Package ‘glmSparseNet’

May 3, 2024

Type Package
Title Network Centrality Metrics for Elastic-Net Regularized Models
Version 1.22.0
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 ``gaussian'', ``poisson'', ``binomial'', ``multinomial'', ``cox'', and ``mgaussian''.

License GPL-3

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

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

Depends R (>= 4.3.0)
Imports biomaRt, checkmate, dplyr, forcats, future.logger, ggplot2, glue, httr, lifecycle, methods, parallel, readr, rlang, glmnet, Matrix, MultiAssayExperiment, SummarizedExperiment, survminer, TCGAutils, utils

Suggests BiocStyle, curatedTCGAData, knitr, magrittr, reshape2, pROC, rmarkdown, survival, testthat, VennDiagram, withr

VignetteBuilder knitr

RdMacros lifecycle

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

Encoding UTF-8
Language en-US
LazyData false
NeedsCompilation no
Roxygen list(markdown = TRUE)
RoxygenNote 7.3.1

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

git_branch RELEASE_3_19

git_last_commit 7b32a0f

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-03

Author André Veríssimo [aut, cre] (<https://orcid.org/0000-0002-2212-339X>),
Susana Vinga [aut],
Eunice Carrasquinha [ctb],
Marta Lopes [ctb]

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

Contents

- glmSparseNet-package ........................................... 3
- .baseDir ......................................................... 4
- .biomartLoad .................................................. 4
- .buildFunctionDigest ........................................... 5
- .cacheCompression ............................................. 6
- .calcPenalty .................................................... 7
- .calculateResult ............................................... 7
- .combinedScore ................................................. 8
- .createDirectoryForCache ...................................... 9
- .curlWorkaround .............................................. 9
- .degreeGeneric .................................................. 10
- .digestCache ................................................... 11
- .glmSparseNetPrivate ........................................... 12
- .networkGenericParallel ....................................... 13
- .networkWorker ............................................... 14
- .runCache ....................................................... 14
- .saveRunCache ................................................ 16
- .showMessage ................................................... 16
- .tempdirCache .................................................. 17
- .writeReadme ................................................... 17
- balancedCvFolds ............................................... 18
- buildLambda .................................................... 18
- buildStringNetwork ............................................ 20
- cv.glmDegree .................................................. 21
- degreeCor ....................................................... 24
- degreeCov ....................................................... 25
- downloadFileLocal ............................................. 27
glmSparseNet-package

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 "gaussian", "poisson", "binomial", "multinomial", "cox", and "mgaussian".

Author(s)

Maintainer: André Veríssimo <andre.verissimo@tecnico.ulisboa.pt> (ORCID)

Authors:

- Susana Vinga <susanavinga@tecnico.ulisboa.pt>

Other contributors:

- Eunice Carrasquinha <eunice.trigueirao@tecnico.ulisboa.pt> [contributor]
- Marta Lopes <marta.lopes@tecnico.ulisboa.pt> [contributor]
### .baseDir

**Description**

Change base dir for `.runCache`

**Usage**

```r
.baseDir(path = NULL)
```

**Arguments**

- `path` to base directory where cache is saved

**Value**

the new path

**Examples**

```r
glmSparseNet::.baseDir("/tmp/cache")
```

### .biomartLoad

**Description**

Common call to biomaRt to avoid repetitive code

**Usage**

```r
.biомartLoad(attributes, filters, values, useCache, verbose)
```
Arguments
attributes  Attributes you want to retrieve. A possible list of attributes can be retrieved using the function biomaRt::listAttributes.
filters  Filters (one or more) that should be used in the query. A possible list of filters can be retrieved using the function biomaRt::listFilters.
values  Values of the filter, e.g. vector of affy IDs. If multiple filters are specified then the argument should be a list of vectors of which the position of each vector corresponds to the position of the filters in the filters argument.
useCache  Boolean indicating if biomaRt cache should be used.
verbose  When using biomaRt in webservice mode and setting verbose to TRUE, the XML query to the webservice will be printed.

