Package: porridge (via r-universe)

October 6, 2026

Type Package

Title Ridge-Type Penalized Estimation of a Potpourri of Models

Version 0.3.3

Date 2024-02-21

Author Wessel N. van Wieringen [aut, cre]
(<https://orcid.org/0000-0002-5100-9123>), 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, 2021) or ii) from
non-replicated data by means of either a ridge estimator with
multiple shrinkage targets (as presented in van Wieringen et
al. 2020, <doi:10.1016/j.jmva.2020.104621>) or 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: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). On another note, the package also
includes functionality for ridge-type estimation of the
generalized linear model (as presented in van Wieringen,
Binder, 2022, <doi:10.1080/10618600.2022.2035231>).

License GPL (>= 2)

LazyLoad yes

URL https://www.math.vu.nl/~wvanwie/

Depends R (>= 3.5.0)

Imports MASS, Matrix, stats, mvtnorm, Rcpp, methods, pracma

Suggests rags2ridges

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository https://cran.r-universe.dev

Date/Publication 2024-02-22 02:30:32 UTC

RemoteUrl https://github.com/cran/porridge

RemoteRef HEAD

RemoteSha 928649f5e7d4453cf7a0b4eeef0eb0906dd51f41

Contents

porridge-package . . . 3
genRidgePenaltyMat . . . 4
makeFoldsGLMcv . . . 6
optPenaltyGGMmixture.kCVauto . . . 7
optPenaltyGLM.kCVauto . . . 9
optPenaltyGLMmultiT.kCVauto . . . 11
optPenaltyPgen.kCVauto.banded . . . 13
optPenaltyPgen.kCVauto.groups . . . 15
optPenaltyPmultiT.kCVauto . . . 17
optPenaltyPrep.kCVauto . . . 19
optPenaltyPrepEdiag.kCVauto . . . 21
ridgeGGMmixture . . . 23
ridgeGLM . . . 25
ridgeGLMdof . . . 27
ridgeGLMmultiT . . . 29
ridgePgen . . . 30
ridgePgen.kCV . . . 32
ridgePgen.kCV.banded . . . 33
ridgePgen.kCV.groups . . . 35
ridgePmultiT . . . 37
ridgePrep . . . 39
ridgePrepEdiag . . . 41

Index **43**

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.
- Multi-targeted ridge estimation of the precision matrix of a Gaussian graphical model (van Wieringen et al., 2020) through the functions `ridgePmultiT`. This function is complemented by the functions `optPenaltyPmultiT.kCVauto` for penalty parameters selection through K-fold cross-validation.
- Gaussian graphical model estimation from data with replicates in ridge penalized fashion (van Wieringen, Chen, 2021) (`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`.
- Targeted and multi-targeted ridge estimation of the regression parameter of the generalized linear model (van Wieringen, Binder, 2022; van Wieringen, 2021; Lettink et al., 2022) through the functions `ridgeGLM` and `ridgeGLMmultiT`. This function is complemented by the functions `optPenaltyGLM.kCVauto` and `optPenaltyGLMmultiT.kCVauto` for penalty parameters selection through K-fold cross-validation, and the `ridgeGLMdof`-function for the evaluation of the fitted model's degrees of freedom.

Future versions aim to include more ridge-type functionality.

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

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.

Lettink, A., Chinapaw, M.J.M., van Wieringen, W.N. (2022), "Two-dimensional fused targeted ridge regression for health indicator prediction from accelerometer data", *submitted*.

Peeters, C.F.W., Bilgrau, A.E., and van Wieringen, W.N. (2021), "rags2ridges: Ridge Estimation of Precision Matrices from High-Dimensional Data", R package version 2.2.5. https://CRAN.R-project.org/package=rags2ridges.

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. (2021), "Lecture notes on ridge regression", *Arxiv preprint*, arXiv:1509.09169.

van Wieringen W.N., Chen, Y. (2021), "Penalized estimation of the Gaussian graphical model from data with replicates", *Statistics in Medicine*, 40(19), 4279-4293.

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.

van Wieringen, W.N. Binder, H. (2022), "Sequential learning of regression models by penalized estimation", *Journal of Computational and Graphical Statistics*, accepted.

