Package ‘porridge’

May 18, 2020

Type Package

Title Ridge-Type Penalized Estimation of a Potpourri of Models

Version 0.1.0

Date 2020-05-17

Author Wessel N. van Wieringen [aut, cre],
Mehran Aflakparast [ctb] (part of the R-code of the mixture
functionality)

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

Description The name of the package is derived from the French, 'pour' ridge, and provides functionality for ridge-type estimation of a potpourri of models. Currently, this estimation concerns that of various Gaussian graphical models from different study designs. Among others it considers the regular Gaussian graphical model and a mixture of such models. The porridge-package implements the estimation of the former either from i) data with replicated observations by penalized loglikelihood maximization using the regular ridge penalty on the parameters (van Wieringen, Chen, 2019) or ii) from non-replicated data by means of the generalized ridge estimator that allows for both the inclusion of quantitative and qualitative prior information on the precision matrix via element-wise penalization and shrinkage (van Wieringen, 2019, <doi.org/10.1080/10618600.2019.1604374>). Additionally, the porridge-package facilitates the ridge penalized estimation of a mixture of Gaussian graphical models (Aflakparast et al., 2018, <doi.org/10.1002/bimj.201700102>).

License GPL (>= 2)

LazyLoad yes

URL http://www.few.vu.nl/~wvanwie

Depends R (>= 2.15.1)

Imports MASS, stats, mvtnorm, Rcpp, methods

Suggests Matrix

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-05-18 07:50:02 UTC

R topics documented:

`porridge-package` *Ridge-Type Penalized Estimation of a Potpourri of Models.*

Description

The following functions facilitate the ridge-type penalized estimation of various models. Currently, it includes:

- Generalized ridge estimation of the precision matrix of a Gaussian graphical model (van Wieringen, 2019) through the function `ridgePgen`. This function is complemented by the functions `ridgePgen.kCV`, `ridgePgen.kCV.banded`, `ridgePgen.kCV.groups`, `optPenaltyPgen.kCVauto.banded` and `optPenaltyPgen.kCVauto.groups` for penalty parameters selection through K-fold cross-validation assuming a particularly structured precision matrix.
- Gaussian graphical model estimation from data with replicates in ridge penalized fashion (Chen, van Wieringen, 2019) (`ridgePrep` and `ridgePrepEdiag`). The two functions `optPenaltyPrep.kCVauto` and `optPenaltyPrepEdiag.kCVauto` implement the corresponding K-fold cross-validation procedures for an optimal choice of the penalty parameter.
- Ridge penalized estimation of a mixture of Gaussian graphical models: `ridgeGGMmixture` and its penalty selection via K-fold cross-validation `optPenaltyGGMmixture.kCVauto`.

Future versions aim to include more ridge-type functionality.

In part the `porridge`-package extends/builds upon the `rags2ridges`- and `ragt2ridges`-packages, in which some or all functionality of the `porridge`-package may be absorbed at some point in the future.

Details

Package:	porridge
Type:	Package
Version:	0.1.0
Date:	2020-05-17
License:	GPL (>= 2)

Author(s)

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

References

Aflakparast, M., de Gunst, M.C.M., van Wieringen, W.N. (2018), "Reconstruction of molecular network evolution from cross-sectional omics data", *Biometrical Journal*, 60(3), 547-563.

Peeters, C.F.W., Bilgrau, A.E., and van Wieringen, W.N. (2019), "`rags2ridges`: Ridge Estimation of Precision Matrices from High-Dimensional Data", R package version 2.2.2. `https://CRAN.R-project.org/package=rags2ridges`.

van Wieringen, W.N. (2020), "`ragt2ridges`: Ridge Estimation of Vector Auto-Regressive (VAR) Processes", R package version 0.3.4. `https://CRAN.R-project.org/package=ragt2ridges`.

van Wieringen, W.N. (2019), "The generalized ridge estimator of the inverse covariance matrix", *Journal of Computational and Graphical Statistics*, 28(4), 932-942.

van Wieringen W.N., Chen, Y. (2019), "Penalized estimation of the Gaussian graphical model from data with replicates", submitted.

van Wieringen, W.N., Stam, K.A., Peeters, C.F.W., van de Wiel, M.A. (2020), "Updating of the Gaussian graphical model through targeted penalized estimation", *Journal of Multivariate Analysis*, 178, Article 104621.

See Also

The `porridge`-package.

`optPenaltyGGMmixture.kCVauto`

Automatic search for optimal penalty parameter (mixture of GGMs).

Description

Function that performs an automatic search for the optimal penalty parameter for the `ridgeGGMmixture` call by employing Brent's method to the calculation of a cross-validated (negative) log-likelihood score.

Usage

```
optPenaltyGGMmixture.kCVauto(Y, K, lambdaMin, lambdaMax,
                lambdaInit=(lambdaMin+lambdaMax)/2,
                fold=nrow(Y), target,
                iWeights=matrix(sample(seq(0+1/nrow(Y),
                                1-1/nrow(Y), by=1/(2*nrow(Y))),
                                nrow(Y)*K, replace=TRUE),
                                nrow=nrow(Y), ncol=K),
                nInit=100, minSuccDiff=10^(-10),
                minMixProp=0.01)
```

Arguments

`Y`	Data `matrix` with samples as rows and variates as columns.
`K`	A `numeric`, specifying the number of mixture components.
`lambdaMin`	A `numeric` giving the minimum value for the penalty parameter.
`lambdaMax`	A `numeric` giving the maximum value for the penalty parameter.
`lambdaInit`	A `numeric` giving the initial (starting) value for the penalty parameter.
`fold`	A `numeric` or `integer` specifying the number of folds to apply in the cross-validation.
`target`	A semi-positive definite target `matrix` towards which the estimate is shrunken.
`iWeights`	Sample-specific positive component weight `matrix`. Rows correspond to samples, while columns to components.
`nInit`	A `numeric` specifying the number of iterations.
`minSuccDiff`	A `numeric`: minimum successive difference (in terms of their penalized loglikelihood) between two succesive estimates to be achieved.
`minMixProp`	Smallest mixing probability tolerated.

Value

The function returns a positive `numeric`, the cross-validated optimal penalty parameter.

