ss" the parameters $\mathbf{A}_1$ and $\mathbf{A}_2$ are estimated by (penalized) sum of squares.
targetPtype	A character indicating the type of target to be used for the precision matrix. When specified it overrules the targetP-option. See the default.target-function for the options.
zerosA1	A matrix with indices of entries of $\mathbf{A}_1$ that are constrained to zero. The matrix comprises two columns, each row corresponding to an entry of $\mathbf{A}_1$. The first column contains the row indices and the second the column indices.
zerosA2	A matrix with indices of entries of $\mathbf{A}_2$ that are constrained to zero. The matrix comprises two columns, each row corresponding to an entry of $\mathbf{A}_2$. The first column contains the row indices and the second the column indices.
zerosA1fit	A character, either "sparse" or "dense". With "sparse", the matrix $\mathbf{A}_1$ is assumed to contain many zeros and a computational efficient implementation of its estimation is employed. If "dense", it is assumed that $\mathbf{A}_1$ contains only few zeros and the estimation method is optimized computationally accordingly.
zerosA2fit	A character, either "sparse" or "dense". With "sparse", the matrix $\mathbf{A}_2$ is assumed to contain many zeros and a computational efficient implementation of its estimation is employed. If "dense", it is assumed that $\mathbf{A}_2$ contain only few zeros and the estimation method is optimized computationally accordingly.
zerosP	A matrix-object with indices of entries of the precision matrix that are constrained to zero. The matrix comprises two columns, each row corresponding to an entry of the adjacency matrix. The first column contains the row indices and the second the column indices. The specified graph should be undirected and decomposable. If not, it is symmetrized and triangulated (unless cliquesP and seperatorsP are supplied). Hence, the employed zero structure may differ from the input zerosP.
cliquesP	A list-object containing the node indices per clique as object from the rip-function.
separatorsP	A list-object containing the node indices per clique as object from the rip-function.
unbalanced	A matrix with two columns, indicating the unbalances in the design. Each row represents a missing design point in the (time x individual)-layout. The first and second column indicate the time and individual (respectively) specifics of the missing design point.
diagP	Logical, indicates whether the inverse error covariance matrix is assumed to be diagonal.
efficient	Logical, affects estimation of $\mathbf{A}_1$ and $\mathbf{A}_2$ directly. Details below.
nInit	Maximum number of iterations (positive numeric of length 1) to be used in maximum likelihood estimation.
minSuccDiff	Minimum distance (positive numeric of length 1) between estimates of two successive iterations to be achieved.

Details

If `diagP=TRUE`, no penalization to estimation of the covariance matrix is applied. Consequently, the arguments `lambdaP` and `targetP` are ignored (if supplied).

The ridge ML estimator employs the following estimator of the variance of the VARX(1) process:

$$\frac{1}{n(\mathcal{T}-1)}\sum_{i=1}^{n}\sum_{t=2}^{\mathcal{T}}\mathbf{Y}_{*,i,t}\mathbf{Y}_{*,i,t}^{\top}.$$

This is used when `efficient=FALSE`. However, a more efficient estimator of this variance can be used

$$\frac{1}{n\mathcal{T}}\sum_{i=1}^{n}\sum_{t=1}^{\mathcal{T}}\mathbf{Y}_{*,i,t}\mathbf{Y}_{*,i,t}^{\top},$$

which is achieved by setting when `efficient=TRUE`. Both estimators are adjusted accordingly when dealing with an unbalanced design.

Value

A list-object with slots:

`A1`	Ridge ML estimate of the matrix $\mathbf{A}_1$, the `matrix` with lag one auto-regressive coefficients.
`A2`	Ridge ML estimate of the matrix $\mathbf{A}_2$, the `matrix` with lag two auto-regressive coefficients.
`P`	Ridge ML estimate of the inverse error covariance `matrix` $\boldsymbol{\Omega}_{\varepsilon}(=\boldsymbol{\Sigma}_{\varepsilon}^{-\mathbf{1}})$.
`lambdaA1`	Positive `numeric` of length one: ridge penalty used in the estimation of $\mathbf{A}_1$.
`lambdaA2`	Positive `numeric` of length one: ridge penalty used in the estimation of $\mathbf{A}_2$.
`lambdaP`	Positive `numeric` of length one: ridge penalty used in the estimation of inverse error covariance matrix $\boldsymbol{\Omega}_{\varepsilon}(=\boldsymbol{\Sigma}_{\varepsilon}^{-\mathbf{1}})$.