Value
data.frame with attributes as columns and values translated to them

See Also
geneNames  ensemblGeneNames  protein2EnsemblGeneNames  biomaRt::getBM()  biomaRt::useEnsembl()

Examples
glmSparseNet:::.biomartLoad(
    attributes = c("external_gene_name", "ensembl_gene_id"),
    filters = "external_gene_name",
    values = c("MOB1A", "RFLNB", "SPIC", "TP53"),
    useCache = TRUE,
    verbose = FALSE
)

Description
Build digest of function from the actual code

Usage
.buildFunctionDigest(fun)
Arguments

fun function call name

Value

a digest

Examples

```r
glmSparseNet::.buildFunctionDigest(sum)
glmSparseNet::.buildFunctionDigest(c)
```

Description

Change cache.compression for run_cache

Usage

```r
.cacheCompression(compression = NULL)
```

Arguments

compression see compression parameter in save function

Value

the new compression

Examples

```r
glmSparseNet::.cacheCompression("bzip2")
```
.calcPenalty

Description

Internal method to calculate the network using data-dependant methods

Usage

`.calcPenalty(xdata, penaltyType, options = networkOptions())`

Arguments

- `xdata`: input data
- `penaltyType`: which method to use
- `options`: options to be used

Value

vector with penalty weights

Examples

```r
xdata <- matrix(rnorm(1000), ncol = 200)
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")
```

.calculateResult

Description

This is where the actual work is done

Usage

`.calculateResult(path, compression, forceRecalc, showMessage, fun, ...)`
.combinedScore

Arguments

- path: path to save cache
- compression: compression used in save
- forceRecalc: force to recalculate cache
- showMessage: boolean to show messages
- fun: function to be called
- ...: arguments to said function

Value

- result of fun(...)

Examples

```r
glmSparseNet:::calculateResult(
  file.path(tempdir(), "calculate_result.Rdata"),
  "gzip",
  FALSE,
  TRUE,
  sum,
  1, 2, 3
)
```

Description

Please note that all the interactions have duplicates as it’s a two way interaction (score(ProteinA-Protein) == score(ProteinB, PorteinA))

Usage

```r
.combinedScore(allInteractions, scoreThreshold, removeText)
```

Arguments

- allInteractions: table with score of all interactions
- scoreThreshold: threshold to keep interactions
- removeText: remove text-based interactions

Details

To better understand how the score is calculated, please see: https://string-db.org/help/faq/#how-are-the-scores-computed
Value
table with combined score

Description
Create directories for cache

Usage
.createDirectoryForCache(baseDir, parentPath)

Arguments
baseDir   tentative base dir to create.
parentPath first 4 characters of digest that will become parent directory for the actual cache file (this reduces number of files per folder)

Value
a list of updated baseDir and parentDir

Examples
glmSparseNet:::.createDirectoryForCache(tempdir(), "abcd")

glmSparseNet:::.createDirectoryForCache(
   file.path(getwd(), "run-cache"), "abcd"
)

Description
Workaround for bug with curl when fetching specific ensembl mirror

Usage
.curlWorkaround(expr)
Arguments

`expr` expression

Value

result of expression

Examples

glmSparseNet:::.curlWorkaround({
  biomaRt::useEnsembl(
    biomart = "genes",
    dataset = "hsapiens_gene_ensembl"
  )
})

Description

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

Usage

```
.degreeGeneric(
  fun = stats::cor,
  funPrefix = "operator",
  xdata,
  cutoff = 0,
  considerUnweighted = FALSE,
  chunks = 1000,
  forceRecalcDegree = FALSE,
  forceRecalcNetwork = FALSE,
  nCores = 1,
  ...
)
```

Arguments

`fun` function that will calculate the edge weight between 2 nodes

`funPrefix` 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
considerUnweighted
  consider all edges as 1 if they are greater than 0
chunks
  calculate function at batches of this value (default is 1000)
forceRecalcDegree
  force recalculation of penalty weights (but not the network), instead of going to cache
forceRecalcNetwork
  force recalculation of network and penalty weights, instead of going to cache
nCores
  number of cores to be used
...
  extra parameters for fun