Note

The elements of `iWeights` may be larger than one as they are rescaled internally to sum to one.

Author(s)

W.N. van Wieringen, M. Aflakparast.

References

Aflakparast, M., de Gunst, M.C.M., van Wieringen, W.N. (2018), "Reconstruction of molecular network evolution from cross-sectional omics data", *Biometrical Journal*, 60(3), 547-563.

See Also

ridgeGGMmixture

Examples

```
# define mixing proportions
pis <- c(0.2, 0.3, 0.4)

# set dimension and sample size
p <- 5
n <- 100

# define population covariance matrices
diags       <- list(rep(1,    p),
                    rep(0.5,  p-1),
                    rep(0.25, p-2),
                    rep(0.1,  p-3))
Omega       <- as.matrix(Matrix::bandSparse(p,
                                            k   =-c(0:3),
                                            diag=c(diags),
                                            symm=TRUE))
Sigma1      <- solve(Omega)
Omega       <- matrix(0.3, p, p)
diag(Omega) <- 1
Sigma2      <- solve(Omega)
Sigma3      <- cov(matrix(rnorm(p*n), ncol=p))

# mean vectors
mean1 <- rep(0,p)
mean2 <- rexp(p)
mean3 <- rnorm(p)

# draw data data from GGM mixture
Z <- sort(sample(c(1:3), n, prob=pis, replace=TRUE))
Y <- rbind(mvtnorm::rmvnorm(sum(Z==1), mean=mean1, sigma=Sigma1),
           mvtnorm::rmvnorm(sum(Z==2), mean=mean2, sigma=Sigma2),
           mvtnorm::rmvnorm(sum(Z==3), mean=mean3, sigma=Sigma3))

# find optimal penalty parameter
optLambda <- optPenaltyGGMmixture.kCVauto(Y,  K,
                                          0.00001, 100,
                                          10, fold=5,
                                          target=0*Sigma1)

# ridge penalized estimation of the GGM mixture
ridgeGGMmixFit <- ridgeGGMmixture(Y, 3, optLambda, target=0*Sigma1)
```

optPenaltyPgen.kCVauto.banded

Automatic search for optimal penalty parameter (generalized ridge precision).

Description

Function that determines the optimal penalty parameters through maximization of the k-fold cross-validated log-likelihood score, with a penalization that encourages banded precisions.

Usage

```
optPenaltyPgen.kCVauto.banded(Y, lambdaMin, lambdaMax,
                              lambdaInit=(lambdaMin + lambdaMax)/2,
                              fold=nrow(Y), target,
                              zeros=matrix(nrow=0, ncol=2),
                              penalize.diag=TRUE, nInit=100,
                              minSuccDiff=10^(-5))
```

Arguments

`Y`	Data `matrix` with samples as rows and variates as columns.
`lambdaMin`	A `numeric` giving the minimum value for the penalty parameters. One value per group. Values should be specified in the same order as the first appearance of a group representative.
`lambdaMax`	A `numeric` giving the maximum value for the penalty parameters. One value per group. Values should be specified in the same order as the first appearance of a group representative.
`lambdaInit`	A `numeric` giving the initial (starting) value for the penalty parameters. One value per group. Values should be specified in the same order as the first appearance of a group representative.
`fold`	A `numeric` or `integer` specifying the number of folds to apply in the cross-validation.
`target`	A semi-positive definite target `matrix` towards which the estimate is shrunken.
`zeros`	A two-column `matrix`, with the first and second column containing the row- and column-index of zero precision elements.
`penalize.diag`	A `logical` indicating whether the diagonal should be penalized.
`nInit`	A `numeric` specifying the number of iterations.
`minSuccDiff`	A `numeric`: minimum successive difference (in terms of their penalized loglikelihood) between two succesive estimates to be achieved.

Details

The penalty matrix $\mathbf{\Lambda}$ is parametrized as follows. The elements of $\mathbf{\Lambda}$ are $(\mathbf{\Lambda})_{j,j'} = \lambda(|j - j'| + 1)$ for $j, j' = 1, \ldots, p$.

Value

The function returns a `numeric` containing the cross-validated optimal positive penalty parameters.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N. (2019), "The generalized ridge estimator of the inverse covariance matrix", *Journal of Computational and Graphical Statistics*, 28(4), 932-942.

See Also

ridgePgen

Examples

```
# set dimension and sample size
p <- 10
n <- 10

# penalty parameter matrix
lambda       <- matrix(1, p, p)
diag(lambda) <- 0.1

# generate precision matrix
Omega       <- matrix(0.4, p, p)
diag(Omega) <- 1
Sigma       <- solve(Omega)

# data
Y <- mvtnorm::rmvnorm(n, mean=rep(0,p), sigma=Sigma)
S <- cov(Y)

# find optimal penalty parameters through cross-validation
lambdaOpt <- optPenaltyPgen.kCVauto.banded(Y, 10^(-10), 10^(10),
                          target=matrix(0, p, p),
                          penalize.diag=FALSE, nInit=100,
                          minSuccDiff=10^(-5))

# format the penalty matrix
lambdaOptMat <- matrix(NA, p, p)
for (j1 in 1:p){
    for (j2 in 1:p){
        lambdaOptMat[j1, j2] <- lambdaOpt * (abs(j1-j2)+1)
    }
}

# generalized ridge precision estimate
Phat <- ridgePgen(S, lambdaOptMat, matrix(0, p, p))
```

---

optPenaltyPgen.kCVauto.groups

Automatic search for optimal penalty parameter (generalized ridge precision).

Description

Function that determines the optimal penalty parameters through maximization of the k-fold cross-validated log-likelihood score, assuming that variates are grouped and penalized group-wise.

