Package 'glmSparseNet'

October 16, 2019

Type Package

Title Network Centrality Metrics for Elastic-Net Regularized Models

Version 1.2.0

Description glmSparseNet is an R-
package that generalizes sparse regression models when the features (e.g. genes)
have a graph structure (e.g. protein-protein interactions), by including network-based regularizers.
glmSparseNet uses the glmnet R-
package, by including centrality measures of the network as penalty
weights in the regularization. The current version implements regularization based on node degree,
i.e. the strength and/or number of its associated edges, either by promoting hubs in the solution or
orphan genes in the solution. All the glmnet distribution families are supported, namely ``gaus-
sian",
``poisson", ``binomial", ``multinomial", ``cox", and ``mgaussian".

License GPL (>=3)

Encoding UTF-8

LazyData true

NeedsCompilation no

biocViews Software, StatisticalMethod, DimensionReduction, Regression,
Classification, Survival, Network, GraphAndNetwork

Depends R (>= 3.5), Matrix, MultiAssayExperiment, glmnet

Imports SummarizedExperiment, STRINGdb, biomaRt, futile.logger,
sparsebn, sparsebnUtils, forcats, dplyr, readr, ggplot2,
survminer, reshape2, stats, stringr, rlang, parallel, methods,
loose.rock (>= 1.0.12)

Suggests testthat, knitr, rmarkdown, survival, survcomp, pROC,
VennDiagram, BiocStyle, curatedTCGAData, TCGAutils

VignetteBuilder knitr

RoxygenNote 6.1.1

BugReports https://www.github.com/sysbiomed/glmSparseNet/issues

URL https://www.github.com/sysbiomed/glmSparseNet

git_url https://git.bioconductor.org/packages/glmSparseNet

git_branch RELEASE_3_9

git_last_commit db841b1

git_last_commit_date 2019-05-10

Date/Publication 2019-10-15

Author André Veríssimo [aut, cre],
Susana Vinga [aut],
Eunice Carrasquinha [ctb],
Marta Lopes [ctb]

Maintainer André Veríssimo <andre.verissimo@tecnico.ulisboa.pt>

R topics documented:

.calcPenalty *Calculate penalty based on data*

Description

Internal method to calculate the network using data-dependant methods

Usage

```
.calcPenalty(xdata, penalty.type, network.options = networkOptions())
```

Arguments

xdata	input data
penalty.type	which method to use
network.options	options to be used

Value

vector with penalty weights

Examples

```
xdata <- matrix(rnorm(100), ncol = 20)
glmSparseNet:::.calcPenalty(xdata, 'none')
glmSparseNet:::.calcPenalty(xdata, 'correlation',
                            networkOptions(cutoff = .6))
glmSparseNet:::.calcPenalty(xdata, 'correlation')
glmSparseNet:::.calcPenalty(xdata, 'covariance',
                            networkOptions(cutoff = .6))
glmSparseNet:::.calcPenalty(xdata, 'covariance')
glmSparseNet:::.calcPenalty(xdata, 'sparsebn')
```

.degreeGeneric *Generic function to calculate degree based on data*

Description

The assumption to use this function is that the network represented by a matrix is symetric and without any connection the node and itself.

Usage

```
.degreeGeneric(fun = stats::cor, fun.prefix = "operator", xdata,
  cutoff = 0, consider.unweighted = FALSE, chunks = 1000,
  force.recalc.degree = FALSE, force.recalc.network = FALSE,
  n.cores = 1, ...)
```

Arguments

fun	function that will calculate the edge weight between 2 nodes
fun.prefix	used to store low-level information on network as it can become to large to be stored in memory
xdata	calculate correlation matrix on each column
cutoff	positive value that determines a cutoff value
consider.unweighted	consider all edges as 1 if they are greater than 0
chunks	calculate function at batches of this value (default is 1000)
force.recalc.degree	force recalculation of penalty weights (but not the network), instead of going to cache
force.recalc.network	force recalculation of network and penalty weights, instead of going to cache
n.cores	number of cores to be used
...	extra parameters for fun

Value

a vector of the degrees

.glmSparseNetPrivate *Calculate GLM model with network-based regularization*

Description

Calculate GLM model with network-based regularization

Usage

```
.glmSparseNetPrivate(fun, xdata, ydata, network, experiment.name = NULL,
  network.options = networkOptions(), ...)
```