Value

  a vector of the degrees

---

**.digestCache**

*Default digest method*

---

**Description**

Sets a default caching algorithm to use with .runCache

**Usage**

`.digestCache(val)`

**Arguments**

- **val**: object to calculate hash over

**Value**

- a hash of the sha256

**Examples**

`glmSparseNet:::.digestCache(c(1, 2, 3, 4, 5))`

`glmSparseNet:::.digestCache("some example")`
Description

Calculate GLM model with network-based regularization

Usage

.glmSparseNetPrivate(fun, xdata, ydata, network, experiment = NULL, 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 when xdata is a MultiAssayExperiment object this parameter is required
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**

```r
.networkGenericParallel(
  fun,
  funPrefix,
  xdata,
  buildOutput = "matrix",
  nCores = 1,
  forceRecalcNetwork = FALSE,
  showMessage = FALSE,
  ...
)
```

**Arguments**

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

**Value**

depends on buildOutput parameter
.networkWorker

Worker to calculate edge weight for each pair of ixI node and following

Description
Note that it assumes it does not calculate for index below and equal to ixI

Usage
.networkWorker(fun, xdata, ixI, ...)

Arguments
- **fun**: function to be used, can be cor, cov or any other defined function
- **xdata**: original data to calculate the function over
- **ixI**: starting index, this can be used to save only upper triu
- **...**: extra parameters for fun

Value
a vector with size ncol(xdata) - ixI

.runCache

Run function and save cache

Description
This method saves the function that’s being called

Usage
.runCache(
  fun,
  ..., 
  seed = NULL, 
  baseDir = NULL, 
  cachePrefix = "generic_cache", 
  cacheDigest = list(), 
  showMessage = NULL, 
  forceRecalc = FALSE, 
  addToHash = NULL
)

## S4 method for signature 'function'
.runCache

.fun(
  fun,
  ..., 
  seed = NULL,
  baseDir = NULL,
  cachePrefix = "generic_cache",
  cacheDigest = list(),
  showMessage = NULL,
  forceRecalc = FALSE,
  addToHash = NULL
)

Arguments

fun  function call name
...
seed  when function call is random, this allows to set seed beforehand
baseDir  directory where data is stored
cachePrefix  prefix for file name to be generated from parameters (...)
cacheDigest  cache of the digest for one or more of the parameters
showMessage  show message that data is being retrieved from cache
forceRecalc  force the recalculation of the values
addToHash  something to add to the filename generation

Value

the result of fun(...)

Functions

• .runCache('function'): accepts function as first argument and save cache

Examples

# [optional] save cache in a temporary directory
#
glmSparseNet:::.baseDir(tempdir())
glmSparseNet:::.runCache(c, 1, 2, 3, 4)
#
# next three should use the same cache
# note, the middle call should be a little faster as digest is not
# calculated
# for the first argument
glmSparseNet:::.runCache(c, 1, 2, 3, 4)
glmSparseNet:::.runCache(c, a = 1, 2, c = 3, 4)

# Using a local folder
# glmSparseNet:::.runCache(c, 1, 2, 3, 4, baseDir = "runcache")
.saveRunCache  Saving the cache

Description

Saving the cache

Usage

.saveRunCache(result, path, compression, showMessage)

Arguments

- **result**: main result to save
- **path**: path to the file to save
- **compression**: compression method to be used
- **showMessage**: TRUE to show messages, FALSE otherwise

Value

result of save operation

Examples

```r
glmSparseNet:::saveRunCache(
  35, file.path(tempdir(), "save_run_cache.Rdata"), FALSE, TRUE
)
```

.showMessage  Show messages option in .runCache

Description

Show messages option in .runCache

Usage

.showMessage(showMessage = NULL)