See Also

The porridge-package.

`genRidgePenaltyMat` *Penalty parameter matrix for generalized ridge regression.*

Description

The function produces an unscaled penalty parameter matrix to be used in the generalized ridge regression estimator.

Usage

```
genRidgePenaltyMat(pr, pc=pr, type="2dimA")
```

Arguments

`pr`	A positive `integer`. The number of covariates if `type="common"` or `type="fused1dim"`. Or, the row dimension of a 2-dimensional covariate layout if `type="fused2dimA"` or `type="fused2dimD"`.
`pc`	A positive `integer`. The column dimension of a 2-dimensional covariate layout if `type="fused2dimA"` or `type="fused2dimD"`. Ignored if `type="common"` or `type="fused1dim"`.
`type`	A `character`. Either `"common"`, `"fused1dim"`, `"fused2dimA"` or `"fused2dimD"`, see details.

Details

Various ridge penalty matrices are implemented.

The `type="common"`-option supports the 'homogeneity' ridge penalization proposed by Anatolyev (2020). The ridge penalty matrix $\boldsymbol{\Delta}$ for a p-dimensional regression parameter $\boldsymbol{\beta}$ is such that:

$$\boldsymbol{\beta}^\top \boldsymbol{\Delta} \boldsymbol{\beta} \;=\; \boldsymbol{\beta}^\top (\mathbf{I}_{pp} - p^{-1}\mathbf{1}_{pp})\boldsymbol{\beta} \;=\; \sum\nolimits_{j=1}^{p} \big(\beta_j - p^{-1} \sum\nolimits_{j'=1}^{p} \beta_{j'}\big)^2.$$

This penalty matrix encourages shrinkage of the elements of $\boldsymbol{\beta}$ to a common effect value.

The `type="fused1dim"`-option facilitates the 1-dimensional fused ridge estimation of Goeman (2008). The ridge penalty matrix $\boldsymbol{\Delta}$ for a p-dimensional regression parameter $\boldsymbol{\beta}$ is such that:

$$\boldsymbol{\beta}^\top \boldsymbol{\Delta} \boldsymbol{\beta} \;=\; \sum\nolimits_{j=2}^{p} (\beta_j - \beta_{j-1})_2^2.$$

This penalty matrix aims to shrink contiguous (as defined by their index) elements of $\boldsymbol{\beta}$ towards each other.

The `type="fused2dimA"`- and `type="fused2dimD"`-options facilitate 2-dimensional ridge estimation as proposed by Lettink et al. (2022). It assumes the regression parameter is endowed with a 2-dimensional layout. The columns of this layout have been stacked to form $\boldsymbol{\beta}$. The 2-dimensional fused ridge estimation shrinks elements of $\boldsymbol{\beta}$ that are neighbors in the 2-dimensional layout towards each other. The two options use different notions of neighbors. If `type="fused2dimA"`, the ridge penalty matrix $\boldsymbol{\Delta}$ for a p-dimensional regression parameter $\boldsymbol{\beta}$ is such that:

$$\begin{aligned}\boldsymbol{\beta}^\top \boldsymbol{\Delta} \boldsymbol{\beta} \;=\; & \sum\nolimits_{j_r=1}^{p_r-1} \sum\nolimits_{j_c=1}^{p_c-1} [(\beta_{j_r,j_c+1} - \beta_{j_r,j_c})^2 + (\beta_{j_r+1,j_c} - \beta_{j_r,j_c})^2] \\ & + \sum\nolimits_{j_c=1}^{p_c-1} (\beta_{p_r,j_c+1} - \beta_{p_r,j_c})^2 + \sum\nolimits_{j_r=1}^{p_r-1} (\beta_{j_r+1,p_c} - \beta_{j_r,p_c})^2,\end{aligned}$$

where p_r and p_c are the row and column dimension, respectively, of the 2-dimensional layout. This penalty matrix intends to shrink the elements of $\boldsymbol{\beta}$ along the axes of the 2-dimensional layout. If `type="fused2dimD"`, the ridge penalty matrix $\boldsymbol{\Delta}$ for a p-dimensional regression parameter $\boldsymbol{\beta}$ is such that:

$$\begin{aligned}\boldsymbol{\beta}^\top \boldsymbol{\Delta} \boldsymbol{\beta} \;=\; & \sum\nolimits_{j_r=1}^{p_r-1} \sum\nolimits_{j_c=1}^{p_c-2} [(\beta_{j_r+1,j_c} - \beta_{j_r,j_c+1})^2 + (\beta_{j_r+1,j_c+2} - \beta_{j_r,j_c+1})^2] \\ & + \sum\nolimits_{j_r=1}^{p_r-1} [(\beta_{j_r+1,2} - \beta_{j_r,1})^2 + (\beta_{j_r+1,p_c-1} - \beta_{j_r,p_c})^2].\end{aligned}$$

This penalty matrix shrinks the elements of $\boldsymbol{\beta}$ along the diagonally to the axes of the 2-dimensional layout. The penalty matrices generated by `type="fused2dimA"`- and `type="fused2dimD"`-options may be combined.

Value

The function returns a non-negative definite `matrix`.

Author(s)

W.N. van Wieringen.

References

Anatolyev, S. (2020), "A ridge to homogeneity for linear models", *Journal of Statistical Computation and Simulation*, 90(13), 2455-2472.

Goeman, J.J. (2008), "Autocorrelated logistic ridge regression for prediction based on proteomics spectra", *Statistical Applications in Genetics and Molecular Biology*, 7(2).

Lettink, A, Chinapaw, M.J.M., van Wieringen, W.N. (2022), "Two-dimensional fused targeted ridge regression for health indicator prediction from accelerometer data", *submitted*.

See Also

`ridgeGLM`

Examples

```
# generate unscaled general penalty parameter matrix
Dfused <- genRidgePenaltyMat(10, type="fused1dim")
```

`makeFoldsGLMcv` *Generate folds for cross-validation of generalized linear models.*

Description

Function that evaluates the targeted ridge estimator of the regression parameter of generalized linear models.

Usage

```
makeFoldsGLMcv(fold, Y, stratified=TRUE, model="linear")
```

Arguments

`fold`	An `integer`, the number of folds to be generated.
`Y`	A `numeric` being the response vector.
`stratified`	A `logical`. If `stratified=TRUE`, the folds are generated such the distribution of the response `Y` is (roughly) the same across folds.
`model`	A `character`, either `"linear"` and `"logistic"` (a reference to the models currently implemented), indicative of the type of response for stratification.

Value

A `list` of length `fold`. Each list item is a fold.

Author(s)

W.N. van Wieringen.

Examples

```
# set the sample size
n <- 50

# set the true parameter
betas <- (c(0:100) - 50) / 20

# generate covariate data
X <- matrix(rnorm(length(betas)*n), nrow=n)

# sample the response
probs <- exp(tcrossprod(betas, X)[1,]) / (1 + exp(tcrossprod(betas, X)[1,]))
Y     <- numeric()
for (i in 1:n){
    Y <- c(Y, sample(c(0,1), 1, prob=c(1-probs[i], probs[i])))
}

# generate folds
folds <- makeFoldsGLMcv(10, Y, model="logistic")
```

---

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=3,
###                                           0.00001, 100,
###                                           10, fold=5,
###                                           target=0*Sigma1)

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

`optPenaltyGLM.kCVauto` *Automatic search for optimal penalty parameters of the targeted ridge GLM estimator.*

Description

Function finds the optimal penalty parameter of the targeted ridge regression estimator of the generalized linear model parameter. The optimum is defined as the minimizer of the cross-validated loss associated with the estimator.

Usage

```
optPenaltyGLM.kCVauto(Y, X, U=matrix(ncol=0, nrow=length(Y)), lambdaInit,
                      lambdaGinit=0, Dg=matrix(0, ncol=ncol(X), nrow=ncol(X)),
                      model="linear", target=rep(0, ncol(X)),
                      folds=makeFoldsGLMcv(min(10, length(X)), Y, model=model),
                      loss="loglik", lambdaMin=10^(-5),
                      lambdaGmin=10^(-5), minSuccDiff=10^(-5), maxIter=100,
                      implementation="org")
```