Note

When the target of the precision matrix is specified through the `targetPtype`-argument, the target is data-driven and updated at each iteration when `fitAB="ml"`.

In case $\lambda_{a1} \neq \lambda_{a2}$, the ridge ML estimates (conditional on the current estimate of $\boldsymbol{\Omega}_{\varepsilon}$) of $\mathbf{A}_1$ and $\mathbf{A}_2$ are approximations. Their explicit evaluation involves a Kronecker product times a vector. Its full expansion is memory inefficient, in particular for (say) $p_y > 50$. If $\lambda_{a1} = \lambda_{a2}$ the estimates of $\mathbf{A}_1$ and $\mathbf{A}_2$ can be evaluated through multiplications of matrices of dimensions $2p_y \times 2p_y$ (instead of $2p_y^2 \times 2p_y^2$). The evaluation cannot be simplified computationally when $\lambda_{a1} \neq \lambda_{a1}$. To avoid the use of prohitedly large matrices an approximation is used. For the approximation it is assumed that the contemporaneous covariance between $\mathbf{Y}_t$ and $\mathbf{X}_t$ is zero (for lag one), after which the estimates can be evaluated using matrices of order $p_y + p_x$.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

default.target, loglikLOOCVVAR1, ridgeP, ridgePchordal.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A1     <- createA(p, "clique", nCliques=1)
A2     <- createA(p, "hub", nHubs=1)

# generate time-varying covariates in accordance with VAR(2) process
Y <- dataVAR2(n, T, A1, A2, SigmaE)

# fit VAR(2) model
ridgeVAR2(Y, 1, 1, 1)
```

---

ridgeVARX1 *Ridge ML estimation of the VARX(1) model*

Description

Ridge penalized maximum likelihood estimation of the parameters of the first-order vector autoregressive model with time-varying covariates, in shorthand VARX(1) model. The VARX(1) model explains the current vector of observations $\mathbf{Y}_{*,t+1}$ by a linear combination of the previous observation endogeneous vector and an exogeneous time-varying covariate: $\mathbf{Y}_{*,t+1} = \mathbf{A}\mathbf{Y}_{*,t} + \mathbf{B}\mathbf{X}_{*,t+1} + \varepsilon_{*,t+1}$, where $\mathbf{A}$ and $\mathbf{B}$ are the lag one autoregression and time-varying regression coefficient matrix, respectively, and $\varepsilon_{*,t+2}$ the vector of errors (or innovations). The VARX(1)-process is assumed to have mean zero. The experimental design is allowed to be unbalanced.

Usage