Arguments

fun	function to be called (glmnet or cv.glmnet)
xdata	input data, can be a matrix or MultiAssayExperiment
ydata	response data compatible with glmnet
network	type of network, see below
experiment.name	when xdata is a MultiAssayExperiment object this parameter is required
network.options	options to calculate network
...	parameters that glmnet accepts

Value

an object just as glmnet network parameter accepts:

* string to calculate network based on data (correlation, covariance) * matrix representing the network * vector with already calculated penalty weights (can also be used directly with glmnet)

.networkGenericParallel *Calculate the upper triu of the matrix*

Description

Calculate the upper triu of the matrix

Usage

```
.networkGenericParallel(fun, fun.prefix, xdata, build.output = "matrix",
  n.cores = 1, force.recalc.network = FALSE, show.message = FALSE,
  ...)
```

Arguments

fun	function that will calculate the edge weight between 2 nodes
fun.prefix	used to store low-level information on network as it can become to large to be stored in memory
xdata	base data to calculate network
build.output	if output returns a 'matrix', 'vector' of the upper triu without the diagonal or NULL with any other argument
n.cores	number of cores to be used
force.recalc.network	force recalculation, instead of going to cache
show.message	shows cache operation messages
...	extra parameters for fun

Value

depends on build.output parameter

.networkWorker *Worker to calculate edge weight for each pair of ix.i node and following*

Description

Note that it assumes it does not calculate for index below and equal to ix.i

Usage

```
.networkWorker(fun, xdata, ix.i, ...)
```

Arguments

fun	function to be used, can be cor, cov or any other defined function
xdata	original data to calculate the function over
ix.i	starting index, this can be used to save ony upper triu
...	extra parameters for fun

Value

a vector with size ‘ncol(xdata) - ix.i‘

---

buildLambda *Auxiliary function to generate suitable lambda parameters*

Description

Auxiliary function to generate suitable lambda parameters

Usage

```
buildLambda(lambda.largest = NULL, xdata = NULL, ydata = NULL,
  family = NULL, orders.of.magnitude.smaller = 3,
  lambda.per.order.magnitude = 150)
```

Arguments

lambda.largest	numeric value for largest number of lambda to consider (usually with a target of 1 selected variable)
xdata	X parameter for glmnet function
ydata	Y parameter for glmnet function
family	family parameter to glmnet function
orders.of.magnitude.smaller	minimum value for lambda (lambda.largest / 10^orders.of.magnitude.smaller)
lambda.per.order.magnitude	how many lambdas to create for each order of magnitude

Value

a numeric vector with suitable lambdas

Examples

```
buildLambda(5.4)
```

`buildStringNetwork` *Build gene network from peptide ids*

Description

This can reduce the dimension of the original network, as there may not be a mapping between peptide and gene id

Usage

```
buildStringNetwork(string.tbl, use.names = "protein")
```

Arguments

`string.tbl`	matrix with colnames and rownames as ensembl peptide id (same order)
`use.names`	default is to use protein names ('protein'), other options are 'ensembl' for ensembl gene id or 'external' for external gene names

Value

a new matrix with gene ids instead of peptide ids. The size of matrix can be different as there may not be a mapping or a peptide mapping can have multiple genes.

See Also

stringDBhomoSapiens

Examples

```
## Not run:
  all.interactions.700 <- stringDBhomoSapiens(score_threshold = 700)
  string.network       <- buildStringNetwork(all.interactions.700,
                                             use.names = 'external')
  # number of edges
  sum(network != 0)

## End(Not run)
```

`cv.glmDegree` *GLMNET cross-validation model penalizing nodes with small degree*

Description

This function overrides the 'trans.fun' options in 'network.options' with the inverse of a degree described in Veríssimo et al. (2015) that penalizes nodes with small degree.

Usage

```
cv.glmDegree(xdata, ydata, network, network.options = networkOptions(),
  ...)
```

Arguments

xdata	input data, can be a matrix or MultiAssayExperiment
ydata	response data compatible with glmnet
network	type of network, see below
network.options	options to calculate network
...	parameters that glmnet accepts

Value

see cv.glmSparseNet

See Also

glmNetSparse

Examples

```
xdata <- matrix(rnorm(100), ncol = 5)
cv.glmDegree(xdata, rnorm(nrow(xdata)), 'correlation',
            family = 'gaussian',
            nfolds = 5,
            network.options = networkOptions(min.degree = .2))
```

---

cv.glmHub *GLMNET cross-validation model penalizing nodes with small degree*

Description

This function overrides the ‘trans.fun‘ options in ‘network.options‘ with an heuristic described in Veríssimo et al. that penalizes nodes with small degree.