Arguments

Y	A numeric being the response vector.
X	The design matrix. The number of rows should match the number of elements of Y.
U	The design matrix of the unpenalized covariates. The number of rows should match the number of elements of Y.
lambdaInit	A numeric, the initial (starting) values for the regular ridge penalty parameter.
lambdaGinit	A numeric, the initial (starting) values for the generalized ridge penalty parameter.
Dg	A non-negative definite matrix of the unscaled generalized ridge penalty.
model	A character, either "linear" and "logistic" (a reference to the models currently implemented), indicating which generalized linear model model instance is to be fitted
target	A numeric towards which the estimate is shrunken.
folds	A list. Each list item representing a fold. It is an integer vector indexing the samples that comprise the fold. This object can be generated with the makeFoldsGLMcv function.
loss	A character, either loss="loglik" or "sos", specifying loss criterion to be used in the cross-validation. Used only if model="linear".
lambdaMin	A positive numeric, the lower bound of search interval of the regular ridge penalty parameter.
lambdaGmin	A positive numeric larger than lambdaGmin, the upper bound of the search interval of the generalized ridge penalty parameter.
minSuccDiff	A numeric, the minimum distance between the loglikelihoods of two successive iterations to be achieved. Used only if model="logistic".
maxIter	A numeric specifying the maximum number of iterations. Used only if model="logistic".
implementation	A character, either "org" or "alt", specifying the implementation to be used. The implementations (should) only differ in computation efficiency.

Value

The function returns a all-positive numeric, the cross-validated optimal penalty parameters. The average loglikelihood over the left-out samples is used as the cross-validation criterion. If model="linear", also the average sum-of-squares over the left-out samples is offered as cross-validation criterion.

Note

The joint selection of penalty parameters λ and λ_g through the optimization of the cross-validated loss may lead to a locally optimal choice. This is due to the fact that the penalties are to some extent communicating vessels. Both shrink towards the same target, only in slightly (dependending on the specifics of the generalized penalty matrix Δ) different ways. As such, the shrinkage achieved by one penalty may be partially compensated for by the other. This may hamper the algorithm in its search for the global optimizers.

Moreover, the penalized IRLS (Iterative Reweighted Least Squares) algorithm for the evaluation of the generalized ridge logistic regression estimator and implemented in the `ridgeGLM`-function may fail to converge for small penalty parameter values in combination with a nonzero shrinkage target. This phenomenon propogates to the `optPenaltyGLM.kCVauto`-function.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N. Binder, H. (2022), "Sequential learning of regression models by penalized estimation", *accepted*.

Lettink, A., Chinapaw, M.J.M., van Wieringen, W.N. et al. (2022), "Two-dimensional fused targeted ridge regression for health indicator prediction from accelerometer data", *submitted*.

Examples

```
# set the sample size
n <- 50

# set the true parameter
betas <- (c(0:100) - 50) / 20

# generate covariate data
X <- matrix(rnorm(length(betas)*n), nrow=n)

# sample the response
probs <- exp(tcrossprod(betas, X)[1,]) / (1 + exp(tcrossprod(betas, X)[1,]))
Y     <- numeric()
for (i in 1:n){
    Y <- c(Y, sample(c(0,1), 1, prob=c(1-probs[i], probs[i])))
}

# tune the penalty parameter
optLambda <- optPenaltyGLM.kCVauto(Y, X, lambdaInit=1, fold=5,
                                   target=betas/2, model="logistic",
                                   minSuccDiff=10^(-3))

# estimate the logistic regression parameter
bHat <- ridgeGLM(Y, X, lambda=optLambda, target=betas/2, model="logistic")
```

`optPenaltyGLMmultiT.kCVauto` *Automatic search for optimal penalty parameters of the targeted ridge GLM estimator.*

Description

Function finds the optimal penalty parameter of the targeted ridge regression estimator of the generalized linear model parameter. The optimum is defined as the minimizer of the cross-validated loss associated with the estimator.