```
ridgeVARX1(Y, X, lambdaA=-1, lambdaB=-1, lambdaP=-1, lagX,
           targetA=matrix(0, dim(Y)[1], dim(Y)[1]),
           targetB=matrix(0, dim(Y)[1], dim(X)[1]),
           targetP=matrix(0, dim(Y)[1], dim(Y)[1]), targetPtype="none",
           fitAB="ml", zerosA=matrix(nrow=0, ncol=2),
           zerosB=matrix(nrow=0, ncol=2), zerosAfit="sparse",
           zerosBfit="sparse", zerosP=matrix(nrow=0, ncol=2), cliquesP=list(),
           separatorsP=list(), unbalanced=matrix(nrow=0, ncol=2), diagP=FALSE,
           efficient=TRUE, nInit=100, minSuccDiff=0.001)
```

Arguments

Y	Three-dimensional array containing the response data. The first, second and third dimensions correspond to variates, time and samples, respectively. The data are assumed to be centered covariate-wise.
X	Three-dimensional array containing the time-varying covariate data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
lambdaA	Ridge penalty parameter (positive numeric of length 1) to be used in the estimation of $\mathbf{A}$, the matrix with autoregression coefficients.
lambdaB	Ridge penalty parameter (positive numeric of length 1) to be used in the estimation of $\mathbf{B}$, the matrix with regression coefficients of the time-varying covariates stored in array X.
lambdaP	Ridge penalty parameter (positive numeric of length 1) to be used in the estimation of the inverse error covariance matrix ($\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$): the precision matrix of the errors.
lagX	Integer, either 0 or 1, specifying whether $\mathbf{X}_t$ or $\mathbf{X}_{t-1}$ affects $\mathbf{Y}_t$, respectively.
targetA	Target matrix to which the matrix $\mathbf{A}$ is to be shrunken.
targetB	Target matrix to which the matrix $\mathbf{B}$ is to be shrunken.
targetP	Target matrix to which the in the inverse error covariance matrix, the precision matrix, is to be shrunken.
fitAB	A character. If fitAB="ml" the parameters $\mathbf{A}$ and $\mathbf{B}$ are estimated by (penalized) maximum likelihood. If fitAB="ss" the parameters $\mathbf{A}$ and $\mathbf{B}$ are estimated by (penalized) sum of squares.
targetPtype	A character indicating the type of target to be used for the precision matrix. When specified it overrules the targetP-option. See the default.target-function for the options.
zerosA	A matrix with indices of entries of $\mathbf{A}$ that are constrained to zero. The matrix comprises two columns, each row corresponding to an entry of $\mathbf{A}$. The first column contains the row indices and the second the column indices.
zerosB	A matrix with indices of entries of $\mathbf{B}$ that are constrained to zero. The matrix comprises two columns, each row corresponding to an entry of $\mathbf{B}$. The first column contains the row indices and the second the column indices.
zerosAfit	A character, either "sparse" or "dense". With "sparse", the matrix $\mathbf{A}$ is assumed to contain many zeros and a computational efficient implementation of its estimation is employed. If "dense", it is assumed that $\mathbf{A}$ contains only few zeros and the estimation method is optimized computationally accordingly.
zerosBfit	A character, either "sparse" or "dense". With "sparse", the matrix $\mathbf{B}$ is assumed to contain many zeros and a computational efficient implementation of its estimation is employed. If "dense", it is assumed that $\mathbf{B}$ contain only few zeros and the estimation method is optimized computationally accordingly.
zerosP	A matrix-object with indices of entries of the precision matrix that are constrained to zero. The matrix comprises two columns, each row corresponding to an entry of the adjacency matrix. The first column contains the row indices

and the second the column indices. The specified graph should be undirected and decomposable. If not, it is symmetrized and triangulated (unless `cliquesP` and `seperatorsP` are supplied). Hence, the employed zero structure may differ from the input `zerosP`.

`cliquesP` A `list`-object containing the node indices per clique as object from the `rip`-function.

`separatorsP` A `list`-object containing the node indices per clique as object from the `rip`-function.

`unbalanced` A `matrix` with two columns, indicating the unbalances in the design. Each row represents a missing design point in the (time x individual)-layout. The first and second column indicate the time and individual (respectively) specifics of the missing design point.

`diagP` A `logical`, indicates whether the inverse error covariance matrix is assumed to be diagonal.

`efficient` A `logical`, affects estimation of $\mathbf{A}$. Details below.

`nInit` Maximum number of iterations (positive `numeric` of length 1) to be used in maximum likelihood estimation.

`minSuccDiff` Minimum distance (positive `numeric` of length 1) between estimates of two successive iterations to be achieved.

Details

If `diagP=TRUE`, no penalization to estimation of the covariance matrix is applied. Consequently, the arguments `lambdaP` and `targetP` are ignored (if supplied).

The ridge ML estimator employs the following estimator of the variance of the VARX(1) process:

$$\frac{1}{n(\mathcal{T}-1)}\sum_{i=1}^{n}\sum_{t=2}^{\mathcal{T}}\mathbf{Y}_{*,i,t}\mathbf{Y}_{*,i,t}^{\top}.$$

This is used when `efficient=FALSE`. However, a more efficient estimator of this variance can be used

$$\frac{1}{n\mathcal{T}}\sum_{i=1}^{n}\sum_{t=1}^{\mathcal{T}}\mathbf{Y}_{*,i,t}\mathbf{Y}_{*,i,t}^{\top},$$

which is achieved by setting when `efficient=TRUE`. Both estimators are adjusted accordingly when dealing with an unbalanced design.

Value

A list-object with slots:

`A` Ridge ML estimate of the matrix $\mathbf{A}$, the `matrix` with lag one autoregression coefficients.

`B` Ridge ML estimate of the matrix $\mathbf{B}$, the `matrix` with regression coefficients of the time-varying covariates.

`P` Ridge ML estimate of the inverse error covariance `matrix` $\boldsymbol{\Omega}_{\varepsilon}(=\boldsymbol{\Sigma}_{\varepsilon}^{-\mathbf{1}})$.

lambdaA	Positive numeric of length one: ridge penalty used in the estimation of $\mathbf{A}$.
lambdaB	Positive numeric of length one: ridge penalty used in the estimation of $\mathbf{B}$.
lambdaP	Positive numeric of length one: ridge penalty used in the estimation of inverse error covariance matrix $\mathbf{\Omega}_\varepsilon (= \mathbf{\Sigma}_\varepsilon^{-1})$.

Note

When the target of the precision matrix is specified through the targetPtype-argument, the target is data-driven and updated at each iteration when fitAB="ml".

In case $\lambda_a \neq \lambda_b$, the ridge ML estimates (conditional on the current estimate of $\mathbf{\Omega}_\varepsilon$) of $\mathbf{A}$ and $\mathbf{B}$ are approximations. Their explicit evaluation involves a Kronecker product times a vector. Its full expansion is memory inefficient, in particular for (say) $p_y > 50$. If $\lambda_a = \lambda_b$ the estimates of $\mathbf{A}$ and $\mathbf{B}$ can be evaluated through multiplications of matrices of order $p_y + p_x$ (instead of $p_y^2 + p_y p_x$). The evaluation cannot be simplified computationally when $\lambda_a \neq \lambda_b$. To avoid the use of matrices of order $p_y^2 + p_y p_x$ an approximation is used. For the approximation it is assumed that the contemporaneous covariance between $\mathbf{Y}_t$ and $\mathbf{X}_t$ is zero (for lag one), after which the estimates can be evaluated using matrices of order $p_y + p_x$.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>

References

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

See Also

default.target, loglikLOOCVVAR1, ridgeP, ridgePchordal.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
Ax     <- createA(p, "chain", nBands=1)

# generate time-varying covariates in accordance with VAR(1) process
X <- dataVAR1(n, T, Ax, SigmaE)

# set model parameters
B <- createA(p, "clique", nCliques=1)
A <- createA(p, "hub", nHubs=1)

# generate time-varying covariates in accordance with VAR(1) process
Y <- dataVARX1(X, A, B, SigmaE, lagX=0)
```

```
# fit VARX(1) model
ridgeVARX1(Y, X, 1, 1, 1, lagX=0)
```