Usage

```
optPenaltyPgen.kCVauto.groups(Y, lambdaMin, lambdaMax,
                              lambdaInit=(lambdaMin + lambdaMax)/2,
                              fold=nrow(Y), groups, target,
                              zeros=matrix(nrow=0, ncol=2),
                              penalize.diag=TRUE, nInit=100,
                              minSuccDiff=10^(-5))
```

Arguments

`Y`	Data `matrix` with samples as rows and variates as columns.
`lambdaMin`	A `numeric` giving the minimum value for the penalty parameters. One value per group. Values should be specified in the same order as the first appearance of a group representative.
`lambdaMax`	A `numeric` giving the maximum value for the penalty parameters. One value per group. Values should be specified in the same order as the first appearance of a group representative.
`lambdaInit`	A `numeric` giving the initial (starting) value for the penalty parameters. One value per group. Values should be specified in the same order as the first appearance of a group representative.
`fold`	A `numeric` or `integer` specifying the number of folds to apply in the cross-validation.
`groups`	A`numeric` indicating to which group a variate belongs. Same values indicate same group.
`target`	A semi-positive definite target `matrix` towards which the estimate is shrunken.
`zeros`	A two-column `matrix`, with the first and second column containing the row- and column-index of zero precision elements.
`penalize.diag`	A `logical` indicating whether the diagonal should be penalized.
`nInit`	A `numeric` specifying the number of iterations.
`minSuccDiff`	A `numeric`: minimum successive difference (in terms of their penalized loglikelihood) between two succesive estimates to be achieved.

Details

The penalty matrix $\mathbf{\Lambda}$ is parametrized as follows. The elements of $\mathbf{\Lambda}$ are $(\mathbf{\Lambda})_{j,j'} = \frac{1}{2}(\lambda_k + \lambda_{k'})$ for $j, j' = 1, \ldots, p$ if j and j' belong to groups k and k', respectively, where λ_k and $\lambda_{k'}$ are the corresponding group-specific penalty parameters.

Value

The function returns a `numeric` containing the cross-validated optimal positive penalty parameters.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N. (2019), "The generalized ridge estimator of the inverse covariance matrix", *Journal of Computational and Graphical Statistics*, 28(4), 932-942.

See Also

`ridgePgen`

Examples

```
# set dimension and sample size
p <- 10
n <- 10

# penalty parameter matrix
lambda       <- matrix(1, p, p)
diag(lambda) <- 0.1

# generate precision matrix
Omega       <- matrix(0.4, p, p)
diag(Omega) <- 1
Sigma       <- solve(Omega)

# data
Y <- mvtnorm::rmvnorm(n, mean=rep(0,p), sigma=Sigma)
S <- cov(Y)

# find optimal penalty parameters through cross-validation
lambdaOpt <- optPenaltyPgen.kCVauto.groups(Y, rep(10^(-10), 2), rep(10^(10), 2),
                          groups=c(rep(0, p/2), rep(1, p/2)),
                          target=matrix(0, p, p),
                          penalize.diag=FALSE, nInit=100,
                          minSuccDiff=10^(-5))

# format the penalty matrix
lambdaOptVec <- c(rep(lambdaOpt[1], p/2), rep(lambdaOpt[2], p/2))
lambdaOptMat <- outer(lambdaOptVec, lambdaOptVec, "+")

# generalized ridge precision estimate
Phat <- ridgePgen(S, lambdaOptMat, matrix(0, p, p))
```

`optPenaltyPmultiT.kCVauto`

Automatic search for optimal penalty parameter (ridge precision with multi-targets).

Description

Function that determines the optimal penalty parameters through maximization of the k-fold cross-validated log-likelihood score, assuming that variates are grouped and penalized group-wise.

Usage

```
optPenaltyPmultiT.kCVauto(Y, lambdaMin, lambdaMax,
                          lambdaInit=(lambdaMin+lambdaMax)/2,
                          fold=nrow(Y), targetList)
```

Arguments

`Y`	Data `matrix` with samples as rows and variates as columns.
`lambdaMin`	A `numeric` giving the minimum value for the penalty parameters. One value per target. Values should be specified in the same order as the target's appearance in `targetList`.
`lambdaMax`	A `numeric` giving the maximum value for the penalty parameters. One value per group. Values should be specified in the same order as the target's appearance in `targetList`.
`lambdaInit`	A `numeric` giving the initial (starting) value for the penalty parameters. One value per group. Values should be specified in the same order as the target's appearance in `targetList`.
`fold`	A `numeric` or `integer` specifying the number of folds to apply in the cross-validation.
`targetList`	A list of semi-positive definite target matrices towards which the precision matrix is potentially shrunken.

Value

The function returns a `numeric` containing the cross-validated optimal positive penalty parameters.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N., Stam, K.A., Peeters, C.F.W., van de Wiel, M.A. (2020), "Updating of the Gaussian graphical model through targeted penalized estimation", *Journal of Multivariate Analysis*, 178, Article 104621.

See Also

`ridgePmultiT`

Examples

```
# set dimension and sample size
p <- 10
n <- 10

# specify vector of penalty parameters
lambda       <- c(2, 1)

# generate precision matrix
T1       <- matrix(0.7, p, p)
diag(T1) <- 1
T2       <- diag(rep(2, p))

# generate precision matrix
Omega       <- matrix(0.4, p, p)
diag(Omega) <- 2
Sigma       <- solve(Omega)

# data
Y <- mvtnorm::rmvnorm(n, mean=rep(0,p), sigma=Sigma)
S <- cov(Y)

# find optimal penalty parameters through cross-validation
lambdaOpt <- optPenaltyPmultiT.kCVauto(Y, rep(10^(-10), 2),
                          rep(10^(10), 2), rep(1, 2),
                          targetList=list(T1=T1, T2=T2))

# unpenalized diagonal estimate
Phat <- ridgePmultiT(S, lambdaOpt, list(T1=T1, T2=T2))
```

`optPenaltyPrep.kCVauto` *Automatic search for optimal penalty parameters (for precision estimation of data with replicates).*

Description

Function that performs an automatic search of the optimal penalty parameter for the `ridgePrep` call by employing either the Nelder-Mead or quasi-Newton method to calculate the cross-validated (negative) log-likelihood score.

