# Package 'GRridge'

October 16, 2019

**Type** Package

**Title** Better prediction by use of co-data: Adaptive group-regularized ridge regression

**Version** 1.8.0

**Date** 2019-4-16

**Author** Mark A. van de Wiel <mark.vdwiel@vumc.nl>, Putri W. Novianti <p.novianti@vumc.nl>

**Maintainer** Mark A. van de Wiel <mark.vdwiel@vumc.nl>

**Depends** R (>= 3.2), penalized, Iso, survival, methods, graph,stats,glmnet,mvtnorm

**Suggests** testthat

**Description** This package allows the use of multiple sources of co-data (e.g. external p-values, gene lists, annotation) to improve prediction of binary, continuous and survival response using (logistic, linear or Cox) group-regularized ridge regression. It also facilitates post-hoc variable selection and prediction diagnostics by cross-validation using ROC curves and AUC.

**biocViews** Classification, Regression, Survival, Bayesian, RNASeq, GenePrediction, GeneExpression, Pathways, GeneSetEnrichment, GO, KEGG, GraphAndNetwork, ImmunoOncology

**License** GPL-3

**LazyLoad** yes

**git_url** https://git.bioconductor.org/packages/GRridge

**git_branch** RELEASE_3_9

**git_last_commit** e780947

**git_last_commit_date** 2019-05-02

**Date/Publication** 2019-10-15

## R topics documented:

---

`GRridge-package` *Implements adaptive group-regularized (logistic) ridge regression by use of co-data.*

---

## Description

This package implements adaptive group-regularized (logistic) ridge regression by use of co-data. It uses co-data to improve predictions of binary and continuous response from high-dimension (e.g. genomics) data. Here, co-data is auxiliary information on variables (e.g. genes), such as annotation or p-values from other studies. The package includes convenience functions to convert such co-data to the correct input format. In additiion, in includes functions for evaluating the predictive performance.

## Details

| | |
|---|---|
| Package: | GRridge |
| Type: | Package |
| Version: | 1.7.3 |
| Date: | 2018-11-29 |
| License: | GPL |

Main functions in the GRridge package are:

`auc`: Computes Area-under-the-ROC-curve
`CreatePartition`: Creates a partition (groups) of variables
`dataFarkas`: Large data set plus external information
`dataSimlin`: Small simulated data set with linear response
`dataVerlaat`: Methylation data plus external information
`dataWurdinger`: RNAseq data plus external information
`simlinsmall`: Simulated data for linear regression
`grridge`: Group-regularized (logistic, survival) ridge regression
`grridgeCV`: Cross-validated predictions for a `grridge` (logistic, survival) regression.
`matchGeneSets`: Creates a grouping of variables (genes) from gene sets
`mergeGroups`: Merge groups in a partition
`PartitionsSelection`: Co-data selection in a Group-regularized ridge regression model
`predict.grridge`: Predictions for new samples from a `grridge` object
`roc`: Computes an ROC-curve for probabilistic classifiers

## Author(s)

Mark A. van de Wiel (mark.vdwiel@vumc.nl), Putri Novianti (p.novianti@vumc.nl)

## References

Mark van de Wiel, Tonje Lien, Wina Verlaat, Wessel van Wieringen, Saskia Wilting. (2016). Better prediction by use of co-data: adaptive group-regularized ridge regression. Statistics in Medicine, 35(3), 368-81.

Novianti PW, Snoek B, Wilting SM, van de Wiel MA (2017). Better diagnostic signatures from RNAseq data through use of auxiliary co-data. Bioinformatics, 33, 1572-1574.

## See Also

`GRridge` depends on: `penalized`. Examples: `grridge`

---

# auc *Area under the ROC curve*

---

## Description

Computes Area-under-the-ROC-curve

## Usage

```
auc(rocout)
```

## Arguments

| | |
|---|---|
| `rocout` | Matrix with two rows |

## Details

Operates on the output of `roc`. The rows of the input matrix represent the False Positive Rates (FPR) and corresponding True Positive Rates (TPR) at fixed thresholds.

## Value

Numeric. Interpolated Area-under-the-ROC-curve.

## Author(s)

Mark A. van de Wiel

## See Also

ROC-curves: `roc`. creating multiple partitions: `CreatePartition`. Examples: `grridge`.

## Examples

```
# Load data objects
data(dataFarkas)

## In this example, we provide one partition only
## see "CreatePartition" for examples in creating multiple partitions
firstPartition <- CreatePartition(CpGannFarkas)

# grFarkas <- grridge(datcenFarkas,respFarkas, firstPartition,monotone=FALSE)

## Prediction of the grridge model to the training samples
#cutoffs <- rev(seq(0,1,by=0.1))
#fakenew <- datcenFarkas
#yhat <- predict.grridge(grFarkas,fakenew)

#rocridgeF <- roc(probs=as.numeric(yhat[,2]),true=respFarkas[1:30],cutoffs=cutoffs)
#auc(rocridgeF)
```

---

`CreatePartition` *Creates a partition (groups) of variables*

---

## Description

Creates a partition (groups) of variables from nominal (factor) or numeric input

## Usage

```
CreatePartition(vec,varnamesdata=NULL,subset=NULL,grsize=NULL,
                decreasing=TRUE,uniform=FALSE,ngroup=10,mingr=25)
```