Usage

```
optPenaltyGLMmultiT.kCVauto(Y, X, lambdaInit, model="linear", targetMat,
                      folds=makeFoldsGLMcv(min(10, length(X)), Y, model=model),
                      loss="loglik", lambdaMin=10^(-5),
                      minSuccDiff=10^(-5), maxIter=100)
```

Arguments

`Y`	A `numeric` being the response vector.
`X`	The design `matrix`. The number of rows should match the number of elements of `Y`.
`lambdaInit`	A `numeric` giving the starting values for search of the optimal penalty parameter.
`model`	A `character`, either `"linear"` and `"logistic"` (a reference to the models currently implemented), indicating which generalized linear model model instance is to be fitted.
`targetMat`	A `matrix` with targets for the regression parameter as columns.
`folds`	A `list`. Each list item representing a fold. It is an `integer` vector indexing the samples that comprise the fold. This object can be generated with the `makeFoldsGLMcv` function.
`loss`	A `character`, either `loss="loglik"` or `"sos"`, specifying loss criterion to be used in the cross-validation. Used only if `model="linear"`.
`lambdaMin`	A positive `numeric`, the lower bound of search interval of the regular ridge penalty parameter.
`minSuccDiff`	A `numeric`, the minimum distance between the loglikelihoods of two successive iterations to be achieved. Used only if `model="logistic"`.
`maxIter`	A `numeric` specifying the maximum number of iterations. Used only if `model="logistic"`.

Value

The function returns an all-positive `numeric`, the cross-validated optimal penalty parameters. The average loglikelihood over the left-out samples is used as the cross-validation criterion. If `model="linear"`, also the average sum-of-squares over the left-out samples is offered as cross-validation criterion.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N. Binder, H. (2022), "Sequential learning of regression models by penalized estimation", *accepted*.

Examples

```
# set the sample size
n <- 50

# set the true parameter
betas <- (c(0:100) - 50) / 20

# generate covariate data
X <- matrix(rnorm(length(betas)*n), nrow=n)

# sample the response
probs <- exp(tcrossprod(betas, X)[1,]) / (1 + exp(tcrossprod(betas, X)[1,]))
Y     <- numeric()
for (i in 1:n){
    Y <- c(Y, sample(c(0,1), 1, prob=c(1-probs[i], probs[i])))
}

# create targets
targets <- cbind(betas/2, rep(0, length(betas)))

# tune the penalty parameter
### optLambdas <- optPenaltyGLMmultiT.kCVauto(Y, X, c(50,0.1), fold=5,
###                                           targetMat=targets, model="logistic",
###                                           minSuccDiff=10^(-3))

# estimate the logistic regression parameter
### bHat <- ridgeGLMmultiT(Y, X, lambdas=optLambdas, targetMat=targets, model="logistic")
```

`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	Anumeric 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 individual.

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 an all-positive numeric, the cross-validated optimal penalty parameters.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N., Chen, Y. (2021), "Penalized estimation of the Gaussian graphical model from data with replicates", *Statistics in Medicine*, 40(19), 4279-4293.

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 individual.
`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 an all-positive numeric, the cross-validated optimal penalty parameters.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N., Chen, Y. (2021), "Penalized estimation of the Gaussian graphical model from data with replicates", *Statistics in Medicine*, 40(19), 4279-4293.

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:

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

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:

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

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=3,
                                          0.00001, 100,
                                          10, fold=5,
                                          target=0*Sigma1)

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

`ridgeGLM` *Ridge estimation of generalized linear models.*

Description

Function that evaluates the targeted ridge estimator of the regression parameter of generalized linear models.

Usage

```
ridgeGLM(Y, X, U=matrix(ncol=0, nrow=length(Y)), lambda,
         lambdaG=0, Dg=matrix(0, ncol=ncol(X), nrow=ncol(X)),
         target=rep(0, ncol(X)), model="linear",
         minSuccDiff=10^(-10), maxIter=100)
```

Arguments

`Y`	A `numeric` being the response vector.
`X`	The design `matrix` of the penalized covariates. The number of rows should match the number of elements of `Y`.
`U`	The design `matrix` of the unpenalized covariates. The number of rows should match the number of elements of `Y`.
`lambda`	A positive `numeric` that is the ridge penalty parameter.
`lambdaG`	A positive `numeric` that is the generalized ridge penalty parameter.
`Dg`	A non-negative definite `matrix` of the unscaled generalized ridge penalty.
`target`	A `numeric` towards which the estimate is shrunken.
`model`	A `character`, either `"linear"` and `"logistic"` (a reference to the models currently implemented), indicating which generalized linear model model instance is to be fitted.
`minSuccDiff`	A `numeric`, the minimum distance between the loglikelihoods of two successive iterations to be achieved. Used only if `model="logistic"`.
`maxIter`	A `numeric` specifying the maximum number of iterations. Used only if `model="logistic"`.