Arguments

- **showMessage**: boolean indicating to show messages or not

Value

the show.message option
Examples

```r
glmSparseNet:::.showMessage(FALSE)
```

---

**.tempdirCache**

*Temporary directory for runCache*

**Description**

Temporary directory for runCache

**Usage**

```r
.tempdirCache()
```

**Value**

a path to a temporary directory used by runCache

---

**.writeReadme**

*Write a file in run-cache directory to explain the origin*

**Description**

Write a file in run-cache directory to explain the origin

**Usage**

```r
.writeReadme(baseDir)
```

**Arguments**

- `baseDir` directory where to build this file

**Value**

the path to the file it has written

**Examples**

```r
glmSparseNet:::.writeReadme(tempdir())
```
balancedCvFolds  
Create balanced folds for cross validation using stratified sampling

**Description**
Create balanced folds for cross validation using stratified sampling

**Usage**

```r
balancedCvFolds(..., nfolds = 10)
```

# deprecated, please use balancedCvFolds()

```r
balanced.cv.folds(..., nfolds = 10)
```

**Arguments**

- `...` vectors representing data
- `nfolds` number of folds to be created

**Value**

list with given input, nfolds and result. The result is a list matching the input with foldid attributed to each position.

**Examples**

```r
balancedCvFolds(seq(10), seq(11, 15), nfolds = 2)
```

# will give a warning

```r
balancedCvFolds(seq(10), seq(11, 13), nfolds = 10)
```  
```r
balancedCvFolds(seq(100), seq(101, 133), nfolds = 10)
```

---

**buildLambda**  
Auxiliary function to generate suitable lambda parameters

**Description**

Auxiliary function to generate suitable lambda parameters
Usage

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

Arguments

- `lambdaLargest`: 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
- `ordersOfMagnitudeSmaller`: minimum value for lambda (lambda.largest / 10^{orders.of.magnitude.smaller})
- `lambdaPerOrderMagnitude`: how many lambdas to create for each order of magnitude
- `lambda.largest` [Deprecated]
- `orders.of.magnitude.smaller` [Deprecated]
- `lambda.per.order.magnitude` [Deprecated]

Value

a numeric vector with suitable lambdas

Examples

```r
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(
  stringTbl, 
  useNames = c("protein", "ensembl", "external"),
  string.tbl = deprecated(),
  use.names = deprecated()
)
```

Arguments

- `stringTbl` data.frame or tibble with colnames and rownames as ensembl peptide id (same order).
- `useNames` character(1) that defaults to use protein names _("protein"), other options are _"ensembl" for ensembl gene id or _"external" for external gene names.
- `string.tbl` [Deprecated]
- `use.names` [Deprecated]

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

```
interactions <- stringDBhomoSapiens(scoreThreshold = 100)
string_network <- buildStringNetwork(interactions)

# number of edges
sum(string_network != 0)
```
**cv.glmDegree**  
*Calculate cross validating GLM model with network-based regularization*

**Description**

Network parameter accepts:

**Usage**

```r
cv.glmDegree(
  xdata,
  ydata,
  network,
  options = networkOptions(),
  experiment = NULL,
  network.options = deprecated(),
  experiment.name = deprecated(),
  ...
)
```

```r
cv.glmHub(
  xdata,
  ydata,
  network,
  options = networkOptions(),
  experiment = NULL,
  network.options = deprecated(),
  experiment.name = deprecated(),
  ...
)
```

```r
cv.glmOrphan(
  xdata,
  ydata,
  network,
  options = networkOptions(),
  experiment = NULL,
  network.options = deprecated(),
  experiment.name = deprecated(),
  ...
)
```

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

```r
options = networkOptions(),
experiment = NULL,

network.options = deprecated(),
experiment.name = deprecated(),
...
```

Arguments

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

Functions

- `cv.glmDegree()`: penalizes nodes with small degree (inversion penalization \( h(x) = 1 / x \)).
- `cv.glmHub()`: penalizes nodes with small degree (normalized heuristic that promotes nodes with many edges).
- `cv.glmOrphan()`: penalizes nodes with high degree (normalized heuristic that promotes nodes with few edges).