## Arguments

| | |
|---|---|
| `vec` | Factor, numeric vector or character vector. |
| `subset` | Character vector. Names of variables (features) that correspond to the values in `vec`. Allows to make a partition on a subset of all variables. Requires `varnamesdata`. |
| `varnamesdata` | Character vector. Names of the variables (features). Only relevant when `vec` is a character vector OR when `subset` is specified. |
| `grsize` | Numeric. Size of the groups. Only relevant when `vec` is a numeric vector and `uniform=TRUE`. |
| `decreasing` | Boolean. If `TRUE` then `vec` is sorted in decreasing order. |
| `uniform` | Boolean. If `TRUE` the group sizes are as equal as possible. |
| `ngroup` | Numeric. Number of the groups to create. Only relevant when `vec` is a numeric vector and `grsize` is NOT specified. |
| `mingr` | Numeric. Minimum group size. Only relevant when `vec` is a numeric vector and `uniform=FALSE`. |

## Details

A convenience function to create partitions of variables from external information that is stored in vec. If vec is a factor then the levels of the factor define the groups. If vec is a character vector, then varnamesdata need to be specified (vec is supposed to be a subset of varnamesdata, e.g. a published gene list). In this case a partition of two groups is created: one with those variables of varnamesdata that also appear in vec and one which do not appear in vec. If vec is a numeric vector, then groups contain the variables corresponding to grsize consecutive values of the values in vec. Alternatively, the group size is determined automatically from ngroup. If uniform=FALSE, a group with rank $r$ is of approximate size mingr*(r^f), where f>1 is determined such that the total number of groups equals ngroup. Such unequal group sizes enable the use of fewer groups (and hence faster computations) while still maintaining a good 'resolution' for the extreme values in vec. About decreasing: if smaller values of components of vec mean 'less relevant' (e.g. test statistics, absolute regression coefficients) use decreasing=TRUE, else use decreasing=FALSE, e.g. for p-values. If subset is defined, then varnamesdata should be specified as well. The parition will then only be applied to variables in subset and in varnamesdata.

## Value

A list the components of which contain the indices of the variables belonging to each of the groups.

## Author(s)

Mark A. van de Wiel

## See Also

For gene sets (overlapping groups): matchGeneSets. Further example in real life dataset: grridge.

## Examples

```
#SOME EXAMPLES ON SMALL NR OF VARIABLES

#EXAMPLE 1: partition based on known gene signature
genset <- sapply(1:100,function(x) paste("Gene",x))
signature <- sapply(seq(1,100,by=2),function(x) paste("Gene",x))
SignaturePartition <- CreatePartition(signature,varnamesdata=genset)

#EXAMPLE 2: partition based on factor variable
Genetype <- factor(sapply(rep(1:4,25),function(x) paste("Type",x)))
TypePartition <- CreatePartition(Genetype)

#EXAMPLE 3: partition based on continuous variable, e.g. p-value
pvals <- rbeta(100,1,4)

#Creating a partition of 10 equally-sized groups, corresponding to increasing p-values.
PvPartition <- CreatePartition(pvals, decreasing=FALSE,uniform=TRUE,ngroup=10)

#Alternatively, create a partition of 5 unequally-sized groups,
#with minimal size at least 10. Group size
#increases with less relevant p-values.
# Recommended when nr of variables is large.
PvPartition2 <- CreatePartition(pvals, decreasing=FALSE,uniform=FALSE,ngroup=5,mingr=10)

#EXAMPLE 4: partition based on subset of variables,
#e.g. p-values only available for 50 genes.
```

```
genset <- sapply(1:100,function(x) paste("Gene",x))

subsetgenes <- sort(sapply(sample(1:100,50),function(x) paste("Gene",x)))

pvals50 <- rbeta(50,1,6)

#Returns the partition for the subset based on the indices of
#the variables in entire genset. Variables not
#present in subset will receive group-penalty = 1 for this partition.

PvPartitionSubset <- CreatePartition(pvals50, varnamesdata = genset,subset = subsetgenes,
                                     decreasing=FALSE,uniform=TRUE, ngroup=5)

#EXAMPLE 5: COMBINING PARTITIONS

#Combines partitions into one list with named components.
#This can be use as input for the grridge() #function.
#NOTE: if one aims to use one partition only, then this can be directly used in grridge().

MyPart <- list(signature=SignaturePartition, type = TypePartition,
               pval = PvPartition, pvalsubset=PvPartitionSubset)
```

---

`dataFarkas` *Contains three R-objects, including the data and the binary response*

---

## Description

The three objects are: datcenFarkas: methylation data for cervix samples (arcsine-transformed beta values); respFarkas: binary response;and CpGannFarkas: annotation of probes according to location

## Format

- datcenFarkas: data frame [1:40000,1:37]
- respfarkas: Factor w/ 2 levels "Normal","Precursor"
- CpGannFarkas: Factor w/ 6 levels "CpG-Island", "North-Shelf", "South-Shelf", "North-Shore", "South-Shore", "Distant"

## Details

This data is used for illustration in the statistical paper below.

## Value

Three R objects, i.e. a matrix contains methylation data, a vector contains binary responses and an annotation matrix.