---

sparsifyVAR1 *Function that determines the support of autoregression parameter of the VARX(1) model.*

Description

Function that determines the null and non-null elements of $\mathbf{A}$, the matrix of autoregression coefficients.

Usage

```
sparsifyVAR1(A, SigmaE, threshold=c("absValue", "localFDR", "top"),
             absValueCut=0.25, FDRcut=0.8, top=10,
             zerosA=matrix(nrow=0, ncol=2), statistics=FALSE,
             verbose=TRUE)
```

Arguments

A	Matrix $\mathbf{A}$ of autoregression parameters.
SigmaE	Covariance matrix of the errors (innovations).
threshold	A character signifying type of sparsification of $\mathbf{A}$ by thresholding. Must be one of: "absValue", "localFDR", or "top".
absValueCut	A numeric giving the cut-off for element selection based on absolute value thresholding.
FDRcut	A numeric giving the cut-off for element selection based on local false discovery rate (FDR) thresholding.
top	A numeric giving the number of elements of A which is to be selected, based on absolute value thresholding.
zerosA	Matrix with indices of entries of $\mathbf{A}$ that are (prior to sparsification) known to be zero. The matrix comprises two columns, each row corresponding to an entry of $\mathbf{A}$. The first column contains the row indices and the second the column indices.
statistics	Logical indicator: should test statistics be returned. This only applies when threshold = "localFDR"
verbose	Logical indicator: should intermediate output be printed on the screen?