Usage

```
cv.glmHub(xdata, ydata, network, network.options = networkOptions(), ...)
```

Arguments

xdata	input data, can be a matrix or MultiAssayExperiment
ydata	response data compatible with glmnet
network	type of network, see below
network.options	options to calculate network
...	parameters that glmnet accepts

Value

see cv.glmSparseNet

See Also

glmNetSparse

Examples

```
xdata <- matrix(rnorm(100), ncol = 5)
cv.glmHub(xdata, rnorm(nrow(xdata)), 'correlation',
          family = 'gaussian',
          nfolds = 5,
          network.options = networkOptions(min.degree = .2))
```

`cv.glmOrphan` *GLMNET cross-validation model penalizing nodes with high degree*

Description

This function overrides the 'trans.fun' options in 'network.options' with an heuristic described in Veríssimo et al. that penalizes nodes with high degree.

Usage

```
cv.glmOrphan(xdata, ydata, network, network.options = networkOptions(),
  ...)
```

Arguments

`xdata`	input data, can be a matrix or MultiAssayExperiment
`ydata`	response data compatible with glmnet
`network`	type of network, see below
`network.options`	options to calculate network
`...`	parameters that glmnet accepts

Value

see cv.glmSparseNet

See Also

glmNetSparse

Examples

```
xdata <- matrix(rnorm(100), ncol = 5)
cv.glmOrphan(xdata, rnorm(nrow(xdata)), 'correlation',
             family = 'gaussian',
             nfolds = 5,
             network.options = networkOptions(min.degree = .2))
```

`cv.glmSparseNet` *Calculate cross validating GLM model with network-based regularization*

Description

network parameter accepts:

Usage

```
cv.glmSparseNet(xdata, ydata, network,
  network.options = networkOptions(), experiment.name = NULL, ...)
```

Arguments

`xdata`	input data, can be a matrix or MultiAssayExperiment
`ydata`	response data compatible with glmnet
`network`	type of network, see below
`network.options`	options to calculate network
`experiment.name`	Name of experiment in MultiAssayExperiment
`...`	parameters that cv.glmnet accepts

Details

* string to calculate network based on data (correlation, covariance) * matrix representing the network * vector with already calculated penalty weights (can also be used directly glmnet)

Value

an object just as cv.glmnet

Examples

```
## Not run:
    # Gaussian model
    xdata <- matrix(rnorm(500), ncol = 5)
    cv.glmSparseNet(xdata, rnorm(nrow(xdata)), 'correlation',
                    family = 'gaussian')
    cv.glmSparseNet(xdata, rnorm(nrow(xdata)), 'covariance',
                    family = 'gaussian')

## End(Not run)

#
#
# Using MultiAssayExperiment with survival model


#
```

```
# load data
xdata <- MultiAssayExperiment::miniACC

#
# build valid data with days of last follow up or to event
event.ix <- which(!is.na(xdata$days_to_death))
cens.ix <- which(!is.na(xdata$days_to_last_followup))
xdata$surv_event_time <- array(NA, nrow(colData(xdata)))
xdata$surv_event_time[event.ix] <- xdata$days_to_death[event.ix]
xdata$surv_event_time[cens.ix] <- xdata$days_to_last_followup[cens.ix]

#
# Keep only valid individuals
valid.ix <- as.vector(!is.na(xdata$surv_event_time) &
                      !is.na(xdata$vital_status) &
                      xdata$surv_event_time > 0)
xdata.valid <- xdata[, rownames(colData(xdata))[valid.ix]]
ydata.valid <- colData(xdata.valid)[,c('surv_event_time', 'vital_status')]
colnames(ydata.valid) <- c('time', 'status')

#
cv.glmSparseNet(xdata.valid,
                ydata.valid,
                nfolds          = 5,
                family          = 'cox',
                network         = 'correlation',
                experiment.name = 'RNASeq2GeneNorm')
```