Usage

```
optPenaltyPrep.kCVauto(Y, ids, lambdaInit,
                       fold=nrow(Y), CVcrit,
                       splitting="stratified",
                       targetZ=matrix(0, ncol(Y), ncol(Y)),
                       targetE=matrix(0, ncol(Y), ncol(Y)),
                       nInit=100, minSuccDiff=10^(-10))
```

Arguments

`Y`	Data `matrix` with samples (including the repetitions) as rows and variates as columns.
`ids`	A `numeric` indicating which rows of `Y` belong to the same individal.
`lambdaInit`	A `numeric` giving the initial (starting) values for the two penalty parameters.
`fold`	A `numeric` or `integer` specifying the number of folds to apply in the cross-validation.
`CVcrit`	A `character` with the cross-validation criterion to applied. Either `CVcrit="LL"` (the loglikelihood) or `CVcrit="Qloss"` (the quadratic loss).
`splitting`	A `character`, either `splitting="replications"`, `splitting="samples"`, or `splitting="stratified"`, specifying either how the splits are to be formed: either replications or samples are randomly divided over the `fold` splits (first two options, respectively), or samples are randomly divided over the `fold` splits but in a stratified manner such that the total number of replicates in each group is roughly comparable.
`targetZ`	A semi-positive definite target `matrix` towards which the signal precision matrix estimate is shrunken.
`targetE`	A semi-positive definite target `matrix` towards which the error precision matrix estimate is shrunken.
`nInit`	A `numeric` specifying the number of iterations.
`minSuccDiff`	A `numeric`: minimum successive difference (in terms of the relative change in the absolute difference of the penalized loglikelihood) between two succesive estimates to be achieved.

Value

The function returns a all-positive `numeric`, the cross-validated optimal penalty parameters.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N., Chen, Y. (2019), "Penalized estimation of the Gaussian graphical model from data with replicates", *submitted.*

See Also

`ridgePrep`

Examples

```
# set parameters
p        <- 10
Se       <- diag(runif(p))
Sz       <- matrix(3, p, p)
```

```
diag(Sz) <- 4

# draw data
n <- 100
ids <- numeric()
Y   <- numeric()
for (i in 1:n){
     Ki <- sample(2:5, 1)
     Zi <- mvtnorm::rmvnorm(1, sigma=Sz)
     for (k in 1:Ki){
          Y   <- rbind(Y, Zi + mvtnorm::rmvnorm(1, sigma=Se))
          ids <- c(ids, i)
     }
}

# find optimal penalty parameters
optLambdas <- optPenaltyPrep.kCVauto(Y, ids,
                                     lambdaInit=c(1,1),
                                     fold=nrow(Y),
                                     CVcrit="LL")

# estimate the precision matrices
Ps <- ridgePrep(Y, ids, optLambdas[1], optLambdas[2])
```

`optPenaltyPrepEdiag.kCVauto` *Automatic search for optimal penalty parameters (for precision estimation of data with replicates).*

Description

Function that performs an automatic search of the optimal penalty parameter for the `ridgePrepEdiag` call by employing either the Nelder-Mead or quasi-Newton method to calculate of the cross-validated (negative) log-likelihood score.

Usage

```
optPenaltyPrepEdiag.kCVauto(Y, ids, lambdaInit,
                            fold=nrow(Y), CVcrit,
                            splitting="stratified",
                            targetZ=matrix(0, ncol(Y), ncol(Y)),
                            nInit=100, minSuccDiff=10^(-10))
```

Arguments

`Y`	Data `matrix` with samples (including the repetitions) as rows and variates as columns.
`ids`	A `numeric` indicating which rows of `Y` belong to the same individal.

lambdaInit	A numeric giving the initial (starting) values for the two penalty parameters.
fold	A numeric or integer specifying the number of folds to apply in the cross-validation.
CVcrit	A character with the cross-validation criterion to applied. Either CVcrit="LL" (the loglikelihood) or CVcrit="Qloss" (the quadratic loss).
splitting	A character, either splitting="replications", splitting="samples", or splitting="stratified", specifying either how the splits are to be formed: either replications or samples are randomly divided over the fold splits (first two options, respectively), or samples are randomly divided over the fold splits but in stratified manner such that the total number of replications in each group is roughly comparable.
targetZ	A semi-positive definite target matrix towards which the signal precision matrix estimate is shrunken.
nInit	A numeric specifying the number of iterations.
minSuccDiff	A numeric: minimum successive difference (in terms of the relative change in the absolute difference of the penalized loglikelihood) between two succesive estimates to be achieved.

Value

The function returns a all-positive numeric, the cross-validated optimal penalty parameters.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N., Chen, Y. (2019), "Penalized estimation of the Gaussian graphical model from data with replicates", *submitted*.

See Also

ridgePrepEdiag

Examples

```
# set parameters
p        <- 10
Se       <- diag(runif(p))
Sz       <- matrix(3, p, p)
diag(Sz) <- 4

# draw data
n <- 100
ids <- numeric()
Y   <- numeric()
for (i in 1:n){
     Ki <- sample(2:5, 1)
```

```
        Zi <- mvtnorm::rmvnorm(1, sigma=Sz)
        for (k in 1:Ki){
             Y   <- rbind(Y, Zi + mvtnorm::rmvnorm(1, sigma=Se))
             ids <- c(ids, i)
        }
}

# find optimal penalty parameters
optLambdas <- optPenaltyPrepEdiag.kCVauto(Y, ids,
                                          lambdaInit=c(1,1),
                                          fold=nrow(Y),
                                          CVcrit="LL")

# estimate the precision matrices
Ps <- ridgePrepEdiag(Y, ids, optLambdas[1], optLambdas[2])
```

ridgeGGMmixture *Ridge penalized estimation of a mixture of GGMs.*

Description

Function that estimates a mixture of GGMs (Gaussian graphical models) through a ridge penalized EM (Expectation-Maximization) algorithm as described in Aflakparast *et al.* (2018).

Usage

```
ridgeGGMmixture(Y, K, lambda, target,
                 iWeights=matrix(sample(seq(0+1/nrow(Y),
                                  1-1/nrow(Y), by=1/(2*nrow(Y))),
                                  nrow(Y)*K, replace=TRUE),
                                  nrow=nrow(Y), ncol=K),
                 nInit=100, minSuccDiff=10^(-10),
                 minMixProp=0.01)
```

Arguments

Y	Data `matrix` with samples as rows and variates as columns.
K	A `numeric`, specifying the number of mixture components.
lambda	A positive `numeric` representing the ridge penalty parameter.
target	A semi-positive definite target `matrix` towards which the estimate is shrunken.
iWeights	Sample-specific positive component weight `matrix`. Rows correspond to samples, while columns to components.
nInit	A `numeric` specifying the number of iterations.
minSuccDiff	A `numeric`: minimum successive difference (in terms of their penalized loglikelihood) between two succesive estimates to be achieved.
minMixProp	Smallest mixing probability tolerated.