Details

When threshold = "localFDR" the function, following Lutkepohl (2005), divides the elements of (possibly regularized) input matrix $\mathbf{A}$ of autoregression coefficients by (an approximation of) their standard errors. Subsequently, the support of the matrix $\mathbf{A}$ is determined by usage of local FDR. In that case a mixture model is fitted to the nonredundant (standardized) elements of $\mathbf{A}$ by fdrtool. The decision to retain elements is then based on the argument FDRcut. Elements with a posterior

probability $>= q$ FDRcut (equalling 1 - local FDR) are retained. See Strimmer (2008) for further details. Alternatively, the support of $\mathbf{A}$ is determined by simple thresholding on the absolute values of matrix entries (threshold = "absValue"). A third option (threshold = "top") is to retain a prespecified number of matrix entries based on absolute values of the elements of $\mathbf{A}$. For example, one could wish to retain those entries representing the ten strongest cross-temporal coefficients.

The argument absValueCut is only used when threshold = "absValue". The argument FDRcut is only used when threshold = "localFDR". The argument top is only used when threshold = "top".

When prior to the sparsification knowledge on the support of $\mathbf{A}$ is specified through the option zerosA, the corresponding elements of $\mathbf{A}$ are then not taken along in the local FDR procedure.

Value

A list-object with slots:

zerosA	Matrix with indices of entries of $\mathbf{A}$ that are identified to be null. It includes the elements of $\mathbf{A}$ assumed to be zero prior to the sparsification as specified through the zerosAknown option.
nonzerosA	Matrix with indices of entries of $\mathbf{A}$ that are identified to be non-null.
statisticsA	Matrix with test statistics employed in the local FDR procedure.

The matrices zerosA and nonzerosA comprises two columns, each row corresponding to an entry of $\mathbf{A}$. The first column contains the row indices and the second the column indices.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>, Carel F.W. Peeters.

References

Lutkepohl, H. (2005), New Introduction to Multiple Time Series Analysis. Springer, Berlin.

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2017), "Ridge estimation of the VAR(1) model and its time series chain graph from multivariate time-course omics data", *Biometrical Journal*, 59(1), 172-191.

Strimmer, K. (2008), “fdrtool: a versatile R package for estimating local and tail area-based false discovery rates”, *Bioinformatics* 24(12): 1461-1462.

Van Wieringen, W.N., Peeters, C.F.W. (2016), “Ridge Estimation of Inverse Covariance Matrices from High-Dimensional Data”, *Computational Statistics and Data Analysis*, 103, 284-303.

See Also

ridgeVAR1, sparsify.

Examples

```
# set dimensions
p <- 3
n <- 4
T <- 10

# set model parameters
SigmaE <- diag(p)/4
A <- matrix(c(-0.1, -0.3, 0.6, 0.5, -0.4, 0, 0.3, -0.5, -0.2),
     byrow=TRUE, ncol=3)

# generate data
Y <- dataVAR1(n, T, A, SigmaE)

# fit VAR(1) model
VAR1hat <- ridgeVAR1(Y, 1, 1)

## determine which elements of A are non-null
## Not run:  Anullornot <- matrix(0, p, p)
## Not run:  Anullornot[sparsifyVAR1(VAR1hat$A, solve(VAR1hat$P),
threshold="localFDR")$nonzeros] <- 1
## End(Not run)
## REASON FOR NOT RUN:
## the employed local FDR approximation is only valid for reasonably sized
## number of elements of A (say) at least p > 10 and,
## consequently, a vector of 100 regression coefficients.

## plot non-null structure of A
## Not run:  edgeHeat(Anullornot)
```

`sparsifyVAR2` *Function that determines the support of autoregression parameters of the VAR(2) model.*

Description

Function that determines the null and non-null elements of $\mathbf{A}_1$ and $\mathbf{A}_2$, the matrices of lag one and two (respectively) autoregression coefficients.