## Source

Farkas, S. et al. (2013). Genome-wide DNA methylation assay reveals novel candidate biomarker genes in cervical cancer. Epigenetics, 8, 1213.

## References

Mark van de Wiel, Tonje Lien, Wina Verlaat, Wessel van Wieringen, Saskia Wilting. (2016). Better prediction by use of co-data: adaptive group-regularized ridge regression. Statistics in Medicine, 35(3), 368-81.

## Examples

```
data(dataFarkas)
```

---

# dataSimlin *Small simulated data set with linear response*

---

## Description

The three objects are:

Y: vector with response values (linear)

simdata: matrix with simulated variables

part5: list with 5 components representing partition of variables

## Usage

```
data(dataSimlin)
```

## Details

This is simulated data allowing a quick test for grridge

## Value

Three R objects (see Description)

## Examples

```
data(dataSimlin)

#apply grridge
grsim <- grridge(highdimdata=simdata, response=Y, partitions=part5, unpenal = ~1, innfold=10)

#apply CV
grsimcv <- grridgeCV(grsim,simdata,Y,outerfold=5)
```

`dataVerlaat` *Contains 5 R-objects, including the data and the binary response*

## Description

The five objects are: datcenVerlaat: methylation data for cervix samples (arcsine-transformed beta values); respVerlaat: binary response; diffmeanFarkas: effect size external study, Cases minus Controls; pvalFarkas: p-values from external study; and CpGann: annotation of probes according to location

## Usage

```
data(dataVerlaat)
```

## Format

The formats are:

datcenVerlaat: data frame [1:9691,1:44];

respVerlaat: numeric [1:44], 0 = Normal, 1 = Precursor;

diffmeanFarkas: numeric [1:44] ;

pvalFarkas: numeric [1:44];

CpGann: Factor w/ 6 levels "CpG-Island", "North-Shelf", "South-Shelf", "North-Shore", "South-Shore", "Distant"

## Details

This data is used for illustration in the statistical paper below.

## Value

Five R objects (see description)

## References

Mark van de Wiel, Tonje Lien, Wina Verlaat, Wessel van Wieringen, Saskia Wilting. (2016). Better prediction by use of co-data: adaptive group-regularized ridge regression. Statistics in Medicine, 35(3), 368-81.

## Examples

```
data(dataVerlaat)
```

dataWurdinger *R-objects related to the mRNAseq data*

## Description

The four objects are:

datWurdinger_BC: A matrix containing preprocessed mRNA sequencing data (quasi-gausian scale, normalized). Columns are samples (81 samples with Breast Cancer and Colorectal Cancer) and rows are features (18410 features).

respWurdinger: A factor containing responses for samples with Breast cancer (n=40) and colorectal cancer (n=41)

annotationWurdinger: A list containing ensembleID, geneSymbol, entrezID and chromosome location.;

coDataWurdinger: A list containing co-data sets from external sources, namely (i) a list of genes that are expressed in platelets; (ii) immunologic signature pathway and; (iii) transcription factor based pathway and a list of cancer somatic genes.

## Usage

```
data(dataWurdinger)
```

## Details

This data is used for illustration in the application paper below.

## Value

Four R objects (see Description)

## Source

Wurdinger,T., et al (2015). RNA-seq of tumor-educated platelets enables blood-based pan-cancer, multiclass, and molecular pathway cancer diagnostics. Cancer Cell, 28(5), 666-676.

## Examples

```
data(dataWurdinger)

# Transform the data set to the square root scale
dataSqrtWurdinger <- sqrt(datWurdinger_BC)
#
#Standardize the transformed data
datStdWurdinger <- t(apply(dataSqrtWurdinger,1,
                           function(x){(x-mean(x))/sd(x)}))
#
# A list of gene names in the primary RNAseq data
genesWurdinger <- as.character(annotationWurdinger$geneSymbol)
```

grridge *Group-regularized (logistic) ridge regression*

## Description

This function implements adaptive group-regularized (logistic) ridge regression by use of co-data. It uses co-data to improve predictions of binary and continuous response from high-dimension (e.g. genomics) data. Here, co-data is auxiliary information on variables (e.g. genes), such as annotation or p-values from other studies.

## Usage

```
grridge(highdimdata, response, partitions, unpenal = ~1,
        offset=NULL, method="exactstable",
        niter=10, monotone=NULL, optl=NULL, innfold=NULL,
        fixedfoldsinn=TRUE, maxsel=c(25,100),selectionEN=FALSE,cvlmarg=1,
        savepredobj="all", dataunpen=NULL, ord = 1:length(partitions),
        comparelasso=FALSE,optllasso=NULL,cvllasso=TRUE,
        compareunpenal=FALSE,trace=FALSE,modus=1,
        EBlambda=FALSE,standardizeX = TRUE)
```

