Package ‘glmSparseNet’

January 10, 2024

Type     Package
Title    Network Centrality Metrics for Elastic-Net Regularized Models
Version  1.20.0

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
Encoding UTF-8
LazyData false

NeedsCompilation no

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

Depends R (>= 4.1), Matrix, MultiAssayExperiment, glmnet
Imports SummarizedExperiment, biomaRt, futile.logger, futile.options, forcats, utils, dplyr, glue, readr, digest, httr, ggplot2, survminer, reshape2, stringr, parallel, methods

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

VignetteBuilder knitr

RoxygenNote 7.1.2

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_18

git_last_commit  31bb472

git_last_commit_date  2023-10-24

Repository  Bioconductor 3.18

Date/Publication  2024-01-10

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 3
- .degreeGeneric 4
- .glmSparseNetPrivate 5
- .networkGenericParallel 6
- .networkWorker 7
- balanced.cv.folds 7
- base.dir 8
- biomart.load 8
- build.function.digest 9
- buildLambda 10
- buildStringNetwork 11
- cache.compression 12
- calculate.combined.score 12
- calculate.result 13
- create.directory.for.cache 14
- curl.workaround 14
- cv.glmDegree 15
- cv.glmHub 16
- cv.glmOrphan 17
- cv.glmSparseNet 18
- degreeCor 19
- degreeCov 20
- digest.cache 21
- downloadFileLocal 22
- ensemblGeneNames 23
- geneNames 23
- glmDegree 24
- glmHub 25
- glmOrphan 25
- glmSparseNet 26
- glmSparseNet.options 28
- hallmarks 28
- heuristicScale 29
.calcPenalty

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

```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')
```

\_.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 symmetric 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 too 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

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
...

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,
  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 too 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 only upper triu
- ...: extra parameters for fun

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

balanced.cv.folds
Create balanced folds for cross validation

Description
Create balanced folds for cross validation

Usage
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
glmSparseNet:::balanced.cv.folds(seq(10), seq(11, 15), nfolds = 2)
# will give a warning
glmSparseNet:::balanced.cv.folds(seq(10), seq(11, 13), nfolds = 10)
glmSparseNet:::balanced.cv.folds(seq(100), seq(101, 133), nfolds = 10)
```

---

### base.dir

`change base.dir for run.cache`

---

**Description**

change base.dir for run.cache

**Usage**

```r
base.dir(path = NULL)
```

**Arguments**

- `path` to base directory where cache is saved

**Value**

the new path

**Examples**

```r
glmSparseNet:::base.dir('/tmp/cache')
```

---

### biomart.load

*Common call to biomaRt to avoid repetitive code*

---

**Description**

Common call to biomaRt to avoid repetitive code

**Usage**

```r
biomart.load(attributes, filters, values, use.cache, verbose)
```
build.function.digest

Build digest of function from the actual code

Description

Build digest of function from the actual code

Usage

build.function.digest(fun)
Arguments

fun function call name

Value

a digest

Examples

glmSparseNet:::build.function.digest(sum)
glmSparseNet:::build.function.digest(c)

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
**buildStringNetwork**

*Build gene network from peptide ids*

**Examples**

```
buildLambda(5.4)
```

---

**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**

```
all.interactions.700 <- stringDBhomoSapiens(score_threshold = 700)
string.network <- buildStringNetwork(all.interactions.700, use.names = 'external')