Usage

```
sparsifyVAR2(A1, A2, SigmaE, threshold=c("absValue", "localFDR", "top"),
             absValueCut=c(0.25, 0.25), FDRcut=c(0.8, 0.8), top=c(10,10),
             zerosA1=matrix(nrow=0, ncol=2),
             zerosA2=matrix(nrow=0, ncol=2),
             statistics=FALSE, verbose=TRUE)
```

Arguments

A1	A matrix $\mathbf{A}_1$ of lag one autoregression parameters.
A2	A matrix $\mathbf{A}_2$ of lag two autoregression parameters.
SigmaE	Covariance matrix of the errors (innovations).
threshold	A character signifying type of sparsification of both $\mathbf{A}_1$ and $\mathbf{A}_2$ by thresholding. Must be one of: "absValue", "localFDR", or "top".
absValueCut	A numeric (of length two) giving the cut-offs for element selection based on absolute value thresholding. The elements are the cut-offs applied to $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively.
FDRcut	A numeric (of length two) giving the cut-offs for element selection based on local false discovery rate (FDR) thresholding. The elements are the cut-offs applied to $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively.
top	A numeric (of length two) specifying the numbers of elements of $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively, which are to be selected.
zerosA1	A matrix with indices of entries of $\mathbf{A}_1$ that are (prior to sparsification) known to be zero. The matrix comprises two columns, each row corresponding to an entry of $\mathbf{A}_1$. The first column contains the row indices and the second the column indices.
zerosA2	A matrix with indices of entries of $\mathbf{A}_2$ that are (prior to sparsification) known to be zero. The matrix comprises two columns, each row corresponding to an entry of $\mathbf{A}_2$. The first column contains the row indices and the second the column indices.
statistics	A logical indicator: should test statistics be returned. This only applies when threshold = "localFDR"
verbose	A logical indicator: should intermediate output be printed on the screen?

Details

When threshold = "localFDR" the function, following Lutkepohl (2005), divides the elements of (possibly regularized) input matrix $\mathbf{A}_1$ (or $\mathbf{A}_2$) of lag one (or two) autoregression coefficients by (an approximation of) their standard errors. Subsequently, the support of the matrix $\mathbf{A}_1$ (or $\mathbf{A}_2$) is determined by usage of the local FDR. In that case a mixture model is fitted to the nonredundant (standardized) elements of $\mathbf{A}_1$ (or $\mathbf{A}_2$) by fdrtool. The decision to retain elements is then based on the argument FDRcut. Elements with a posterior probability $>= q$ FDRcut (equalling 1 - local FDR) are retained. See Strimmer (2008) for sparsifyfurther details. Alternatively, the support of $\mathbf{A}_1$ (or $\mathbf{A}_2$) is determined by simple thresholding on the absolute values of matrix entries (threshold = "absValue"). A third option (threshold = "top") is to retain a prespecified number of matrix entries based on absolute values of the elements of $\mathbf{A}_1$ (or $\mathbf{A}_2$). For example, one could wish to retain those entries representing the ten strongest cross-temporal coefficients.

The argument absValueCut is only used when threshold = "absValue". The argument FDRcut is only used when threshold = "localFDR". The argument top is only used when threshold = "top".

When prior to the sparsification knowledge on the support of $\mathbf{A}_1$ (or $\mathbf{A}_2$) is specified through the option zerosA1 (or zerosA2), the corresponding elements of $\mathbf{A}_1$ (or $\mathbf{A}_2$) are then not taken along in the local FDR procedure.

Value

A `list`-object with slots:

`zerosA1`	`Matrix` with indices of entries of $\mathbf{A}_1$ that are identified to be null. It includes the elements of $\mathbf{A}_1$ assumed to be zero prior to the sparsification as specified through the `zerosA1` option.
`nonzerosA1`	`Matrix` with indices of entries of $\mathbf{A}_1$ that are identified to be non-null.
`statisticsA1`	`Matrix` with test statistics employed in the local FDR procedure.
`zerosA2`	`Matrix` with indices of entries of $\mathbf{A}_2$ that are identified to be null. It includes the elements of $\mathbf{A}_2$ assumed to be zero prior to the sparsification as specified through the `zerosA2` option.
`nonzerosA2`	`Matrix` with indices of entries of $\mathbf{A}_2$ that are identified to be non-null.
`statisticsA2`	`Matrix` with test statistics employed in the local FDR procedure.

The matrices `zerosA1`, `nonzerosA1`, `zerosA2` and `nonzerosA2` comprise two columns, each row corresponding to an entry of $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively. The first column contains the row indices and the second the column indices.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>, Carel F.W. Peeters.