Details

This function finds the maximizer of the following penalized loglikelihood: $\mathcal{L}(\mathbf{Y}, \mathbf{X}, \mathbf{U}; \boldsymbol{\beta}, \boldsymbol{\gamma}) - \frac{1}{2}\lambda\|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_2^2 - \frac{1}{2}\lambda_g(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^\top \Delta_g(\boldsymbol{\beta} - \boldsymbol{\beta}_0)$, with loglikelihood $\mathcal{L}(\mathbf{Y}, \mathbf{X}; \boldsymbol{\beta})$, response $\mathbf{Y}$, design matrices $\mathbf{X}$ and $\mathbf{U}$, regression parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, penalty parameter λ, shrinkage target $\boldsymbol{\beta}_0$, and generalized ridge penalty matrix Δ_g. For more details, see van Wieringen, Binder (2020) and Lettink et al. (2022).

Value

A `numeric`, the generalized ridge estimate of the regression parameter. If a nonempty $\mathbf{U}$ is supplied, the first few elements are the unpenalized effect estimates of the covariates that comprise this design matrix.

Note

The penalized IRLS (Iterative Reweighted Least Squares) algorithm for the evaluation of the generalized ridge logistic regression estimator may fail to converge for small penalty parameter values in combination with a nonzero shrinkage target.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N. Binder, H. (2022), "Sequential learning of regression models by penalized estimation", *submitted*.

Lettink, A., Chinapaw, M.J.M., van Wieringen, W.N. (2022), "Two-dimensional fused targeted ridge regression for health indicator prediction from accelerometer data", *submitted*.

Examples

```
# set the sample size
n <- 50

# set the true parameter
betas <- (c(0:100) - 50) / 20

# generate covariate data
X <- matrix(rnorm(length(betas)*n), nrow=n)

# sample the response
probs <- exp(tcrossprod(betas, X)[1,]) / (1 + exp(tcrossprod(betas, X)[1,]))
Y     <- numeric()
for (i in 1:n){
    Y <- c(Y, sample(c(0,1), 1, prob=c(1-probs[i], probs[i])))
}

# set the penalty parameter
lambda <- 3

# estimate the logistic regression parameter
bHat <- ridgeGLM(Y, X, lambda=lambda, target=betas/2, model="logistic")
```

---

ridgeGLMdof *Degrees of freedom of the generalized ridge estimator.*

Description

Function that evaluates the degrees of freedom of the generalized ridge estimator of the regression parameter of generalized linear models.

Usage

```
ridgeGLMdof(X, U=matrix(ncol=0, nrow=nrow(X)), lambda,
         lambdaG, Dg=matrix(0, ncol=ncol(X), nrow=ncol(X)),
         model="linear", linPred=rep(0,nrow(X)))
```

Arguments

X	The design matrix of the penalized covariates. The number of rows should match the number of elements of Y.
U	The design matrix of the unpenalized covariates. The number of rows should match the number of elements of Y.
lambda	A positive numeric that is the ridge penalty parameter.
lambdaG	A positive numeric that is the generalized ridge penalty parameter.
Dg	A non-negative definite matrix of the unscaled generalized ridge penalty.
model	A character, either "linear" and "logistic" (a reference to the models currently implemented), indicating for which generalized linear model model instance the degrees of freedom is to be evaluated.
linPred	A numeric, the linear predictor associated with the provided X, U, lambda, lambdaG, and Dg. The number of elements of linPred should match the number of rows of X and U.

Details

The degrees of freedom of the regular ridge regression estimator is usually defined the trace of the ridge hat matrix: $\text{tr}[\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_{pp})^{-1}\mathbf{X}^\top]$. That of the regular ridge logistic regression estimator is defined analoguously by Park, Hastie (2008). Lettink et al. (2022) translates these definitions to the generalized ridge (logistic) regression case.