# number of edges
sum(string.network != 0)
```
**cache.compression**  
*change cache.compression for run.cache*

**Description**  
*change cache.compression for run.cache*

**Usage**  
```r  
cache.compression(compression = NULL)  
```

**Arguments**  
- **compression**  
  see compression parameter in save function

**Value**  
*the new compression*

**Examples**  
```r  
glmSparseNet::cache.compression('bzip2')  
```

**calculate.combined.score**  
*Calculate combined score for STRINGdb interactions*

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

**Usage**  
```r  
calculate.combined.score(all.interactions, score_threshold, remove.text)  
```

**Arguments**  
- **all.interactions**  
  table with score of all interactions  
- **score_threshold**  
  threshold to keep interactions  
- **remove.text**  
  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

calculate.result Calculate/load result and save if necessary

Description
This is where the actual work is done

Usage
calculate.result(path, compression, force.recalc, show.message, fun, ...)

Arguments
path path to save cache
compression compression used in save
force.recalc force to recalculate cache
show.message boolean to show messages
fun function to be called
... arguments to said function

Value
result of fun(...) 

Examples

glmSparseNet:::calculate.result(
  file.path(tempdir(),'calculate.result.Rdata'),
  'gzip',
  FALSE,
  TRUE,
  sum,
  1, 2, 3
)
create.directory.for.cache

Create directories for cache

Description

Create directories for cache

Usage

create.directory.for.cache(base.dir, parent.path)

Arguments

base.dir  tentative base dir to create.
parent.path  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 base.dir and parent.dir

Examples

```
glmSparseNet:::create.directory.for.cache(tempdir(), 'abcd')
glmSparseNet:::create.directory.for.cache(file.path(getwd(), 'run-cache'), 'abcd')
```

curl.workaround

Workaround for bug with curl when fetching specific ensembl mirror

Description

Should be solved in issue #39, will test to remove it.

Usage

curl.workaround(expr)

Arguments

expr  expression
Value

result of expression

Examples

```r
glmSparseNet:::curl.workaround({
  biomaRt::useEnsembl(
    biomart = "genes",
    dataset = 'hsapiens_gene_ensembl')
})
```

---

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

```r
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

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

```r
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**

```r
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**

```
# 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')

# Using MultiAssayExperiment with survival model

data('miniACC', package='MultiAssayExperiment')
xdata <- miniACC

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]

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', 
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,
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

da 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)
```

digest.cache

Default digest method

Description

Sets a default caching algorithm to use with run.cache

Usage

```
digest.cache(val)
```
downloadFileLocal

**Arguments**

- `val`: object to calculate hash over

**Value**

- a hash of the sha256

**Examples**

```R
glmSparseNet:::digest.cache(c(1,2,3,4,5))
glmSparseNet:::digest.cache('some example')
```

downloadFileLocal  **Download files to local temporary path**

**Description**

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

**Usage**

```R
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**

```R
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**

```r
ensemblGeneNames(gene.id, use.cache = TRUE, verbose = FALSE)
```

**Arguments**

- `gene.id`: character vector with gene names
- `use.cache`: 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**

a dataframe with external gene names, ensembl_id

**Examples**

```r
ensemblGeneNames(c('MOB1A', 'RFLNB', 'SPIC', 'TP53'))
```

---

**geneNames**

*Retrieve gene names from biomaRt*

**Description**

Retrieve gene names from biomaRt

**Usage**

```r
geneNames(ensembl.genes, use.cache = TRUE, verbose = FALSE)
```

**Arguments**

- `ensembl.genes`: character vector with gene names in ensembl_id format
- `use.cache`: 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.
### 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

```r
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

```r
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 a heuristic described in Veríssimo et al. that penalizes nodes with small degree.

**Usage**
```r
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**
```r
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 a heuristic described in Veríssimo et al. that penalizes nodes with high degree.

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

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

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

```r
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
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',
```
glmSparseNet.options  Constants for 'glmSparseNet' package

Description

Log level constants and the logger options.

Usage

glmSparseNet.options(..., simplify = FALSE, update = list())

Arguments

...  TODO
simplify  TODO
update  pair list of update to options

Details

The logging configuration is managed by 'glmSparseNet.options', a function generated by OptionsManager within 'futile.options'.