Details

The data are assumed to follow a mixture of K Gaussian graphical models:

$$\mathbf{Y}_i \sim \sum\nolimits_{k=1}^{K} \theta_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Omega}_k^{-1}),$$

where $\theta_k = P(Z_i = k)$ is the probability that the i-th sample stems from the k-the component. The model parameters are estimated by ridge penalized likelihood maximization:

$$\sum\nolimits_{i=1}^{n} \log[\sum\nolimits_{k=1}^{K} \theta_k P(\mathbf{Y}_i \,|\, Z_i = k; \boldsymbol{\mu}_k, \boldsymbol{\Omega}_k)] + \lambda \sum\nolimits_{k=1}^{K} \|\boldsymbol{\Omega}_k - \mathbf{T}_k\|_F^2,$$

where λ is the penalty parameter and $\mathbf{T}_k$ is the shrinkage target of the k-th component's precision matrix. This function yields the maximizer of this penalized loglikelihood, which is found by means of a penalized EM algorithm.

Value

The function returns a regularized inverse covariance `list`-object with slots:

`mu`	A `matrix` with estimated mean vectors are rows.
`P`	A `matrix` with estimated mixture precision matrices stacked on top of each other.
`pi`	A `numeric` wth estimated mixing probabilities.
`weights`	A `matrix` wth estimated component memberships.
`penLL`	A `numeric` with the penalized loglikelihood of the estimated model.

Note

The elements of `iWeights` may be larger than one as they are rescaled internally to sum to one.

Author(s)

W.N. van Wieringen, M. Aflakparast.

References

Aflakparast, M., de Gunst, M.C.M., van Wieringen, W.N. (2018), "Reconstruction of molecular network evolution from cross-sectional omics data", *Biometrical Journal*, 60(3), 547-563.

See Also

`optPenaltyGGMmixture.kCVauto`

Examples

```
# define mixing proportions
pis <- c(0.2, 0.3, 0.4)

# set dimension and sample size
p <- 5
n <- 100

# define population covariance matrices
diags       <- list(rep(1, p),
                    rep(0.5, p-1),
                    rep(0.25, p-2),
                    rep(0.1, p-3))
Omega       <- as.matrix(Matrix::bandSparse(p,
                                            k=-c(0:3),
                                            diag=c(diags),
                                            symm=TRUE))
Sigma1      <- solve(Omega)
Omega       <- matrix(0.3, p, p)
diag(Omega) <- 1
Sigma2      <- solve(Omega)
Sigma3      <- cov(matrix(rnorm(p*n), ncol=p))

# mean vectors
mean1 <- rep(0,p)
mean2 <- rexp(p)
mean3 <- rnorm(p)

# draw data data from GGM mixture
Z <- sort(sample(c(1:3), n, prob=pis, replace=TRUE))
Y <- rbind(mvtnorm::rmvnorm(sum(Z==1), mean=mean1, sigma=Sigma1),
           mvtnorm::rmvnorm(sum(Z==2), mean=mean2, sigma=Sigma2),
           mvtnorm::rmvnorm(sum(Z==3), mean=mean3, sigma=Sigma3))

# find optimal penalty parameter
optLambda <- optPenaltyGGMmixture.kCVauto(Y,  K,
                                          0.00001, 100,
                                          10, fold=5,
                                          target=0*Sigma1)

# ridge penalized estimation of the GGM mixture
ridgeGGMmixFit <- ridgeGGMmixture(Y, 3, 1, target=0*Sigma1)
```

`ridgePgen` *Ridge estimation of the inverse covariance matrix with element-wise penalization and shrinkage.*

Description

Function that evaluates the generalized ridge estimator of the inverse covariance matrix with element-wise penalization and shrinkage.

Usage

```
ridgePgen(S, lambda, target, nInit=100, minSuccDiff=10^(-10))
```

Arguments

`S`	Sample covariance `matrix`.
`lambda`	A symmetric `matrix` with element-wise positive penalty parameters.
`target`	A semi-positive definite target `matrix` towards which the estimate is shrunken.
`nInit`	A `numeric` specifying the number of iteration.
`minSuccDiff`	A `numeric`: minimum distance between two succesive estimates to be achieved.

Details

This function generalizes the `ridgeP`-function in the sense that, besides element-wise shrinkage, it allows for element-wise penalization in the estimation of the precision matrix of a zero-mean multivariate normal distribution. Hence, it assumes that the data stem from $\mathcal{N}(\mathbf{0}_p, \mathbf{\Omega}^{-1})$. The estimator maximizes the following penalized loglikelihood:

$$\log(|\mathbf{\Omega}|) - \text{tr}(\mathbf{\Omega}\mathbf{S}) - \|\mathbf{\Lambda} \circ (\mathbf{\Omega} - \mathbf{T})\|_F^2,$$

where $\mathbf{S}$ the sample covariance matrix, $\mathbf{\Lambda}$ a symmetric, positive matrix of penalty parameters, the $\circ$-operator represents the Hadamard or element-wise multipication, and $\mathbf{T}$ the precision matrix' shrinkage target. For more details see van Wieringen (2019).

Value

The function returns a regularized inverse covariance `matrix`.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N. (2019), "The generalized ridge estimator of the inverse covariance matrix", *Journal of Computational and Graphical Statistics*, 28(4), 932-942.

See Also

`ridgeP`, `ridgePchordal`.

Examples

```
# set dimension and sample size
p <- 10
n <- 10

# penalty parameter matrix
lambda       <- matrix(1, p, p)
```

```
diag(lambda) <- 0.1

# generate precision matrix
Omega       <- matrix(0.4, p, p)
diag(Omega) <- 1
Sigma       <- solve(Omega)

# data
Y <- mvtnorm::rmvnorm(n, mean=rep(0,p), sigma=Sigma)
S <- cov(Y)

# unpenalized diagonal estimate
Phat <- ridgePgen(S, lambda, 0*S)
```

---

ridgePgen.kCV *K-fold cross-validated loglikelihood of ridge precision estimator.*

Description

Function that calculates of the k-fold cross-validated negative (!) loglikelihood of the generalized ridge precision estimator.