See Also

Model with the same penalizations `glmSparseNet`.
Examples

# Degree penalization

```r
data <- matrix(rnorm(100), ncol = 5)
cv.glmDegree(
  data,
  rnorm(nrow(data)),
  "correlation",
  family = "gaussian",
  nfolds = 5,
  options = networkOptions(minDegree = .2)
)
```

# Hub penalization

```r
data <- matrix(rnorm(100), ncol = 5)
cv.glmHub(
  data,
  rnorm(nrow(data)),
  "correlation",
  family = "gaussian",
  nfolds = 5,
  options = networkOptions(minDegree = .2)
)
```

# Orphan penalization

```r
data <- matrix(rnorm(100), ncol = 5)
cv.glmOrphan(
  data,
  rnorm(nrow(data)),
  "correlation",
  family = "gaussian",
  nfolds = 5,
  options = networkOptions(minDegree = .2)
)
```

# Gaussian model

```r
data <- matrix(rnorm(500), ncol = 5)
cv.glmSparseNet(
  data, rnorm(nrow(data)), "correlation",
  family = "gaussian"
)
cv.glmSparseNet(
  data, rnorm(nrow(data)), "covariance",
  family = "gaussian"
)
```

# Using MultiAssayExperiment with survival model

```r
library(MultiAssayExperiment)
data("miniACC", package = "MultiAssayExperiment")
```
xdata <- 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 = "RNASeq2GeneNorm"
)

degreeCor <- function(xdata, cutoff = 0, considerUnweighted = FALSE, forceRecalcDegree = FALSE, forceRecalcNetwork = FALSE, nCores = 1, ...)
   consider.unweighted = deprecated(),
   family = "cox",
   network = "correlation",
   experiment = "RNASeq2GeneNorm"

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,
  considerUnweighted = FALSE,
  forceRecalcDegree = FALSE,
  forceRecalcNetwork = FALSE,
  nCores = 1,
  ...
)
force.recalc.degree = deprecated()
force.recalc.network = deprecated()
n.cores = deprecated()
)

Arguments

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

Value

- a vector of the degrees.

Examples

```r
n.col <- 6
oxdata <- matrix(rnorm(n.col * 4), ncol = n.col)
degreeCor(xdata)
degreeCor(xdata, cutoff = .5)
degreeCor(xdata, cutoff = .5, considerUnweighted = 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,
  considerUnweighted = FALSE,
  forceRecalcDegree = FALSE,
  forceRecalcNetwork = FALSE,
  nCores = 1,
  ...
)

Arguments

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

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, considerUnweighted = TRUE)
downloadFileLocal

Download files to local temporary path

Description
In case of new call it uses the temporary cache instead of downloading again.

Usage
downloadFileLocal(urlStr, oD = tempdir())

Arguments
urlStr url of file to download
oD temporary directory to store file

Details
Inspired by STRINGdb Bioconductor package, but using curl as file may be too big to handle.

Value
path to file

Examples

glmSparseNet::downloadFileLocal(
  "https://string-db.org/api/tsv-no-header/version"
)

ensemblGeneNames

Retrieve ensembl gene names from biomaRt

Description
Retrieve ensembl gene names from biomaRt

Usage
ensemblGeneNames(
geneId,
useCache = TRUE,
verbose = FALSE,
gene.id = deprecated(),
use.cache = deprecated()
)
Arguments

geneId character vector with gene names
useCache Boolean indicating if biomaRt cache should be used
verbose When using biomaRt in webservice mode and setting verbose to TRUE, the XML query to the webservice will be printed.
gene.id [Deprecated]
use.cache [Deprecated]

Value

a dataframe with external gene names, ensembl_id

Examples

ensemblGeneNames(c("MOB1A", "RFLNB", "SPIC", "TP53"))

geneNames

Retrieve gene names from biomaRt

Description

Retrieve gene names from biomaRt

Usage

geneNames(
  ensemblGenes,
  useCache = TRUE,
  verbose = FALSE,
  ensembl.genes = deprecated(),
  use.cache = deprecated()
)

Arguments

ensemblGenes character vector with gene names in ensembl_id format
useCache Boolean indicating if biomaRt cache should be used
verbose When using biomaRt in webservice mode and setting verbose to TRUE, the XML query to the webservice will be printed.
ensembl.genes [Deprecated]
use.cache [Deprecated]

Value

a dataframe with external gene names, ensembl_id
Examples

geneNames(c("ENSG00000114978", "ENSG00000166211", "ENSG00000183688"))

glmSparseNet

Calculate GLM model with network-based regularization

Description

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)

Usage

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

glmDegree(
  xdata,
  ydata,
  network,
  options = networkOptions(),
  experiment = NULL,
  network.options = deprecated(),
  experiment.name = deprecated(),
  ...
)

glmHub(
  xdata,
  ydata,
  network,
  options = networkOptions(),
  experiment = NULL,
  network.options = deprecated(),
  experiment.name = deprecated(),
  ...
glmOrphan(
  xdata,
  ydata,
  network,
  options = networkOptions(),
  experiment = NULL,
  network.options = deprecated(),
  experiment.name = deprecated(),
  ...
)

Arguments

**xdata**  
input data, can be a matrix or MultiAssayExperiment.

**ydata**  
response data compatible with glmnet.

**network**  
type of network, see below.

**options**  
options to calculate network.

**experiment**  
name of experiment to use as input in MultiAssayExperiment object (only if xdata is an object of this class).

**network.options**  
[Deprecated]

**experiment.name**  
[Deprecated]

...  
parameters that `glmnet::glmnet()` accepts.

Value

an object just as glmnet

Functions

- `glmDegree()`: penalizes nodes with small degree (*inversion penalization* \( h(x) = 1 / x \)).
- `glmHub()`: Penalizes nodes with small degree (*normalized heuristic that promotes nodes with many edges*).
- `glmOrphan()`: Penalizes nodes with high degree (*normalized heuristic that promotes nodes with few edges*).

See Also

Cross-validation functions `cv.glmSparseNet()`.
Examples