Value

futile.options::OptionsManager object

See Also

futile.options

hallmarks

Retrieve hallmarks of cancer count for genes

Usage

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

heuristicScale

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 chosen 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

```r
hallmarks(c('MOB1A', 'RFLNB', 'SPIC'))

hallmarks(c('MOB1A', 'RFLNB', 'SPIC'), metric = 'cprob')
```

heuristicScale

Heuristic function to use in high dimensions

Description

Heuristic function to use in high dimensions

Usage

```r
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 exponential, 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))

my.colors

Custom pallete of colors

Description
Custom pallete of colors

Usage
my.colors(ix = NULL)

Arguments
ix index for a color

Value
a color

Examples
my.colors()
my.colors(5)
my.symbols

Custom palette of symbols in plots

Description

Custom palette of symbols in plots

Usage

my.symbols(ix = NULL)

Arguments

ix

index for symbol

Value

a symbol

Examples

my.symbols()

my.symbols(2)

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,
  ...
)

networkCovParallel

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

```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,
  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
networkOptions

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 cutoff 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

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.

trans.fun 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, use.cache = TRUE, verbose = FALSE)
Arguments

- `ensembl.proteins` (character vector): gene names in ensembl_peptide_id format
- `use.cache` (Boolean): whether to use biomaRt cache
- `verbose` (when using biomaRt in webservice mode with verbose = TRUE): XML query to the webservice will be printed.

Value

A dataframe with external gene names, ensembl_peptide_id

Examples

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

---

### run.cache

#### Run function and save cache

**Description**

This method saves the function that’s being called.

**Usage**

```r
run.cache(
  fun,
  ..., seed = NULL,
  base.dir = NULL,
  cache.prefix = "generic_cache",
  cache.digest = list(),
  show.message = NULL,
  force.recalc = FALSE,
  add.to.hash = NULL
)
```

**Arguments**

- `fun` (function call name)
- `...` (parameters for function call)
- `seed` (when function call is random; allows setting seed beforehand)
- `base.dir` (directory where data is stored)
cache.prefix  prefix for file name to be generated from parameters (...)
cache.digest  cache of the digest for one or more of the parameters
show.message  show message that data is being retrieved from cache
force.recalc  force the recalculation of the values
add.to.hash   something to add to the filename generation

Value
the result of fun(...)

Examples

# [optional] save cache in a temporary directory
#
glmSparseNet:::base.dir(tempdir())
glmSparseNet:::run.cache(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:::run.cache(c, 1, 2, 3, 4)
glmSparseNet:::run.cache(c, a=1, 2, c= 3, 4)

# Using a local folder
# glmSparseNet:::run.cache(c, 1, 2, 3, 4, base.dir = "runcache")

run.cache(function-method

Run function and save cache

Description
Run function and save cache

Usage

## S4 method for signature `'function'`
run.cache(  
  fun,
  ...,  
  seed = NULL,
  base.dir = NULL,
  cache.prefix = "generic_cache",
  cache.digest = list(),
  show.message = NULL,

run.cache.function-method

Run function and save cache

Description
Run function and save cache

Usage

## S4 method for signature `'function'`
run.cache(  
  fun,
  ...,  
  seed = NULL,
  base.dir = NULL,
  cache.prefix = "generic_cache",
  cache.digest = list(),
  show.message = NULL,
```r
force.recalc = FALSE,
add.to.hash = NULL
)
```

**Arguments**

- **fun**: function call name
- **...**: parameters for function call
- **seed**: when function call is random, this allows to set seed beforehand
- **base.dir**: directory where data is stored
- **cache.prefix**: prefix for file name to be generated from parameters (...)
- **cache.digest**: cache of the digest for one or more of the parameters
- **show.message**: show message that data is being retrieved from cache
- **force.recalc**: force the recalculation of the values
- **add.to.hash**: something to add to the filename generation

**Value**

the result of fun(...)