Usage

```
ridgePgen.kCV(lambda, Y, fold=nrow(Y), target,
               nInit=100, minSuccDiff=10^(-5))
```

Arguments

`lambda`	A symmetric `matrix` with element-wise positive penalty parameters.
`Y`	Data `matrix` with samples as rows and variates as columns.
`fold`	A `numeric` or `integer` specifying the number of folds to apply in the cross-validation.
`target`	A semi-positive definite target `matrix` towards which the estimate is shrunken.
`nInit`	A `numeric` specifying the number of iterations.
`minSuccDiff`	A `numeric`: minimum successive difference (in terms of their penalized loglikelihood) between two succesive estimates to be achieved.

Value

The function returns a `numeric` containing the cross-validated negative loglikelihood.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N. (2019), "The generalized ridge estimator of the inverse covariance matrix", *Journal of Computational and Graphical Statistics*, 28(4), 932-942.

See Also

ridgePgen

Examples

```
# set dimension and sample size
p <- 10
n <- 10

# penalty parameter matrix
lambda       <- matrix(1, p, p)
diag(lambda) <- 0.1

# generate precision matrix
Omega       <- matrix(0.4, p, p)
diag(Omega) <- 1
Sigma       <- solve(Omega)

# data
Y <- mvtnorm::rmvnorm(n, mean=rep(0,p), sigma=Sigma)
S <- cov(Y)

# find optimal penalty parameters through cross-validation
lambdaOpt <- optPenaltyPgen.kCVauto.banded(Y, 10^(-10), 10^(10),
                          target=matrix(0, p, p),
                          penalize.diag=FALSE, nInit=100,
                          minSuccDiff=10^(-5))

# format the penalty matrix
lambdaOptMat <- matrix(NA, p, p)
for (j1 in 1:p){
    for (j2 in 1:p){
        lambdaOptMat[j1, j2] <- lambdaOpt * (abs(j1-j2)+1)
    }
}

# generalized ridge precision estimate
Phat <- ridgePgen(S, lambdaOptMat, matrix(0, p, p))
```

`ridgePgen.kCV.banded` *K-fold cross-validated loglikelihood of ridge precision estimator for banded precisions.*

Description

Function that calculates of the k-fold cross-validated negative (!) loglikelihood of the generalized ridge precision estimator, with a penalization that encourages a banded precision matrix.

Usage

```
ridgePgen.kCV.banded(lambda, Y, fold=nrow(Y), target,
                     zeros=matrix(nrow=0, ncol=2),
                     penalize.diag=TRUE, nInit=100,
                     minSuccDiff=10^(-5))
```

Arguments

`lambda`	A `numeric` with the penalty parameter value.
`Y`	Data `matrix` with samples as rows and variates as columns.
`fold`	A `numeric` or `integer` specifying the number of folds to apply in the cross-validation.
`target`	A semi-positive definite target `matrix` towards which the estimate is shrunken.
`zeros`	A two-column `matrix`, with the first and second column containing the row- and column-index of zero precision elements.
`penalize.diag`	A `logical` indicating whether the diagonal should be penalized.
`nInit`	A `numeric` specifying the number of iterations.
`minSuccDiff`	A `numeric`: minimum successive difference (in terms of their penalized loglikelihood) between two succesive estimates to be achieved.

Details

The penalty matrix $\mathbf{\Lambda}$ is parametrized as follows. The elements of $\mathbf{\Lambda}$ are $(\mathbf{\Lambda})_{j,j'} = \lambda(|j - j'| + 1)$ for $j, j' = 1, \ldots, p$.

Value

The function returns a `numeric` containing the cross-validated negative loglikelihood.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N. (2019), "The generalized ridge estimator of the inverse covariance matrix", *Journal of Computational and Graphical Statistics*, 28(4), 932-942.

See Also

`ridgePgen`

Examples

```
# set dimension and sample size
p <- 10
n <- 10

# penalty parameter matrix
lambda       <- matrix(1, p, p)
diag(lambda) <- 0.1

# generate precision matrix
Omega       <- matrix(0.4, p, p)
diag(Omega) <- 1
Sigma       <- solve(Omega)

# data
Y <- mvtnorm::rmvnorm(n, mean=rep(0,p), sigma=Sigma)
S <- cov(Y)

# find optimal penalty parameters through cross-validation
lambdaOpt <- optPenaltyPgen.kCVauto.banded(Y, 10^(-10), 10^(10),
                          target=matrix(0, p, p),
                          penalize.diag=FALSE, nInit=100,
                          minSuccDiff=10^(-5))

# format the penalty matrix
lambdaOptMat <- matrix(NA, p, p)
for (j1 in 1:p){
    for (j2 in 1:p){
        lambdaOptMat[j1, j2] <- lambdaOpt * (abs(j1-j2)+1)
    }
}

# generalized ridge precision estimate
Phat <- ridgePgen(S, lambdaOptMat, matrix(0, p, p))
```

`ridgePgen.kCV.groups` *K-fold cross-validated loglikelihood of ridge precision estimator with group-wise penalized variates.*

Description

Function that calculates of the k-fold cross-validated negative (!) loglikelihood of the generalized ridge precision estimator, assuming that variates are grouped and penalized group-wise.

Usage

```
ridgePgen.kCV.groups(lambdaGrps, Y, fold=nrow(Y),
                       groups, target,
                       zeros=matrix(nrow=0, ncol=2),
                       penalize.diag=TRUE, nInit=100,
                       minSuccDiff=10^(-5))
```

Arguments

`lambdaGrps`	A `numeric` with penalty parameter values, one per group. Values should be specified in the same order as the first appearance of a group representative.
`Y`	Data `matrix` with samples as rows and variates as columns.
`fold`	A `numeric` or `integer` specifying the number of folds to apply in the cross-validation.
`groups`	A`numeric` indicating to which group a variate belongs. Same values indicate same group.
`target`	A semi-positive definite target `matrix` towards which the estimate is shrunken.
`zeros`	A two-column `matrix`, with the first and second column containing the row- and column-index of zero precision elements.
`penalize.diag`	A `logical` indicating whether the diagonal should be penalized.
`nInit`	A `numeric` specifying the number of iterations.
`minSuccDiff`	A `numeric`: minimum successive difference (in terms of their penalized loglikelihood) between two succesive estimates to be achieved.