`degreeCor` *Calculate the degree of the correlation network based on xdata*

Description

Calculate the degree of the correlation network based on xdata

Usage

```
degreeCor(xdata, cutoff = 0, consider.unweighted = FALSE,
  force.recalc.degree = FALSE, force.recalc.network = FALSE,
  n.cores = 1, ...)
```

Arguments

`xdata`	calculate correlation matrix on each column
`cutoff`	positive value that determines a cutoff value
`consider.unweighted`	consider all edges as 1 if they are greater than 0
`force.recalc.degree`	force recalculation of penalty weights (but not the network), instead of going to cache
`force.recalc.network`	force recalculation of network and penalty weights, instead of going to cache
`n.cores`	number of cores to be used
`...`	extra parameters for cor function

Value

a vector of the degrees

Examples

```
n.col <- 6
xdata <- matrix(rnorm(n.col * 4), ncol = n.col)
degreeCor(xdata)
degreeCor(xdata, cutoff = .5)
degreeCor(xdata, cutoff = .5, consider.unweighted = TRUE)
```

`degreeCov` *Calculate the degree of the covariance network based on xdata*

Description

Calculate the degree of the covariance network based on xdata

Usage

```
degreeCov(xdata, cutoff = 0, consider.unweighted = FALSE,
  force.recalc.degree = FALSE, force.recalc.network = FALSE,
  n.cores = 1, ...)
```

Arguments

`xdata`	calculate correlation matrix on each column
`cutoff`	positive value that determines a cutoff value
`consider.unweighted`	consider all edges as 1 if they are greater than 0
`force.recalc.degree`	force recalculation of penalty weights (but not the network), instead of going to cache
`force.recalc.network`	force recalculation of network and penalty weights, instead of going to cache
`n.cores`	number of cores to be used
`...`	extra parameters for cov function

Value

a vector of the degrees

Examples

```
n.col <- 6
xdata <- matrix(rnorm(n.col * 4), ncol = n.col)
degreeCov(xdata)
degreeCov(xdata, cutoff = .5)
degreeCov(xdata, cutoff = .5, consider.unweighted = TRUE)
```

degreeSparsebn *Calculate degree of correlation matrix*

Description

Calculate degree of correlation matrix

Usage

```
degreeSparsebn(xdata, type = "continuous", levels = NULL, ivn = NULL,
  n = NULL, object = NULL, cutoff = 0, consider.unweighted = FALSE,
  n.cores = 1, show.message = FALSE, force.recalc.degree = FALSE,
  force.recalc.network = FALSE, ...)
```

Arguments

xdata	calculate correlation matrix on each column
type	either "discrete" or "continuous", see sparsebnUtils::sparsebnData
levels	(optional) list of levels for each node. see sparsebnUtils::sparsebnData
ivn	(optional) list of interventions for each observation, see sparsebnUtils::sparsebnData
n	(optional) number of rows from data matrix to print, see sparsebnUtils::sparsebnData
object	(optional) an object of type sparsebnData, see sparsebnUtils::sparsebnData
cutoff	positive value that determines a cutoff value
consider.unweighted	consider all edges as 1 if they are greater than 0
n.cores	number of cores to be used
show.message	shows cache operation messages
force.recalc.degree	force recalculation, instead of going to cache
force.recalc.network	force recalculation of network and penalty weights, instead of going to cache
...	parameters for sparsebn::estimate.dag

Value

a vector of the degrees

Examples

```
# generate a random matrix of observations
xdata <- matrix(rnorm(1000), nrow = 20)
degreeSparsebn(xdata)
```

ensemblGeneNames *Retrieve ensembl gene names from biomaRt*

Description

Retrieve ensembl gene names from biomaRt

Usage

```
ensemblGeneNames(gene.id)
```

Arguments

gene.id	character vector with gene names

Value

a dataframe with external gene names, ensembl_id

Examples

```
## Not run:
    ensemblGeneNames(c('MOB1A','RFLNB', 'SPIC', 'TP53'))

## End(Not run)
```

geneNames *Retrieve gene names from biomaRt*

Description

Retrieve gene names from biomaRt

Usage

```
geneNames(ensembl.genes)
```

Arguments

ensembl.genes	character vector with gene names in ensembl_id format

Value

a dataframe with external gene names, ensembl_id

Examples

```
geneNames(c('ENSG00000114978','ENSG00000166211', 'ENSG00000183688'))
```

glmDegree *GLMNET model penalizing nodes with small degree*

Description

This function overrides the 'trans.fun' options in 'network.options' with the inverse of a degree described in Veríssimo et al. (2015) that penalizes nodes with small degree.