## Arguments

| | |
|---|---|
| highdimdata | Matrix or numerical data frame. Contains the primary data of the study. Columns are samples, rows are variables (features). |
| response | Factor, numeric, binary or survival. Response values. The number of response values should equal ncol(highdimdata). |
| partitions | List of lists. Each list component contains a partition of the variables, which is again a list. See details. |
| unpenal | Formula. Includes unpenalized variables. Set to unpenal = ~0 if an intercept is not desired. |
| offset | Numeric (vector). Optional offset, either one constant or sample-specific, in which case length(offset)=ncol(highdimdata) |
| method | Character. Equal to "exactstable": the stable iterative, systems-based method, "stable": the iterative non-systems-based method, "exact": the non-iterative, systems-based method, "adaptridge": adaptive ridge (not recommended). |
| niter | Integer. Maximum number of re-penalization iterations. |
| monotone | Vector of booleans. If the jth component of monotone equals TRUE, then the group-penalties are forced to be monotone. If monotone=NULL monotony is not imposed for any partition. |
| optl | Numeric. Value of the global regularization parameter (lambda). If specified, it skips optimization by cross-validation. |
| innfold | Integer. The fold for cross-validating the global regularization parameter lambda and for computing cross-validated likelihoods. Defaults too LOOCV. |
| fixedfoldsinn | Boolean. Use fixed folds for inner cross-validation? |
| selectionEN | Boolean. If selectionEN=TRUE then post-hoc variable selection by weighted elastic net is performed. |

| | |
|---|---|
| maxsel | Vector of integers. The maximum number of selected variables. Can be multiple to allow comparing models of various sizes. |
| cvlmarg | Numeric. Maximum margin (in percentage) that the cross-validated likelihood of the model with selected variables may deviate from the optimum one. |
| savepredobj | Character. If savepredobj="last", only the last penalized prediction object is saved; if savepredobj="all" all are saved; if savepredobj="none", none are saved. |
| dataunpen | Data frame. Optional data for unpenalized variables. |
| ord | Integer vector. The order in which the partitions in partitions are used. |
| comparelasso | Boolean. If comparelasso=TRUE the results of lasso regression are included. |
| optllasso | Numeric. Value of the global regularization parameter (lambda) in the lasso. If specified, optimization by cross-validation is skipped. |
| cvllasso | Boolean. If cvllasso=TRUE it returns the cross-validated likelihood for lasso when comparelasso=TRUE. |
| compareunpenal | Boolean. If compareunpenal=TRUE the results of regression with unpenalized covariates only are included. Only relevant when dataunpenal is specified. |
| trace | Boolean. If trace=TRUE the results of the cross-validation for parameter (lambda) tuning are shown. |
| modus | Integer. Please use modus=1. Only use modus=2 when backward compatibility with versions <= 1.6 is desired. |
| EBlambda | Boolean. If EBlambda=TRUE global lambda is estimated by empirical Bayes (currently only available for linear model). |
| standardizeX | Boolean. If standardizeX=TRUE variables in X are standardized prior to the analysis. |

## Details

About partitions: this is a list of partitions or one partition represented as a simple list. Each partition is a (named) list that contains the indices (row numbers) of the variables in the concerning group. Such a partition is usually created by CreatePartition. About savepredobj: use savepredobj="all" if you want to compare performances of the various predictors (e.g. ordinary ridge, group-regularized ridge, group-regularized ridge + selection) using grridgeCV. About monotone: We recommend to set the jth component of monotone to TRUE when the jth partition is based on external p-values, test statistics or regression coeeficients. This increases stability of the predictions. If selectionEN=TRUE, EN selection will, for all elements m of maxsel, select exactly m or fewer variables. Note that EN is only used for selection; the final predictive model is a group-ridge model fitted only on the selected variables using the penalties estimated by GR-ridge. Using multiple values for maxsel allows comparing models of various sizes, also in terms of cross-validated performance when using grridgeCV. About cvlmarg: We recommended to use values between 0 and 2. A larger value will generally result in fewer selected variables by forward selection. About innfold: for large data sets considerable computing time may be saved when setting innfold=10 instead of default leave-one-out-cross-validation (LOOCV). About method: "exactstable" is recommended. If the number of variables is not very large, say <2000, the faster non-iterative "exact" method can be used as an alternative. grridge uses the penalized package to fit logistic and survival ridge models; glmnet is used for linear response and for fitting lasso when comparelasso=TRUE.

## Value

A list object containing:

| | |
|---|---|
| `true` | True values of the response |
| `cvfit` | Measure of fit. Cross-validated likelihoods from the iterations for linear and survival model; minus CV error for linear model |
| `lambdamults` | List of lists object containing the penalty multipliers per group per partition |
| `optl` | Global penalty parameter lambda |
| `lambdamultvec` | Vector with penalty multipliers per variable |
| `predobj` | List of prediction objects |
| `betas` | Estimated regression coefficients |
| `reslasso` | Results of the lasso. `NULL` when `comparelasso=FALSE` |
| `resEN` | Results of the Elastic Net selection for all elements of `maxsel`. `list()` when `selectionEN=FALSE` |
| `model` | Model used for fitting: logistic, linear or survival |
| `arguments` | Arguments used to call the function |
| `allpreds` | Predictions on the same data |

## Author(s)

Mark A. van de Wiel

## References

Mark van de Wiel, Tonje Lien, Wina Verlaat, Wessel van Wieringen, Saskia Wilting. (2016). Better prediction by use of co-data: adaptive group-regularized ridge regression. Statistics in Medicine, 35(3), 368-81.