**Examples**

```r
# [optional] save cache in a temporary directory
#
glmSparseNet:::base.dir(tempdir())
glmSparseNet:::run.cache(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:::run.cache(c, 1, 2, 3, 4)
glmSparseNet:::run.cache(c, a=1, 2, c= 3, 4)

# Using a local folder
# glmSparseNet:::run.cache(c, 1, 2, 3, 4, base.dir = "runcache")
```

---

**Description**

Saving the cache
Usage

\[
\text{save.run.cache}(\text{result}, \text{path}, \text{compression}, \text{show.message})
\]

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{result}</td>
<td>main result to save</td>
</tr>
<tr>
<td>\text{path}</td>
<td>path to the file to save</td>
</tr>
<tr>
<td>\text{compression}</td>
<td>compression method to be used</td>
</tr>
<tr>
<td>\text{show.message}</td>
<td>TRUE to show messages, FALSE otherwise</td>
</tr>
</tbody>
</table>

Value

result of save operation

Examples

```r
glmSparseNet:::save.run.cache(
  35, file.path(tempdir(), 'save.run.cache.Rdata'), FALSE, TRUE
)
```

---

`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

```r
separate2GroupsCox(
  \text{chosen.btas},
  \text{xdata},
  \text{ydata},
  \text{probs} = \text{c}(0.5, 0.5),
  \text{no.plot} = \text{FALSE},
  \text{plot.title} = "SurvivalCurves",
  \text{xlim} = \text{NULL},
  \text{ylim} = \text{NULL},
  \text{expand.yzero} = \text{FALSE},
  \text{legend.outside} = \text{FALSE},
  \text{stop.when.overlap} = \text{TRUE},
  ...
)
```
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

stop.when.overlap when probs vector allows for overlapping of samples in both groups, then stop. Otherwise it will calculate with duplicate samples, i.e. simply adding them to xdata and ydata (in a different group)

... 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('cancer', package = 'survival')
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)
show.message  

*Show messages option in run.cache*

### Description
Show messages option in run.cache

### Usage
```
show.message(show.message = NULL)
```

### Arguments
- `show.message`: boolean indicating to show messages or not

### Value
the show.message option

### Examples
```
glmSparseNet::show.message(FALSE)
```

---

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/](https://string-db.org/)
stringDBhomoSapiens  Download protein-protein interactions from STRING DB

**Description**

Download protein-protein interactions from STRING DB

**Usage**

```r
stringDBhomoSapiens(version = "11.0", 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**

```r
stringDBhomoSapiens(score_threshold = 800)
```

---

tempdir.cache  Temporary directory for runCache

**Description**

Temporary directory for runCache

**Usage**

```r
tempdir.cache()
```

**Value**

a path to a temporary directory used by runCache
write.readme

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

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

Usage
write.readme(base.dir)

Arguments
base.dir directory where to build this file

Value
the path to the file it has written

Examples
glmSparseNet::write.readme(tempdir())
Index

* data
  - string.network.700.cache, 40
  - .calcPenalty, 3
  - .degreeGeneric, 4
  - .glmSparseNetPrivate, 5
  - .networkGenericParallel, 6
  - .networkWorker, 7
  - balanced.cv.folds, 7
  - base.dir, 8
  - biomart.load, 8
  - build.function.digest, 9
  - buildLambda, 10
  - buildStringNetwork, 11
  - cache.compression, 12
  - calculate.combined.score, 12
  - calculate.result, 13
  - create.directory.for.cache, 14
  - curl.workaround, 14
  - cv.glmDegree, 15
  - cv.glmHub, 16
  - cv.glmOrphan, 17
  - cv.glmSparseNet, 18
  - degreeCor, 19
  - degreeCov, 20
  - digest.cache, 21
  - downloadFileLocal, 22
  - ensemblGeneNames, 23
  - geneNames, 23
  - glmDegree, 24
  - glmHub, 25
  - glmOrphan, 25
  - glmSparseNet, 26
  - glmSparseNet.options, 28
  - hallmarks, 28
  - heuristicScale, 29
  - hubHeuristic, 30
  - my.colors, 30
  - my.symbols, 31
  - networkCorParallel, 31
  - networkCovParallel, 32
  - networkOptions, 33
  - orphanHeuristic, 34
  - protein2EnsemblGeneNames, 34
  - run.cache, 35
  - run.cache.function-method, 36
  - save.run.cache, 37
  - separate2GroupsCox, 38
  - show.message, 40
  - string.network.700.cache, 40
  - stringDBhomoSapiens, 41
  - tempdir.cache, 41
  - write.readme, 42