Usage

```
glmDegree(xdata, ydata, network, network.options = networkOptions(), ...)
```

Arguments

xdata	input data, can be a matrix or MultiAssayExperiment
ydata	response data compatible with glmnet
network	type of network, see below
network.options	options to calculate network
...	parameters that glmnet accepts

Value

see glmNetSparse

See Also

glmNetSparse

Examples

```
xdata <- matrix(rnorm(100), ncol = 5)
glmDegree(xdata, rnorm(nrow(xdata)), 'correlation',
          family = 'gaussian',
          network.options = networkOptions(min.degree = .2))
```

glmHub *GLMNET model penalizing nodes with small degree*

Description

This function overrides the 'trans.fun' options in 'network.options' with an heuristic described in Veríssimo et al. that penalizes nodes with small degree.

Usage

```
glmHub(xdata, ydata, network, network.options = networkOptions(), ...)
```

Arguments

xdata	input data, can be a matrix or MultiAssayExperiment
ydata	response data compatible with glmnet
network	type of network, see below
network.options	options to calculate network
...	parameters that glmnet accepts

Value

see glmNetSparse

See Also

glmNetSparse

Examples

```
xdata <- matrix(rnorm(100), ncol = 5)
glmHub(xdata, rnorm(nrow(xdata)), 'correlation', family = 'gaussian',
       network.options = networkOptions(min.degree = .2))
```

---

glmOrphan *GLMNET model penalizing nodes with high degree*

Description

This function overrides the ‘trans.fun‘ options in ‘network.options‘ with an heuristic described in Veríssimo et al. that penalizes nodes with high degree.

Usage

```
glmOrphan(xdata, ydata, network, network.options = networkOptions(), ...)
```

Arguments

xdata	input data, can be a matrix or MultiAssayExperiment
ydata	response data compatible with glmnet
network	type of network, see below
network.options	options to calculate network
...	parameters that glmnet accepts

Value

see glmNetSparse

See Also

glmNetSparse

Examples

```
xdata <- matrix(rnorm(100), ncol = 5)
glmOrphan(xdata, rnorm(nrow(xdata)), 'correlation', family = 'gaussian',
          network.options = networkOptions(min.degree = .2))
```

---

glmSparseNet *Calculate GLM model with network-based regularization*

Description

network parameter accepts:

Usage

```
glmSparseNet(xdata, ydata, network, network.options = networkOptions(),
  experiment.name = NULL, ...)
```

Arguments

xdata	input data, can be a matrix or MultiAssayExperiment
ydata	response data compatible with glmnet
network	type of network, see below
network.options	options to calculate network
experiment.name	name of experiment to use as input in MultiAssayExperiment object (only if xdata is an object of this class)
...	parameters that glmnet accepts

Details

* string to calculate network based on data (correlation, covariance) * matrix representing the network * vector with already calculated penalty weights (can also be used directly with glmnet)

Value

an object just as glmnet

Examples

```
xdata <- matrix(rnorm(100), ncol = 20)
glmSparseNet(xdata, rnorm(nrow(xdata)), 'correlation', family = 'gaussian')
glmSparseNet(xdata, rnorm(nrow(xdata)), 'covariance', family = 'gaussian')

#
#
# Using MultiAssayExperiment
# load data
xdata <- MultiAssayExperiment::miniACC
# TODO aking out x indivudals missing values
# build valid data with days of last follow up or to event
```

```
event.ix <- which(!is.na(xdata$days_to_death))
cens.ix <- which(!is.na(xdata$days_to_last_followup))
xdata$surv_event_time <- array(NA, nrow(colData(xdata)))
xdata$surv_event_time[event.ix] <- xdata$days_to_death[event.ix]
xdata$surv_event_time[cens.ix] <- xdata$days_to_last_followup[cens.ix]
# Keep only valid individuals
valid.ix <- as.vector(!is.na(xdata$surv_event_time) &
                      !is.na(xdata$vital_status) &
                      xdata$surv_event_time > 0)
xdata.valid <- xdata[, rownames(colData(xdata))[valid.ix]]
ydata.valid <- colData(xdata.valid)[,c('surv_event_time', 'vital_status')]
colnames(ydata.valid) <- c('time', 'status')
glmSparseNet(xdata.valid,
             ydata.valid,
             family          = 'cox',
             network         = 'correlation',
             experiment.name = 'RNASeq2GeneNorm')
```