References

Lutkepohl, H. (2005), New Introduction to Multiple Time Series Analysis. Springer, Berlin.

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

Strimmer, K. (2008), "fdrtool: a versatile R package for estimating local and tail area-based false discovery rates", *Bioinformatics* 24(12): 1461-1462.

Van Wieringen, W.N., Peeters, C.F.W. (2016), "Ridge Estimation of Inverse Covariance Matrices from High-Dimensional Data", *Computational Statistics and Data Analysis*, 103, 284-303.

See Also

`ridgeVAR2`, `sparsify`, `sparsifyVAR1`.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 12; T <- 10

# set model parameters
SigmaE <- diag(p)/4
A1 <- -createA(p, "clique", nCliques=1, nonzeroA=0.1)
A2 <- t(createA(p, "chain", nBands=1, nonzeroA=0.1))

# generate data
Y <- dataVAR2(n, T, A1, A2, SigmaE)
```

```
# fit VAR(1) model
VAR2hat <- ridgeVAR2(Y, 1, 1, 1)

# obtain support of adjacancy matrix
A1nullornot <- matrix(0, p, p)
A1nullornot[sparsifyVAR2(VAR2hat$A1, VAR2hat$A2, solve(VAR1hat$P),
                         threshold="top", top=c(3,3))$nonzerosA1] <- 1

## plot non-null structure of A1
edgeHeat(A1nullornot)
```

`sparsifyVARX1` *Function that determines the support of (auto)regression parameters of the VARX(1) model.*

Description

Function that determines the support of $\mathbf{A}$, the matrix with autoregression coefficients, and $\mathbf{B}$, the matrix with regression coefficients of the time-varying covariates, of the VARX(1) model.

Usage