Details

The penalty matrix $\mathbf{\Lambda}$ is parametrized as follows. The elements of $\mathbf{\Lambda}$ are $(\mathbf{\Lambda})_{j,j'} = \frac{1}{2}(\lambda_k + \lambda_{k'})$ for $j, j' = 1, \ldots, p$ if j and j' belong to groups k and k', respectively, where λ_k and $\lambda_{k'}$ are the corresponding group-specific penalty parameters.

Value

The function returns a `numeric` containing the cross-validated negative loglikelihood.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N. (2019), "The generalized ridge estimator of the inverse covariance matrix", *Journal of Computational and Graphical Statistics*, 28(4), 932-942.

See Also

`ridgePgen`

Examples

```
# set dimension and sample size
p <- 10
n <- 10

# penalty parameter matrix
lambda       <- matrix(1, p, p)
diag(lambda) <- 0.1

# generate precision matrix
Omega       <- matrix(0.4, p, p)
diag(Omega) <- 1
Sigma       <- solve(Omega)

# data
Y <- mvtnorm::rmvnorm(n, mean=rep(0,p), sigma=Sigma)
S <- cov(Y)

# find optimal penalty parameters through cross-validation
lambdaOpt <- optPenaltyPgen.kCVauto.groups(Y, rep(10^(-10), 2), rep(10^(10), 2),
                          groups=c(rep(0, p/2), rep(1, p/2)),
                          target=matrix(0, p, p),
                          penalize.diag=FALSE, nInit=100,
                          minSuccDiff=10^(-5))

# format the penalty matrix
lambdaOptVec <- c(rep(lambdaOpt[1], p/2), rep(lambdaOpt[2], p/2))
lambdaOptMat <- outer(lambdaOptVec, lambdaOptVec, "+")

# generalized ridge precision estimate
Phat <- ridgePgen(S, lambdaOptMat, matrix(0, p, p))
```

`ridgePmultiT` *Ridge estimation of the inverse covariance matrix with multi-target shrinkage.*

Description

Function that evaluates the ridge estimator of the inverse covariance matrix with multi-target shrinkage.

Usage

```
ridgePmultiT(S, lambda, targetList)
```

Arguments

`S`	Sample covariance `matrix`.

`lambda`	A `numeric` of positive penalty parameters. Values should be specified in the same order as the target's appearance in `targetList`.
`targetList`	A list of semi-positive definite target matrices towards which the precision matrix is potentially shrunken.

Details

This function generalizes the `ridgeP`-function in the sense that multiple shrinkage targets can be provided in the estimation of the precision matrix of a zero-mean multivariate normal distribution. Hence, it assumes that the data stem from $\mathcal{N}(\mathbf{0}_p, \mathbf{\Omega}^{-1})$. The estimator maximizes the following penalized loglikelihood:

$$\log(|\mathbf{\Omega}|) - \mathrm{tr}(\mathbf{\Omega}\mathbf{S}) - \sum_{g=1}^{G} \lambda_g \|\mathbf{\Omega} - \mathbf{T}_g\|_F^2,$$

where $\mathbf{S}$ the sample covariance matrix, $\{\lambda_g\}_{g=1}^{G}$ the penalty parameters of each target matrix, and the $\{\mathbf{T}_g\}_{g=1}^{G}$ the precision matrix' shrinkage targets. For more details see van Wieringen *et al.* (2020).

Value

The function returns a regularized inverse covariance `matrix`.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N., Stam, K.A., Peeters, C.F.W., van de Wiel, M.A. (2020), "Updating of the Gaussian graphical model through targeted penalized estimation", *Journal of Multivariate Analysis*, 178, Article 104621.

See Also

`ridgeP`, `ridgePchordal`.

Examples

```
# set dimension and sample size
p <- 10
n <- 10

# specify vector of penalty parameters
lambda       <- c(2, 1)

# generate precision matrix
T1       <- matrix(0.7, p, p)
diag(T1) <- 1
T2       <- diag(rep(2, p))
```

```
# generate precision matrix
Omega       <- matrix(0.4, p, p)
diag(Omega) <- 2
Sigma       <- solve(Omega)

# data
Y <- mvtnorm::rmvnorm(n, mean=rep(0,p), sigma=Sigma)
S <- cov(Y)

# unpenalized diagonal estimate
Phat <- ridgePmultiT(S, lambda, list(T1=T1, T2=T2))
```

---

ridgePrep *Ridge penalized estimation of the precision matrix from data with replicates.*

Description

Estimation of the precision matrix from data with replicates through a ridge penalized EM (Expectation-Maximization) algorithm. It assumes a simple 'signal+noise' model, both random variables are assumed to be drawn from a multivariate normal distribution with their own unstructured precision matrix. These precision matrices are estimated.

Usage

```
ridgePrep(Y, ids, lambdaZ, lambdaE,
          targetZ=matrix(0, ncol(Y), ncol(Y)),
          targetE=matrix(0, ncol(Y), ncol(Y)),
          nInit=100, minSuccDiff=10^(-10))
```

Arguments

`Y`	Data `matrix` with samples (including the repetitions) as rows and variates as columns.
`ids`	A `numeric` indicating which rows of `Y` belong to the same individal.
`lambdaZ`	A positive `numeric` representing the ridge penalty parameter for the signal precision matrix estimate.
`lambdaE`	A positive `numeric` representing the ridge penalty parameter for the error precision matrix estimate.
`targetZ`	A semi-positive definite target `matrix` towards which the signal precision matrix estimate is shrunken.
`targetE`	A semi-positive definite target `matrix` towards which the error precision matrix estimate is shrunken.
`nInit`	A `numeric` specifying the number of iterations.
`minSuccDiff`	A `numeric`: minimum successive difference (in terms of the relative change in the absolute difference of the penalized loglikelihood) between two succesive estimates to be achieved.