Novianti PW, Snoek B, Wilting SM, van de Wiel MA (2017). Better diagnostic signatures from RNAseq data through use of auxiliary co-data. Bioinformatics, 33, 1572-1574.

## See Also

Creating partitions: `CreatePartition`; Cross-validation for assessing predictive performance: `grridgeCV`.

## Examples

```
## NOTE:
## 1. EXAMPLE DEVIATES SOMEWHAT FROM THE EXAMPLE IN THE MANUSCRIPT IN ORDER TO SHOW SOME
##    OTHER FUNCTIONALITIES.
## 2. HERE WE SHOW A SIMPLE EXAMPLE FROM THE FARKAS DATA SET
## MORE EXTENSIVE EXAMPLES OF FUNCTIONALITIES IN THE GRRIGDE PACKAGE ARE PROVIDED IN
## VIGNETTE DOCUMENTATION FILE


## 1ST EXAMPLE: Farkas DATA, USING ANNOTATION: DISTANCE TO CpG

##load data objects:
##datcenFarkas: methylation data for cervix samples (arcsine-transformed beta values)
##respFarkas: binary response (Normal and Precursor)
##CpGannFarkas: annotation of probes according to location
```

```
##(CpG-Island, North-Shelf, South-Shelf, North-Shore, South-Shore, Distant)
data(dataFarkas)

##Create list of partition(s), here only one partition included
partitionFarkas <- list(cpg=CreatePartition(CpGannFarkas))

##Group-regularized ridge applied to data datcenFarkas,
##response respFarkas and partition partitionFarkas.
##Saves the prediction objects from ordinary and group-regularized ridge.
##Includes unpenalized intercept by default.

#grFarkas <- grridge(datcenFarkas,respFarkas, optl=5.680087,
#                      partitionFarkas,monotone=FALSE)

## 2ND EXAMPLE: Verlaat DATA, USING P-VALUES AND SIGN OF EFFECT FROM FARKAS DATA
## see vignette documentation file!
```

---

`grridgeCV` *Returns the cross-validated predictions*

---

## Description

Returns the cross-validated predictions for a `grridge` logistic, linear or Cox regression.

## Usage

```
grridgeCV(grr, highdimdata, response, outerfold = length(response),
           fixedfolds = TRUE, recalibrate=FALSE)
```

## Arguments

| | |
|---|---|
| `grr` | List. Output of GRridge function. |
| `highdimdata` | Matrix or numerical data frame. Contains the primary data of the study. Columns are samples, rows are variables (features). |
| `response` | Factor, numeric, binary or survival. Response values. The number of response values should equal `ncol(highdimdata)`. |
| `outerfold` | Integer. Fold used for cross-validation loop. |
| `fixedfolds` | Boolean. Use fixed folds for cross-validation? |
| `recalibrate` | Boolean. Should the prediction model be recalibrated on the test samples? Only implemented for logistic and linear regression with only penalized covariates. |

## Details

This convenience function returns cross-validated predictions from `grridge`, including those from ordinary (logistic) ridge regression. It can be used to compute ROC-curves. About `recalibrate`: this option allows to compare recalibrated models, but only if the test sample size is large enough. See `predict.grridge`

## Value

For linear and logistic regression: A matrix containing the predictions. The first column contains the sample indices, the second the prediction by ordinary ridge, the third the predictions by group-regularized ridge, the next (optionally) the predictions by group-regularized ridge plus Elastic Net selection, possibly for multiple model sizes (as specified by the argument `maxsel` when using `grridge`). Finally, it may contain predictions by lasso and/or a regression model with unpenalized covariates only. For Cox regression: the relative hazard is returned, which equals the exponentiated linear predictor.

## Author(s)

Mark A. van de Wiel

## References

Mark van de Wiel, Tonje Lien, Wina Verlaat, Wessel van Wieringen, Saskia Wilting. (2016). Better prediction by use of co-data: adaptive group-regularized ridge regression. Statistics in Medicine, 35(3), 368-81.

## See Also

For logistic regression: ROC-curves: `roc`. AUC: `auc`. GRridge: `link{grridge}`.

## Examples

```
# load data objects
data(dataFarkas)

# In this example, we provide one partition only
# see "CreatePartition" for examples in creating multiple partitions
firstPartition <- CreatePartition(CpGannFarkas)

# the optimum lambda2 is provided in this example
# worth to try:
# grFarkas <- grridge(datcenFarkas,respFarkas,firstPartition,monotone=FALSE)
# grFarkas$optl
# grFarkas <- grridge(datcenFarkas,respFarkas, optl=5.680087, firstPartition,monotone=FALSE)

###
# grFarkasCV <- grridgeCV(grFarkas,datcenFarkas,respFarkas,outerfold=10)
```

---

`hello` *Hello, World!*

---

## Description

Prints 'Hello, world!'.

## Usage

```
hello()
```

## Value

An empty object.