Value

A numeric, the degrees of freedom consumed by the (generalized) ridge (logistic) regression estimator.

Author(s)

W.N. van Wieringen.

References

Park, M. Y., & Hastie, T. (2008). Penalized logistic regression for detecting gene interactions. *Biostatistics*, 9(1), 30-50.

Lettink, A., Chinapaw, M.J.M., van Wieringen, W.N. (2022), "Two-dimensional fused targeted ridge regression for health indicator prediction from accelerometer data", *submitted*.

Examples

```
# set the sample size
n <- 50

# set the true parameter
betas <- (c(0:100) - 50) / 20

# generate covariate data
```

```
X <- matrix(rnorm(length(betas)*n), nrow=n)

# set the penalty parameter
lambda <- 3

# estimate the logistic regression parameter
dofs <- ridgeGLMdof(X, lambda=lambda, lambdaG=0,
                    model="logistic",
                    linPred=tcrossprod(X, t(betas)))
```

`ridgeGLMmultiT` *Multi-targeted ridge estimation of generalized linear models.*

Description

Function that evaluates the multi-targeted ridge estimator of the regression parameter of generalized linear models.

Usage

```
ridgeGLMmultiT(Y, X, U=matrix(ncol=0, nrow=length(Y)),
               lambdas, targetMat, model="linear",
               minSuccDiff=10^(-10), maxIter=100)
```

Arguments

`Y`	A `numeric` being the response vector.
`X`	The design `matrix` of the penalized covariates. The number of rows should match the number of elements of `Y`.
`U`	The design `matrix` of the unpenalized covariates. The number of rows should match the number of elements of `Y`.
`lambdas`	An all-positive `numeric`, vector of penalty parameters, one per target.
`targetMat`	A `matrix` with targets for the regression parameter as columns.
`model`	A `character`, either `"linear"` and `"logistic"` (a reference to the models currently implemented), indicating which generalized linear model model instance is to be fitted.
`minSuccDiff`	A `numeric`, the minimum distance between the loglikelihoods of two successive iterations to be achieved. Used only if `model="logistic"`.
`maxIter`	A `numeric` specifying the maximum number of iterations. Used only if `model="logistic"`.

Details

This function finds the maximizer of the following penalized loglikelihood: $\mathcal{L}(\mathbf{Y}, \mathbf{X}; \boldsymbol{\beta}) - \frac{1}{2}\sum_{k=1}^{K} \lambda_k \|\boldsymbol{\beta} - \boldsymbol{\beta}_{k,0}\|_2^2$, with loglikelihood $\mathcal{L}(\mathbf{Y}, \mathbf{X}; \boldsymbol{\beta})$, response $\mathbf{Y}$, design matrix $\mathbf{X}$, regression parameter $\boldsymbol{\beta}$, penalty parameter λ, and the k-th shrinkage target $\boldsymbol{\beta}_{k,0}$. For more details, see van Wieringen, Binder (2020).

Value

The ridge estimate of the regression parameter.

Author(s)

W.N. van Wieringen.

References

van Wieringen, W.N. Binder, H. (2020), "Online learning of regression models from a sequence of datasets by penalized estimation", *submitted*.

Examples

```
# set the sample size
n <- 50

# set the true parameter
betas <- (c(0:100) - 50) / 20

# generate covariate data
X <- matrix(rnorm(length(betas)*n), nrow=n)

# sample the response
probs <- exp(tcrossprod(betas, X)[1,]) / (1 + exp(tcrossprod(betas, X)[1,]))
Y     <- numeric()
for (i in 1:n){
    Y <- c(Y, sample(c(0,1), 1, prob=c(1-probs[i], probs[i])))
}

# set the penalty parameter
lambdas <- c(1,3)

# estimate the logistic regression parameter
# bHat <- ridgeGLMmultiT(Y, X, lambdas, model="logistic",
#                        targetMat=cbind(betas/2, rnorm(length(betas))))
```

`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 $\mathcal{N}(\mathbf{0}_p, \boldsymbol{\Omega}^{-1})$. The estimator maximizes the following penalized loglikelihood:

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

where $\mathbf{S}$ the sample covariance matrix, $\boldsymbol{\Lambda}$ 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.