```
sparsifyVARX1(X, A, B, SigmaE, threshold=c("absValue", "localFDR", "top"),
              absValueCut=rep(0.25, 2), FDRcut=rep(0.8, 2), top=rep(10, 2),
              zerosA=matrix(nrow=0, ncol=2), zerosB=matrix(nrow=0, ncol=2),
              statistics=FALSE, verbose=TRUE)
```

Arguments

`X`	Three-dimensional `array` containing the time-varying covariate data. The first, second and third dimensions correspond to covariates, time and samples, respectively. The data are assumed to be centered covariate-wise.
`A`	A `matrix` $\mathbf{A}$ of autoregression parameters.
`B`	A `matrix` $\mathbf{B}$ of regression parameters of the time-varying covariates stored in the array `X`.
`SigmaE`	Covariance `matrix` of the errors (innovations).
`threshold`	A `character` signifying type of sparsification of $\mathbf{A}$ and $\mathbf{B}$ by thresholding. Must be one of: "absValue", "localFDR", or "top".
`absValueCut`	A `numeric` of length two giving the cut-offs for element selection based on absolute value thresholding.
`FDRcut`	A `numeric` of length two giving the cut-off (for $\mathbf{A}$ and $\mathbf{B}$, respectively) for element selection based on local false discovery rate (FDR) thresholding.
`top`	A `numeric` of length two giving the number of elements of `A` and $\mathbf{B}$, respectively, which is to be selected, based on absolute value thresholding.

zerosA	Matrix with indices of entries of $\mathbf{A}$ that are (prior to sparsification) known to be zero. The matrix comprises two columns, each row corresponding to an entry of $\mathbf{A}$. The first column contains the row indices and the second the column indices.
zerosB	A Matrix with indices of entries of $\mathbf{B}$ that are (prior to sparsification) known to be zero. The matrix comprises two columns, each row corresponding to an entry of $\mathbf{B}$. The first column contains the row indices and the second the column indices.
statistics	A Logical indicator: should test statistics be returned. This only applies when threshold = "localFDR"
verbose	Logical indicator: should intermediate output be printed on the screen?

Details

When threshold = "localFDR" the function, following Lutkepohl (2005), divides the elements of (possibly regularized) input matrix $\mathbf{A}$ of autoregression coefficients by (approximations of) their standard errors. Subsequently, the support of the matrix $\mathbf{A}$ is determined by usage of local FDR. In that case a mixture model is fitted to the nonredundant (standardized) elements of $\mathbf{A}$ by fdrtool. The decision to retain elements is then based on the argument FDRcut. Elements with a posterior probability $>= q$ FDRcut (equalling 1 - local FDR) are retained. See Strimmer (2008) for further details. Alternatively, the support of $\mathbf{A}$ is determined by simple thresholding on the absolute values of matrix entries (threshold = "absValue"). A third option (threshold = "top") is to retain a prespecified number of matrix entries based on absolute values of the elements of $\mathbf{A}$. For example, one could wish to retain those entries representing the ten strongest cross-temporal coefficients.

The argument absValueCut is only used when threshold = "absValue". The argument FDRcut is only used when threshold = "localFDR". The argument top is only used when threshold = "top".

When prior to the sparsification knowledge on the support of $\mathbf{A}$ and/or $\mathbf{B}$ is specified through the options zerosA and/or zerosB, the corresponding elements of $\mathbf{A}$ and $\mathbf{B}$ are then not taken along in the local FDR procedure.

Value

A list-object with slots:

zerosA	Matrix with indices of entries of $\mathbf{A}$ that are identified to be null. It includes the elements of $\mathbf{A}$ assumed to be zero prior to the sparsification as specified through the zerosAknown option.
nonzerosA	Matrix with indices of entries of $\mathbf{A}$ that are identified to be non-null.
statisticsA	Matrix with test statistics employed in the local FDR procedure for the sparsification of $\mathbf{A}$.
zerosB	Matrix with indices of entries of $\mathbf{B}$ that are identified to be null. It includes the elements of $\mathbf{B}$ assumed to be zero prior to the sparsification as specified through the zerosBknown option.
nonzerosB	Matrix with indices of entries of $\mathbf{B}$ that are identified to be non-null.
statisticsB	Matrix with test statistics employed in the local FDR procedure for the sparsification of $\mathbf{B}$.

The matrices zerosA, nonzerosA, zerosB and nonzerosB comprise two columns, each row corresponding to an entry of either $\mathbf{A}$ or $\mathbf{B}$. The first column contains the row indices and the second the column indices.

Author(s)

Wessel N. van Wieringen <w.vanwieringen@vumc.nl>, Carel F.W. Peeters.

References

Lutkepohl, H. (2005), *New Introduction to Multiple Time Series Analysis*. Springer, Berlin.

Miok, V., Wilting, S.M., Van Wieringen, W.N. (2019), "Ridge estimation of network models from time-course omics data", *Biometrical Journal*, 61(2), 391-405.

Strimmer, K. (2008), "fdrtool: a versatile R package for estimating local and tail area-based false discovery rates", *Bioinformatics* 24(12): 1461-1462.

Van Wieringen, W.N., Peeters, C.F.W. (2016), "Ridge Estimation of Inverse Covariance Matrices from High-Dimensional Data", *Computational Statistics and Data Analysis*, 103, 284-303.

See Also

ridgeVAR1, sparsify, sparsifyVAR1.

Examples

```
# set dimensions (p=covariates, n=individuals, T=time points)
p <- 3; n <- 4; T <- 10

# set model parameters
SigmaE <- diag(p)/4
Ax     <- createA(p, "chain", nBands=1)

# generate time-varying covariates in accordance with VAR(1) process
X <- dataVAR1(n, T, Ax, SigmaE)

# set model parameters
B <- createA(p, "clique", nCliques=1)
A <- createA(p, "hub", nHubs=1)

# generate time-varying covariates in accordance with VAR(1) process
Y <- dataVARX1(X, A, B, SigmaE, lagX=0)

# fit VARX(1) model
VARX1hat <- ridgeVARX1(Y, X, 1, 1, 1, lagX=0)

## determine which elements of A are non-null
## Not run:  Anullornot <- matrix(0, p, p)
## Not run:  Anullornot[sparsifyVARX1(X, VARX1hat$A, VARX1hat$B,
        solve(VARX1hat$P), threshold="localFDR")$nonzeros] <- 1
## End(Not run)
## REASON FOR NOT RUN:
## the employed local FDR approximation is only valid for reasonably sized
```

```
## number of elements of A (say) at least p > 10 and,
## consequently, a vector of 100 regression coefficients.

## plot non-null structure of A
## Not run:  edgeHeat(Anullornot)
```

Index