## Examples

```
hello()
```

---

`matchGeneSets` *Creates a grouping of variables (genes) from gene sets*

---

## Description

Creates a grouping of variables (genes) from gene sets by matching the IDs of the genes with the IDs of the members of the gene sets

## Usage

```
matchGeneSets(GeneIds, GeneSets, minlen = 25, remain = TRUE)
```

## Arguments

| | |
|---|---|
| `GeneIds` | Character vector. Vector of gene IDs. Can be any ID (gene symbol, entrezID, etc) as long as it matches with IDs used in `GeneSets`. |
| `GeneSets` | Named list of character vectors. Each component of the list represents a named gene set. Each vector a list of member genes. |
| `minlen` | Integer. Minimum number of members of a gene set. |
| `remain` | Boolean. If `remain=TRUE`, all genes that are not in any list will be grouped in one remainder group. |

## Details

About `minlen`: to avoid overfitting in the `grridge` function, we recommend to not use groups with less than 25 members, unless `monotone=TRUE` is used in the `grridge` function, in which case 10 members may suffice as a lower bound. About `remain`: it is often beneficial to down-weight genes that are not part of any gene set, so we recommend to use `remain=TRUE`

## Value

A list the components of which contain the indices of the variables belonging to each of the groups.

## Author(s)

Mark A. van de Wiel

## See Also

`grridge`, `CreatePartition`

## Examples

```
# Load data objects
data(dataWurdinger)

# Transform the data set to the square root scale
dataSqrtWurdinger <- sqrt(datWurdinger_BC)

#Standardize the transformed data
datStdWurdinger <- t(apply(dataSqrtWurdinger,1,function(x){(x-mean(x))/sd(x)}))

# A list of gene names in the primary RNAseq data
genesWurdinger <- as.character(annotationWurdinger$geneSymbol)

# We  show an example of GRridge classification model by using overlapping groups,
# i.e. pathway-based grouping. Transcription factor based pathway was extracted from
# the MSigDB (Section C3: motif gene sets; subsection: transcription factor targets;
# file's name: "c3.tft.v5.0.symbols.gmt"). The gene sets are based on
# TRANSFAC version 7.5 database (http://www.gene-regulation.com/).

# Some features may belong to more than one group. The argument minlen=25 implies
# the minimum number of features in a gene set. If remain=TRUE, gene sets with less
# than 25 members are grouped to the "remainder" group. "genesWurdinger" is an object
# containing gene names from the mRNA sequencing data set.
# See help(matchGeneSets) for more detail information.
# Also see Vignette for more detail examples

## The "TFsym" is available on https://github.com/markvdwiel/GRridgeCodata
# gseTF <- matchGeneSets(genesWurdinger,TFsym,minlen=25,remain=TRUE)
```

---

`mergeGroups` *Merge groups in a partition*

---

## Description

Pathway-based partition often contains a considerable number of gene sets (or groups). This function merges groups resulted by `matchGeneSets`. The first principal component in each group is calculated. Hierarchical clustering analysis is then performed on the first principal components from all groups. Important Note: re-grouping is only done in the non-reminder group.

## Usage

```
mergeGroups(highdimdata, initGroups=initGroups,maxGroups=maxGroups,
                    methodDistance="manhattan", methodClust="complete")
```

## Arguments

| | |
|---|---|
| `highdimdata` | Matrix or numerical data frame. Contains the primary data of the study. Columns are samples, rows are features. |
| `initGroups` | A list of initial given groups resulted from `matchGeneSets`. |
| `maxGroups` | Numeric. The new desired number of groups. |

`methodDistance` The distance method used for clustering. See `dist` for futher options. Default: manhattan distance.

`methodClust` The agglomeration method used for Grouping. See `hclust` for further options. Default: complete.

## Value

A list object containing:

`newGroups` A list the components of which contain the indices of the features belonging to each of the group. This object is the same as the object created by `matchGeneSets` and `CreatePartition`

`newGroupMembers` A list of members in the new merged groups.

## Author(s)

Putri W. Novianti

## See Also

Creating partitions: `CreatePartition`. Creating partitions based on overlaping groups (gene sets/pathways): `matchGeneSets`.

## Examples

```
# Source: http://software.broadinstitute.org/gsea/msigdb/collections.jsp
# A GMT file containing information about transcription factor targets should
# be downloaded first from the aforementioned source.
# Section C3: motif gene sets; subsection: transcription factor targets;
# file: "c3.tft.v5.0.symbols.gmt"
# Details of the gene sets:
# Gene sets contain genes that share a transcription factor binding site
# defined in the TRANSFAC (version 7.4, http://www.gene-regulation.com/) database.
# Each of these gene sets is annotated by a TRANSFAC record.

# Load data objects
data(dataWurdinger)

# Transform the data set to the square root scale
dataSqrtWurdinger <- sqrt(datWurdinger_BC)

#Standardize the transformed data
datStdWurdinger <- t(apply(dataSqrtWurdinger,1,function(x){(x-mean(x))/sd(x)}))

# A list of gene names in the primary RNAseq data
genesWurdinger <- as.character(annotationWurdinger$geneSymbol)

## Creating partitions based on pathways information (e.g. GSEA object)
## Some variables may belong to more than one groups (gene sets).
## The argument minlen=25 implies the minimum number of members in a gene set
## If remain=TRUE, gene sets with less than 25 members are grouped to the
## "remainder" group.
## The "TFsym" is available on https://github.com/markvdwiel/GRridgeCodata
# gseTF <- matchGeneSets(genesWurdinger,TFsym,minlen=25,remain=TRUE)
```

```
## Regrouping gene sets by hierarchical clustering analysis.
## The number of gene sets from the GSEA database is relatively too high to be used
## in the GRridge model. Here, the initial gene sets are re-grouped into maxGroups=5, using
## information from the primary data set.
# gseTF_newGroups <- mergeGroups(highdimdata=datStdWurdinger, initGroups =gseTF, maxGroups=5);

## Extracting indices of new groups
## This following object (gseTF2) can be used further as an input
## in the "partitions" argument in the "grridge" function
# gseTF2 <- gseTF_newGroups$newGroups

## Members of the new groups
# newGroupMembers <- gseTF_newGroups$newGroupMembers
```