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	Anumeric 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 $\mathcal{N}(\mathbf{0}_p, \boldsymbol{\Omega}^{-1})$. The estimator maximizes the following penalized loglikelihood:

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

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`.

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 individual.
`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: $\mathbf{Y}_{i,k_i} = \mathbf{Z}_i + \boldsymbol{\varepsilon}_{i,k_i}$, where $\mathbf{Z}_i$ and $\boldsymbol{\varepsilon}_{i,k_i}$ represent the signal and noise, respectively. Each observation $\mathbf{Y}_{i,k_i}$ follows a multivariate normal law of the form $\mathbf{Y}_{i,k_i} \sim \mathcal{N}(\mathbf{0}_p, \boldsymbol{\Omega}_z^{-1} + \boldsymbol{\Omega}_\varepsilon^{-1})$, which results from the distributional assumptions of the signal and the noise, $\mathbf{Z}_i \sim \mathcal{N}(\mathbf{0}_p, \boldsymbol{\Omega}_z^{-1})$ and $\boldsymbol{\varepsilon}_{i,k_i} \sim \mathcal{N}(\mathbf{0}_p, \boldsymbol{\Omega}_\varepsilon^{-1})$, and their independence. The model parameters are estimated by means of a penalized EM algorithm that maximizes the loglikelihood augmented with the penalty $\lambda_z \|\boldsymbol{\Omega}_z - \mathbf{T}_z\|_F^2 + \lambda_\varepsilon \|\boldsymbol{\Omega}_\varepsilon - \mathbf{T}_\varepsilon\|_F^2$, 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. (2021), "Penalized estimation of the Gaussian graphical model from data with replicates", *Statistics in Medicine*, 40(19), 4279-4293.

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 individual.
`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: $\mathbf{Y}_{i,k_i} = \mathbf{Z}_i + \boldsymbol{\varepsilon}_{i,k_i}$, where $\mathbf{Z}_i$ and $\boldsymbol{\varepsilon}_{i,k_i}$ represent the signal and noise, respectively. Each observation $\mathbf{Y}_{i,k_i}$ follows a multivariate normal law of the form $\mathbf{Y}_{i,k_i} \sim \mathcal{N}(\mathbf{0}_p, \boldsymbol{\Omega}_z^{-1} + \boldsymbol{\Omega}_\varepsilon^{-1})$, which results from the distributional assumptions of the signal and the noise, $\mathbf{Z}_i \sim \mathcal{N}(\mathbf{0}_p, \boldsymbol{\Omega}_z^{-1})$ and $\boldsymbol{\varepsilon}_{i,k_i} \sim \mathcal{N}(\mathbf{0}_p, \boldsymbol{\Omega}_\varepsilon^{-1})$ with $\boldsymbol{\Omega}_\varepsilon$ 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. (2021), "Penalized estimation of the Gaussian graphical model from data with replicates", *Statistics in Medicine*, 40(19), 4279-4293.

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

∗ **package**
porridge-package, 3

genRidgePenaltyMat, 4

makeFoldsGLMcv, 6

optPenaltyGGMmixture.kCVauto, *3*, 7
optPenaltyGLM.kCVauto, *3*, 9
optPenaltyGLMmultiT.kCVauto, *3*, 11
optPenaltyPgen.kCVauto.banded, *3*, 13
optPenaltyPgen.kCVauto.groups, *3*, 15
optPenaltyPmultiT.kCVauto, *3*, 17
optPenaltyPrep.kCVauto, *3*, 19
optPenaltyPrepEdiag.kCVauto, *3*, 21

porridge *(*porridge-package*)*, 3
porridge-package, 3

rags2ridges, *3*
ridgeGGMmixture, *3*, 23
ridgeGLM, *3*, 25
ridgeGLMdof, 27
ridgeGLMmultiT, *3*, 29
ridgeP, *31*, *38*
ridgePgen, *3*, *14*, 30, *33*, *34*
ridgePgen.kCV, *3*, 32
ridgePgen.kCV.banded, *3*, 33
ridgePgen.kCV.groups, *3*, 35
ridgePmultiT, *3*, 37
ridgePrep, *3*, 39
ridgePrepEdiag, *3*, 41