`hallmarks` *Retrieve hallmarks of cancer count for genes*

Description

Retrieve hallmarks of cancer count for genes

Usage

```
hallmarks(genes, metric = "count", hierarchy = "full",
  generate.plot = TRUE, show.message = FALSE)
```

Arguments

`genes`	gene names
`metric`	see below
`hierarchy`	see below
`generate.plot`	flag to indicate if return object has a ggplot2 object
`show.message`	flag to indicate if run.cache method shows messages

Value

data.frame with choosen metric and hierarchy It also returns a vector with genes that do not have any hallmarks.

See http://chat.lionproject.net/api for more details on the metric and hallmarks parameters

To standardize the colors in the gradient you can use scale_fill_gradientn(limits=c(0,1), colours=topo.colors(3)) to limit between 0 and 1 for cprob and -1 and 1 for npmi

Examples

```
## Not run:
    hallmarks(c('MOB1A', 'RFLNB', 'SPIC'))
    hallmarks(c('MOB1A', 'RFLNB', 'SPIC'), metric = 'cprob')

## End(Not run)
```

heuristicScale *Heuristic function to use in high dimensions*

Description

Heuristic function to use in high dimensions

Usage

```
heuristicScale(x, sub.exp10 = -1, exp.mult = -1, sub.exp = -1)
```

Arguments

x	vector of values to scale
sub.exp10	value to subtract to base 10 exponential, for example: ‘10^0 - sub.exp10 = 1 - sub.exp10‘
exp.mult	parameter to multiply exponential, i.e. to have a negative exponential or positive
sub.exp	value to subtract for exponentional, for example if x = 0, ‘exp(0) - sub.exp = 1 - sub.exp‘

Value

a vector of scaled values

Examples

```
heuristicScale(rnorm(1:10))
```

hubHeuristic *Heuristic function to penalize nodes with low degree*

Description

Heuristic function to penalize nodes with low degree

Usage

```
hubHeuristic(x)
```

Arguments

x	single value of vector

Value

transformed

Examples

```
hubHeuristic(rnorm(1:10))
```

networkCorParallel *Calculates the correlation network*

Description

Calculates the correlation network

Usage

```
networkCorParallel(xdata, build.output = "matrix", n.cores = 1,
  force.recalc.network = FALSE, show.message = FALSE, ...)
```

Arguments

xdata	base data to calculate network
build.output	if output returns a 'matrix', 'vector' of the upper triu without the diagonal or NULL with any other argument
n.cores	number of cores to be used
force.recalc.network	force recalculation, instead of going to cache
show.message	shows cache operation messages
...	extra parameters for fun

Value

depends on build.output parameter

Examples

```
n.col <- 6
xdata <- matrix(rnorm(n.col * 4), ncol = n.col)
networkCorParallel(xdata)
```

networkCovParallel *Calculates the covariance network*

Description

Calculates the covariance network

Usage

```
networkCovParallel(xdata, build.output = "matrix", n.cores = 1,
  force.recalc.network = FALSE, show.message = FALSE, ...)
```

Arguments

xdata	base data to calculate network
build.output	if output returns a 'matrix', 'vector' of the upper triu without the diagonal or NULL with any other argument
n.cores	number of cores to be used
force.recalc.network	force recalculation, instead of going to cache
show.message	shows cache operation messages
...	extra parameters for fun

Value

depends on build.output parameter

Examples

```
n.col <- 6
xdata <- matrix(rnorm(n.col * 4), ncol = n.col)
networkCovParallel(xdata)
```

---

networkOptions *Setup network options*

Description

Setup network options, such as using weighted or unweighted degree, which centrality measure to use

Usage

```
networkOptions(method = "pearson", unweighted = TRUE, cutoff = 0,
  centrality = "degree", min.degree = 0, n.cores = 1,
  trans.fun = function(x) {     x })
```