---

`PartitionsSelection` *Co-data selection in a Group-regularized ridge regression model*

---

## Description

This function implements a procedure to optimize the use of co-data in a GRridge model. Although there is no harm to include as much as co-data in a GRridge model, ordering and selecting co-data can optimized the performance of a GRridge model. This procedure is similar with forward feature selection in a classical regression model

## Usage

```
PartitionsSelection(highdimdata, response, partitions,
                    monotoneFunctions, optl=NULL, innfold=NULL)
```

## Arguments

| | |
|---|---|
| `highdimdata` | Matrix or numerical data frame. Contains the primary data of the study. Columns are samples, rows are features. |
| `response` | Factor, numeric, binary or survival. Response values. The number of response values should equal `ncol(highdimdata)`. |
| `partitions` | List of lists. Each list component contains a partition of the variables, which is again a list. |
| `monotoneFunctions` | Vector. Monotone functions from each partition. This argument is necesarily specified. If the jth component of monotone equals TRUE, then the group-penalties are forced to be monotone. If `monotone=NULL` monotony is not imposed for any partition. |
| `optl` | Global penalty parameter lambda |
| `innfold` | Integer. The fold for cross-validating the global regularization parameter lambda and for computing cross-validated likelihoods. Defaults LOOCV. |

## Value

A list containing (i) the indeces of the selected and ordered partitions and (ii) the optimum lambda penalty from the ridge regression.

## Author(s)

Putri W. Novianti

## See Also

Creating partitions: CreatePartition. Group-regularized ridge regression: grridge.

## Examples

```
# # Load data objects
# data(dataWurdinger)
#
# # Transform the data set to the square root scale
# dataSqrtWurdinger <- sqrt(datWurdinger_BC)
#
# #Standardize the transformed data
# datStdWurdinger <- t(apply(dataSqrtWurdinger,1,function(x){(x-mean(x))/sd(x)}))
#
# # A list of gene names in the primary RNAseq data
# genesWurdinger <- as.character(annotationWurdinger$geneSymbol)
#
# # co-data 1: a partition based on immunologic signature pathway
# # The initial gene sets (groups) are merged into five new groups, using the "mergeGroups" function
# immunPathway <- coDataWurdinger$immunologicPathway
# parImmun <- immunPathway$newClust

# # co-data 2: a partition based on a list of platelets expressed genes
# plateletsExprGenes <- coDataWurdinger$plateletgenes
# # The genes are grouped into either "NormalGenes" or "Non-overlapGenes"
# is <- intersect(plateletsExprGenes,genesWurdinger)
# im <- match(is, genesWurdinger)
# plateletsGenes <- replicate(length(genesWurdinger),"Non-overlapGenes")
# plateletsGenes[im] <- "NormalGenes"
# plateletsGenes <- as.factor(plateletsGenes)
# parPlateletGenes <- CreatePartition(plateletsGenes)
#
# # co-data 3: a partition based on chromosomal location.
# # A list of chromosomal location based on {\tt biomaRt} data bases.
# ChromosomeWur0 <- as.vector(annotationWurdinger$chromosome)
# ChromosomeWur <- ChromosomeWur0
# idC <- which(ChromosomeWur0=="MT" | ChromosomeWur0=="notBiomart" |
#                ChromosomeWur0=="Un")
# ChromosomeWur[idC] <- "notMapped"
# table(ChromosomeWur)
# parChromosome <- CreatePartition(as.factor(ChromosomeWur))
#
# partitionsWurdinger <- list(immunPathway=parImmun,
#                             plateletsGenes=parPlateletGenes,
#                             chromosome=parChromosome)
#
# #A list of monotone functions from the corresponding partitions
# monotoneWurdinger <- c(FALSE,FALSE,FALSE)
#
# # Start ordering and selecting partitions
# optPartitions <- PartitionsSelection(datStdWurdinger, respWurdinger,
```

```
#                                       partitions=partitionsWurdinger,
#                                       monotoneFunctions=monotoneWurdinger)
```

---

`predict.grridge` *Predictions for new samples*

---

## Description

Returns predictions for new samples from a `grridge` object

## Usage

```
## S3 method for class 'grridge'
predict(object, datanew, printpred = FALSE, dataunpennew=NULL,
        responsetest=NULL, recalibrate=FALSE, ...)
```