```r
data <- 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
library(MultiAssayExperiment)
data("miniACC", package = "MultiAssayExperiment")

xdata <- miniACC
# TODO asking out x individuals 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 = "RNASeq2GeneNorm"
)

# Degree penalization
xdata <- matrix(rnorm(100), ncol = 5)
glmDegree(
  xdata,
  rnorm(nrow(xdata)),
  "correlation",
  family = "gaussian",
  options = networkOptions(minDegree = .2)
)
```

hallmarks

Retrieve hallmarks of cancer count for genes

Description

[Defunct] The API has been removed and this function is no longer available.

Usage

```r
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
**heuristicScale**

*Heuristic function to use in high dimensions*

**Description**

Heuristic function to use in high dimensions

**Usage**

```r
heuristicScale(
  x,
  subExp10 = -1,
  expMult = -1,
  subExp = -1,
  sub.exp10 = deprecated(),
  exp.mult = deprecated(),
  sub.exp = deprecated()
)
```

**Arguments**

- `x` vector of values to scale
- `subExp10` value to subtract to base 10 exponential, for example: \(10^0 - \text{subExp10} = 1 - \text{subExp10}\)
- `expMult` parameter to multiply exponential, i.e. to have a negative exponential or positive
- `subExp` value to subtract for exponential, for example if \(x = 0\), \(\exp(0) - \text{sub.exp} = 1 - \text{sub.exp}\)
- `sub.exp10` [Deprecated]
- `exp.mult` [Deprecated]
- `sub.exp` [Deprecated]