Arguments

method	in case of correlation and covariance, which method to use
unweighted	calculate degree using unweighted network
cutoff	cuttoff value in network edges to trim the network
centrality	centrality measure to use, currently only supports degree
min.degree	minimum value that individual penalty weight can take
n.cores	number of cores to use, default to 1
trans.fun	The trans.fun argument takes a function definition that will apply a transformation to the penalty vector calculated from the degree. This transformation allows to change how the penalty is applied. see below

Value

a list of options

See Also

glmOrphan glmDegree

Examples

```
networkOptions(unweighted = FALSE)
```

---

orphanHeuristic *Heuristic function to penalize nodes with high degree*

Description

Heuristic function to penalize nodes with high degree

Usage

```
orphanHeuristic(x)
```

Arguments

x	single value of vector

Value

transformed

Examples

```
orphanHeuristic(rnorm(1:10))
```

---

protein2EnsemblGeneNames *Retrieve ensembl gene ids from proteins*

Description

Retrieve ensembl gene ids from proteins

Usage

```
protein2EnsemblGeneNames(ensembl.proteins)
```

Arguments

ensembl.proteins	character vector with gene names in ensembl_peptide_id format

Value

a dataframe with external gene names, ensembl_peptide_id

Examples

```
protein2EnsemblGeneNames(c('ENSP00000235382',
                           'ENSP00000233944',
                           'ENSP00000216911'))
```

`separate2GroupsCox` *Separate data in High and Low risk groups (based on Cox model)*

Description

Draws multiple kaplan meyer survival curves (or just 1) and calculates logrank test

Usage

```
separate2GroupsCox(chosen.btas, xdata, ydata, probs = c(0.5, 0.5),
  no.plot = FALSE, plot.title = "SurvivalCurves", xlim = NULL,
  ylim = NULL, expand.yzero = FALSE, legend.outside = FALSE, ...)
```

Arguments

`chosen.btas`	list of testing coefficients to calculate prognostic indexes, for example "list(Age = some_vector)"
`xdata`	n x m matrix with n observations and m variables
`ydata`	Survival object
`probs`	How to separate high and low risk patients 50%-50% is the default, but for top and bottom 40% -> c(.4,.6)
`no.plot`	Only calculate p-value and do not generate survival curve plot
`plot.title`	Name of file if
`xlim`	Optional argument to limit the x-axis view
`ylim`	Optional argument to limit the y-axis view
`expand.yzero`	expand to y = 0
`legend.outside`	If TRUE legend will be outside plot, otherwise inside
`...`	additional parameters to survminer::ggsurvplot

Value

object with logrank test and kaplan-meier survival plot

A list with plot, p-value and kaplan-meier object. The plot was drawn from survminer::ggsurvplot with only the palette, data and fit arguments being defined and keeping all other defaults that can be customized as additional parameters to this function.

See Also

survminer::ggsurvplot

Examples

```
data('ovarian', package = 'survival')
xdata <- ovarian[,c('age', 'resid.ds')]
ydata <- data.frame(time = ovarian$futime, status = ovarian$fustat)
separate2GroupsCox(c(age = 1, 0), xdata, ydata)
separate2GroupsCox(c(age = 1, 0.5), xdata, ydata)
separate2GroupsCox(c(age = 1), c(1,0,1,0,1,0),
                   data.frame(time = runif(6), status = rbinom(6, 1, .5)))
separate2GroupsCox(list(aa = c(age = 1, 0.5),
                        bb = c(age = 0, 1.5)), xdata, ydata)
```

---

string.network.700.cache

Cache of protein-protein network, as it takes some time to retrieve and process this will facilitate the vignette building

---

Description

It was filtered with combined_scores and individual scores below 700 without text-based scores

Usage

```
string.network.700.cache
```

Format

An object of class `dgCMatrix` with 11033 rows and 11033 columns.

References

https://string-db.org/

Examples

```
head(string.network.700.cache)
```

---

stringDBhomoSapiens *Download protein-protein interactions from STRING DB*

Description

Download protein-protein interactions from STRING DB

Usage

```
stringDBhomoSapiens(version = "10", score_threshold = 0,
  remove.text = TRUE)
```

Arguments

version	version of the database to use
score_threshold	remove scores below threshold
remove.text	remove text mining-based scores

Value

a data.frame with rows representing an interaction between two proteins, and columns the count of scores above the given score_threshold

Examples

```
## Not run:
    stringDBhomoSapiens(score_threshold = 800)

## End(Not run)
```

Index