## Arguments

| | |
|---|---|
| `object` | A model object resulted from the grridge function. |
| `datanew` | Vector or data frame. Contains the new data. For a data frame: columns are samples, rows are variables (features). |
| `printpred` | Boolean. Should the predictions be printed on the screen? |
| `dataunpennew` | Vector or data frame. Optional new data for unpenalized variables. NOTE: columns are covariates, rows are samples. |
| `responsetest` | Factor, numeric, binary or survival. Response values of test samples. The number of response values should equal `ncol(datanew)`. Only relevant if `recalibrate=TRUE`. |
| `recalibrate` | Boolean. Should the prediction model be recalibrated on the test samples? Only implemented for logistic and linear regression with only penalized covariates. |
| `...` | There is no further argument is used. |

## Details

This function returns predictions of the response using the `grridge` output. It should be applied to samples NOT used for fitting of the models. About `recalibrate`: we noticed that recalibration of the linear predictor using a simple regression (with intercept and slope) can improve predictive performance, e.g. in terms of brier score or mean square error (not in terms of AUC, which is rank-based). We recommend to use it for large enough test sets (say >= 25 samples), in particular when one suspects that the test set could have somewhat different properties than the training set. For survival, the proportional hazards are returned, i.e. exp(linear predictor). If survival time predictions are desired, these may be obtained by applying `predict` to fit objects, which are stored as output of `grridge`.

## Value

A matrix containing the predictions from all models available in `grridge`.

## Author(s)

Mark A. van de Wiel

## References

Mark van de Wiel, Tonje Lien, Wina Verlaat, Wessel van Wieringen, Saskia Wilting. (2016). Better prediction by use of co-data: adaptive group-regularized ridge regression. Statistics in Medicine, 35(3), 368-81.

## See Also

Cross-validated predictions: grridgeCV. Examples: grridge.

## Examples

```
#data(dataFarkas)

#firstPartition <- CreatePartition(CpGannFarkas)

#sdsF <- apply(datcenFarkas,1,sd)
#secondPartition <- CreatePartition(sdsF,decreasing=FALSE, uniform=TRUE, grsize=5000)

## Concatenate two partitions
#partitionsFarkas <- list(cpg=firstPartition, sds=secondPartition)

## A list of monotone functions from the corresponding partition
#monotoneFarkas <- c(FALSE,TRUE)

#Hold-out first two samples
#testset <- datcenFarkas[,1:2]
#resptest <- respFarkas[1:2]

#trainingset <- datcenFarkas[,-(1:2)]
#resptraining <- respFarkas[-(1:2)]

#grFarkas <- grridge(trainingset,resptraining,optl=5.680087,
#                    partitionsFarkas,monotone=monotoneFarkas)

## Prediction of the grridge model to the test samples

#Standardize variables in testset, becasuse by default standardizeX = TRUE in grridge function.
#Here, test set is small so we use the summaries of the training set. If the test set
#is large one may opt to standardize wrt to the test set itself.

#sds <- apply(trainingset,1,sd)
#sds2 <- sapply(sds,function(x) max(x,10^{-5}))
#teststd <- (testset-apply(trainingset,1,mean))/sds2


#yhat <- predict.grridge(grFarkas,teststd)
```

---

roc *Produces ROC curve for probabilistic classifiers (e.g. logistic regression)*

---

## Description

Computes an ROC-curve for probabilistic classifiers.

## Usage

```
roc(probs, true, cutoffs)
```

## Arguments

| | |
|---|---|
| probs | Numeric vector, with values between 0 and 1 |
| true | Binary vector. |
| cutoffs | Numeric vector, with DECREASING values between 1 and 0. |

## Details

The vector probs contains predicted probabilities for the response to equal 1, as produced by a probabilistic classifier like logistic regression. The cutoffs can simply represent a grid of values between 0 and 1.

## Value

A matrix with two rows which contain corresponding False Positive and True Positive Rates for all cutoffs.

## Author(s)

Mark A. van de Wiel

## See Also

For area-under-the ROC-curve: auc. Examples: grridge.

## Examples

```
# Load data objects
data(dataFarkas)

firstPartition <- CreatePartition(CpGannFarkas)

sdsF <- apply(datcenFarkas,1,sd)
secondPartition <- CreatePartition(sdsF,decreasing=FALSE, uniform=TRUE, grsize=5000)

# Concatenate two partitions
partitionsFarkas <- list(cpg=firstPartition, sds=secondPartition)

# A list of monotone functions from the corresponding partition
monotoneFarkas <- c(FALSE,TRUE)

#grFarkas <- grridge(datcenFarkas,respFarkas,optl=5.680087,partitionsFarkas,monotone=monotoneFarkas)
#grFarkascv <- grridgeCV(grFarkas,datcenFarkas,respFarkas,outerfold=10)

#cutoffs <- rev(seq(0,1,by=0.01))
#rocgrridgeF <- roc(probs=grFarkascv[,3],true=grFarkascv[,1],cutoffs=cutoffs)
#rocridgeF <- roc(probs=grFarkascv[,2],true=grFarkascv[,1],cutoffs=cutoffs)
#plot(rocridgeF[1,],rocridgeF[2,],type="l",lty=1,ann=FALSE,col="grey")
#points(rocgrridgeF[1,],rocgrridgeF[2,],type="l",lty=1,col="black")
#legend(0.75,0.1, legend=c("ridge","GRridge"),
#       lty=c(1,1), lwd=c(1,1),col=c("grey","black"))
```

# Index