**Value**

- a vector of scaled values

**Examples**

```r
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))`

**myColors**  
*Custom pallete of colors*

**Description**  
Custom pallete of colors

**Usage**  
`myColors(ix = NULL)`

# deprecated, please use `myColors()`  
`my.colors(ix = NULL)`

**Arguments**  
- `ix`: index for a color

**Value**  
a color

**Examples**  
`myColors()`  
`myColors(5)`
mySymbols

Custom palette of symbols in plots

Description

Custom palette of symbols in plots

Usage

mySymbols(ix = NULL)

# deprecated, please use mySymbols()
my.symbols(ix = NULL)

Arguments

ix index for symbol

Value

a symbol

Examples

mySymbols()
mySymbols(2)

networkCorParallel

Calculates the correlation network

Description

Calculates the correlation network

Usage

networkCorParallel(
  xdata,
  buildOutput = "matrix",
  nCores = 1,
  forceRecalcNetwork = FALSE,
  showMessage = FALSE,
  ...,
  build.output = deprecated(),
  n.cores = deprecated(),
  force.recalc.network = deprecated(),
  show.message = deprecated()
)
networkCovParallel

Arguments

- **xdata**: base data to calculate network
- **buildOutput**: if output returns a 'matrix', 'vector' of the upper triu without the diagonal or NULL with any other argument
- **nCores**: number of cores to be used
- **forceRecalcNetwork**: force recalculation, instead of going to cache
- **showMessage**: shows cache operation messages
- **...**: extra parameters for fun
- **build.output**: lifecycle::badge("deprecated") without the diagonal or NULL with any other argument
- **n.cores**: lifecycle::badge("deprecated")
- **force.recalc.network**: lifecycle::badge("deprecated")
- **show.message**: lifecycle::badge("deprecated")

Value

depends on build.output parameter

Examples

```r
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

```r
networkCovParallel(
  xdata,
  buildOutput = "matrix",
  nCores = 1,
  forceRecalcNetwork = FALSE,
  showMessage = FALSE,
  ...
)
```
Arguments

- **xdata**: base data to calculate network
- **buildOutput**: if output returns a 'matrix', 'vector' of the upper triu without the diagonal or NULL with any other argument
- **nCores**: number of cores to be used
- **forceRecalcNetwork**: force recalculation, instead of going to cache
- **showMessage**: shows cache operation messages
- **...**: extra parameters for fun
- **build.output**: lifecycle::badge("deprecated") without the diagonal or NULL with any other argument
- **n.cores**: lifecycle::badge("deprecated")
- **force.recalc.network**: lifecycle::badge("deprecated")
- **show.message**: lifecycle::badge("deprecated")

Value

depends on build.output parameter

Examples

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

Description

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

Usage

```r
networkOptions(
  method = "pearson",
  unweighted = TRUE,
  cutoff = 0,
  centrality = "degree",
  minDegree = 0,
  nCores = 1,
  transFun = function(x) x,
  min.degree = deprecated(),
)```
orphanHeuristic

n.cores = deprecated(),
trans.fun = deprecated()
)

Arguments

method in case of correlation and covariance, which method to use.
unweighted calculate degree using unweighted network.
cutoff cutoff value in network edges to trim the network.
centrality centrality measure to use, currently only supports degree.
minDegree minimum value that individual penalty weight can take.
nCores number of cores to use, default to 1.
transFun See details below.
min_degree [Deprecated]
n.cores [Deprecated]
trans.fun [Deprecated]
The transFun 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.

Value

a list of options

See Also

glmOrphan() and 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
protein2EnsemblGeneNames

Value
transformed

Examples
orphanHeuristic(rnorm(1:10))

protein2EnsemblGeneNames

Retrieve ensembl gene ids from proteins

Description
Retrieve ensembl gene ids from proteins

Usage
protein2EnsemblGeneNames(
  ensemblProteins,
  useCache = TRUE,
  verbose = FALSE,
  ensembl.proteins = deprecated(),
  use.cache = deprecated()
)

Arguments
ensemb1Proteins  character vector with gene names in ensembl.peptide_id format
useCache        Boolean indicating if biomaRt cache should be used
verbose         When using biomaRt in webservice mode and setting verbose to TRUE, the
                 XML query to the webservice will be printed.
ensemb1.proteins [Deprecated]
use.cache       [Deprecated]

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(
  chosenBetas,
  xdata,
  ydata,
  probs = c(0.5, 0.5),
  noPlot = FALSE,
  plotTitle = "SurvivalCurves",
  xlim = NULL,
  ylim = NULL,
  expandYZero = FALSE,
  legendOutside = FALSE,
  stopWhenOverlap = TRUE,
  ..., 
  chosen.btas = deprecated(),
  no.plot = deprecated(),
  plot.title = deprecated(),
  expand.yzero = deprecated(),
  legend.outside = deprecated(),
  stop.when.overlap = deprecated()
)

Arguments

chosenBetas 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).
noPlot Only calculate p-value and do not generate survival curve plot.
plotTitle Name of file if.
xlim Optional argument to limit the x-axis view.
ylim Optional argument to limit the y-axis view.
expandYZero expand to y = 0.
legendOutside If TRUE legend will be outside plot, otherwise inside.
stopWhenOverlap
when probs vector allows for overlapping of samples in both groups, then stop.

... additional parameters to survminer::ggsurvplot

chosen.betas [Deprecated]
no.plot [Deprecated]
plot.title [Deprecated]
expand.yzero [Deprecated]
legend.outside [Deprecated]
stop.when.overlap [Deprecated]

Otherwise it will calculate with duplicate samples, i.e. simply adding them to xdata and ydata (in a different group).

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