Details

Data are assumed to originate from a design with replicates. Each observation $\mathbf{Y}_{i,k_i}$ with k_i ($k_i = 1, \ldots, K_i$) the k_i-th replicate of the i-th sample, is described by a 'signal+noise' model: $\mathbf{Y}_{i,k_i} = \mathbf{Z}_i + \varepsilon_{i,k_i}$, where $\mathbf{Z}_i$ and ε_{i,k_i} represent the signal and noise, respectively. Each observation $\mathbf{Y}_{i,k_i}$ follows a multivariate normal law of the form $\mathbf{Y}_{i,k_i} \sim \mathcal{N}(\mathbf{0}_p, \mathbf{\Omega}_z^{-1} + \mathbf{\Omega}_\varepsilon^{-1})$, which results from the distributional assumptions of the signal and the noise, $\mathbf{Z}_i \sim \mathcal{N}(\mathbf{0}_p, \mathbf{\Omega}_z^{-1})$ and $\varepsilon_{i,k_i} \sim \mathcal{N}(\mathbf{0}_p, \mathbf{\Omega}_\varepsilon^{-1})$, and their independence. The model parameters are estimated by means of a penalized EM algorithm that maximizes the loglikelihood augmented with the penalty $\lambda_z \|\mathbf{\Omega}_z - \mathbf{T}_z\|_F^2 + \lambda_\varepsilon \|\mathbf{\Omega}_\varepsilon - \mathbf{T}_\varepsilon\|_F^2$, in which $\mathbf{T}_z$ and $\mathbf{T}_\varepsilon$ are the shrinkage targets of the signal and noise precision matrices, respectively. For more details see van Wieringen and Chen (2019).

Value

The function returns the regularized inverse covariance `list`-object with slots:

`Pz`	The estimated signal precision matrix.
`Pz`	The estimated error precision matrix.
`penLL`	The penalized loglikelihood of the estimated model.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N., Chen, Y. (2019), "Penalized estimation of the Gaussian graphical model from data with replicates", *submitted*.

See Also

`optPenaltyPrep.kCVauto`

Examples

```
# set parameters
p        <- 10
Se       <- diag(runif(p))
Sz       <- matrix(3, p, p)
diag(Sz) <- 4

# draw data
n <- 100
ids <- numeric()
Y   <- numeric()
for (i in 1:n){
     Ki <- sample(2:5, 1)
     Zi <- mvtnorm::rmvnorm(1, sigma=Sz)
     for (k in 1:Ki){
          Y   <- rbind(Y, Zi + mvtnorm::rmvnorm(1, sigma=Se))
          ids <- c(ids, i)
```

```
        }
    }

    # estimate
    Ps <- ridgePrep(Y, ids, 1, 1)
```

ridgePrepEdiag — *Ridge penalized estimation of the precision matrix from data with replicates.*

Description

Estimation of precision matrices from data with replicates through a ridge penalized EM (Expectation-Maximization) algorithm. It assumes a simple 'signal+noise' model, both random variables are assumed to be drawn from a multivariate normal distribution with their own precision matrix. The signal precision matrix is unstructured, while the former is diagonal. These precision matrices are estimated.

Usage

```
ridgePrepEdiag(Y, ids, lambdaZ,
                 targetZ=matrix(0, ncol(Y), ncol(Y)),
                 nInit=100, minSuccDiff=10^(-10))
```

Arguments

`Y`	Data `matrix` with samples (including the repetitions) as rows and variates as columns.
`ids`	A `numeric` indicating which rows of `Y` belong to the same individal.
`lambdaZ`	A positive `numeric` representing the ridge penalty parameter for the signal precision matrix estimate.
`targetZ`	A semi-positive definite target `matrix` towards which the signal precision matrix estimate is shrunken.
`nInit`	A `numeric` specifying the number of iterations.
`minSuccDiff`	A `numeric`: minimum successive difference (in terms of the relative change in the absolute difference of the penalized loglikelihood) between two succesive estimates to be achieved.

Details

Data are assumed to originate from a design with replicates. Each observation $\mathbf{Y}_{i,k_i}$ with k_i ($k_i = 1, \ldots, K_i$) the k_i-th replicate of the i-th sample, is described by a 'signal+noise' model: $\mathbf{Y}_{i,k_i} = \mathbf{Z}_i + \varepsilon_{i,k_i}$, where $\mathbf{Z}_i$ and ε_{i,k_i} represent the signal and noise, respectively. Each observation $\mathbf{Y}_{i,k_i}$ follows a multivariate normal law of the form $\mathbf{Y}_{i,k_i} \sim \mathcal{N}(\mathbf{0}_p, \mathbf{\Omega}_z^{-1} + \mathbf{\Omega}_\varepsilon^{-1})$, which results from the distributional assumptions of the signal and the noise, $\mathbf{Z}_i \sim \mathcal{N}(\mathbf{0}_p, \mathbf{\Omega}_z^{-1})$ and $\varepsilon_{i,k_i} \sim \mathcal{N}(\mathbf{0}_p, \mathbf{\Omega}_\varepsilon^{-1})$ with $\mathbf{\Omega}_\varepsilon$ diagonal, and their independence. The model parameters are estimated by means of a penalized EM algorithm that maximizes the loglikelihood augmented with the penalty $\lambda_z \| \mathbf{\Omega}_z - \mathbf{T}_z \|_F^2$, in which $\mathbf{T}_z$ is the shrinkage target of the signal precision matrix. For more details see van Wieringen and Chen (2019).

Value

The function returns the regularized inverse covariance `list`-object with slots:

`Pz`	The estimated signal precision matrix.
`Pe`	The estimated error precision matrix.
`penLL`	The penalized loglikelihood of the estimated model.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N., Chen, Y. (2019), "Penalized estimation of the Gaussian graphical model from data with replicates", *submitted*.

See Also

`optPenaltyPrepEdiag.kCVauto`

Examples

```
# set parameters
p        <- 10
Se       <- diag(runif(p))
Sz       <- matrix(3, p, p)
diag(Sz) <- 4

# draw data
n <- 100
ids <- numeric()
Y   <- numeric()
for (i in 1:n){
     Ki <- sample(2:5, 1)
     Zi <- mvtnorm::rmvnorm(1, sigma=Sz)
     for (k in 1:Ki){
          Y   <- rbind(Y, Zi + mvtnorm::rmvnorm(1, sigma=Se))
          ids <- c(ids, i)
     }
}

# estimate
Ps <- ridgePrepEdiag(Y, ids, 1)
```

Index