```r
xdata <- survival::ovarian[, c("age", "resid.ds")]
ydata <- data.frame(
  time = survival::ovarian$futime,
  status = survival::ovarian$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

data('string.network.700.cache', package = 'glmSparseNet')

Format

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

References

https://string-db.org/

stringDBhomoSapiens  Download protein-protein interactions from STRING DB

Description

Download protein-protein interactions from STRING DB

Usage

stringDBhomoSapiens(
    version = "11.0",
    scoreThreshold = 0,
    removeText = TRUE,
    score_threshold = deprecated(),
    remove.text = deprecated()
)

Arguments

version version of the database to use
scoreThreshold remove scores below threshold
removeText remove text mining-based scores
score_threshold [Deprecated]
remove.text [Deprecated]
Value

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

Examples

stringDBhomoSapiens(scoreThreshold = 800)
Index

* data
  string.network.700.cache, 42
* internal
  .glmSparseNetPrivate, 12
  glmSparseNet-package, 3
  .baseDir, 4
  .biomartLoad, 4
  .buildFunctionDigest, 5
  .cacheCompression, 6
  .calcPenalty, 7
  .calculateResult, 7
  .combinedScore, 8
  .createDirectoryForCache, 9
  .curlWorkaround, 9
  .degreeGeneric, 10
  .digestCache, 11
  .glmSparseNetPrivate, 12
  .networkGenericParallel, 13
  .networkWorker, 14
  .runCache, 14
  .runCache, function-method (.runCache), 14
  .saveRunCache, 16
  .showMessage, 16
  .tempdirCache, 17
  .writeReadme, 17

balanced.cv.folds (balancedCvFolds), 18
balancedCvFolds, 18
buildLambda, 18
buildStringNetwork, 20

cv.glmDegree, 21
cv.glmHub (cv.glmDegree), 21
cv.glmOrphan (cv.glmDegree), 21
cv.glmSparseNet (cv.glmDegree), 21

cv.glmSparseNet(), 30
degreeCor, 24
degreeCov, 25
downloadFileLocal, 27
ensemb1GeneNames, 27
geneNames, 28
glmDegree (glmSparseNet), 29
glmDegree(), 38
glmHub (glmSparseNet), 29
glmnet::cv.glmnet(), 22
glmnet::glmnet(), 30
glmOrphan (glmSparseNet), 29
glmOrphan(), 38
glmSparseNet, 29
glmSparseNet(), 22
glmSparseNet-package, 3

hallmarks, 32
heuristicScale, 33
hubHeuristic, 34

my.colors (myColors), 34
my.symbols (mySymbols), 35
myColors, 34
mySymbols, 35

networkCorParallel, 35
networkCovParallel, 36
networkOptions, 37

orphanHeuristic, 38

protein2EnsemblGeneNames, 39

separate2GroupsCox, 40
string.network.700.cache, 42
stringDBhomoSapiens, 42
stringDBhomoSapiens(), 20
survminer::ggsurvplot(), 41