Package 'pulsar'

July 23, 2025

Title Parallel Utilities for Lambda Selection along a Regularization Path

Version 0.3.11

Encoding UTF-8

Description Model selection for penalized graphical models using the Stability Approach to Regularization Selection ('StARS'), with options for speed-ups including Bounded StARS (B-StARS), batch computing, and other stability metrics (e.g., graphlet stability G-StARS). Christian L. Müller, Richard Bonneau, Zachary Kurtz (2016) <doi:10.48550/arXiv.1605.07072>.

URL https://github.com/zdk123/pulsar, https://arxiv.org/abs/1605.07072

BugReports https://github.com/zdk123/pulsar/issues

Depends R (>= 3.2.0)

License GPL (>= 3)

Suggests batchtools (>= 0.9.10), fs (>= 1.2.2), checkmate (>= 1.8.5), orca, huge, MASS, clime, glmnet, network, cluster, testthat, knitr, rmarkdown

Imports methods, parallel, graphics, stats, utils, tools, Matrix (>= 1.5)

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Zachary Kurtz [aut, cre], Christian Müller [aut, ctb]

Maintainer Zachary Kurtz <zdkurtz@gmail.com>

Repository CRAN

Date/Publication 2023-09-24 18:00:02 UTC

Contents

`pulsar-package` *The pulsar package*

Description

Graphical model selection with the pulsar package

Details

This package provides methods to select a sparse, undirected graphical model by choosing a penalty parameter (lambda or λ) among a list of ordered values of lambda. We use an implementation of the Stability Approach to Regularization Selection (StARS, see references) inspired by the **huge** package.

However, **pulsar** includes some major differences from other R packages for graphical model estimation and selection (**glasso**, **huge**, **QUIC**, **XMRF**, **clime**, **flare**, etc). The underlying graphical model is computed by passing a function as an argument to `pulsar`. Thus, any algorithm for penalized graphical models can be used in this framework (see `pulsar-function` for more details), including those from the above packages. **pulsar** brings computational experiments under one roof by separating subsampling and calculation of summary criteria from the user-specified core model. The typical workflow in **pulsar** is to perform subsampling first (via the `pulsar`) and then refit the model on the full dataset using `refit`.

Previous StARS implementations can be inefficient for large graphs or when many subsamples are required. `pulsar` can compute upper and lower bounds on the regularization path for the StARS criterion after only 2 subsamples which makes it possible to neglect lambda values that are far from the desired StARS regularization parameter, reducing computation time for the rest of the $N - 2$ subsamples (Bounded StARS (B-StARS)).

We also implement additional subsampling-based graph summary criteria which can be used for more informed model selection. For example, we have shown that induced subgraph (graphlet) stability (G-StARS) improves empirical performance over StARS but other criteria are also offered.

Subsampling amounts to running the specified core model for N independent computations. Using the **batchtools** framework, we provide a simple wrapper, `batch.pulsar`, for running `pulsar` in embarrassingly parallel mode in an hpc environment. Summary criteria are computed using a Map/Reduce strategy, which lowers memory footprint for large models.

References

Müller, C. L., Bonneau, R. A., & Kurtz, Z. D. (2016).Generalized Stability Approach for Regularized Graphical Models.arXiv: https://arxiv.org/abs/1605.07072.

See Also

`pulsar-function`, `pulsar`, `batch.pulsar`

`batch.pulsar` *pulsar: batch mode*

Description

Run pulsar using stability selection, or another criteria, to select an undirected graphical model over a lambda-path.

Usage

```
batch.pulsar(
  data,
  fun = huge::huge,
  fargs = list(),
  criterion = c("stars"),
  thresh = 0.1,
  subsample.ratio = NULL,
  lb.stars = FALSE,
  ub.stars = FALSE,
  rep.num = 20,
  seed = NULL,
  wkdir = getwd(),
  regdir = NA,
  init = "init",
  conffile = "",
  job.res = list(),
  cleanup = FALSE,
  refit = TRUE
)
```

Arguments

data	A $n * p$ matrix of data matrix input to solve for the $p * p$ graphical model
fun	pass in a function that returns a list representing $p * p$ sparse, undirected graphical models along the desired regularization path. The expected inputs to this function are: a data matrix input and a sequence of decreasing lambdas and must return a list or S3 object with a member *named* path. This should be a list of adjacency matrices for each value of lambda. See pulsar-function for more information.
fargs	arguments to argument fun. Must be a named list and requires at least one member lambda, a numeric vector with values for the penalty parameter.
criterion	A character vector of selection statistics. Multiple criteria can be supplied. Only StARS can be used to automatically select an optimal index for the lambda path. See details for additional statistics.
thresh	threshold (referred to as scalar β in StARS publication) for selection criterion. Only implemented for StARS. thresh=0.1 is recommended.
subsample.ratio	determine the size of the subsamples (referred to as $b(n)/n$). Default is 10*sqrt(n)/n for n > 144 or 0.8 otherwise. Should be strictly less than 1.
lb.stars	Should the lower bound be computed after the first $N = 2$ subsamples (should result in considerable speedup and only implemented if stars is selected). If this option is selected, other summary metrics will only be applied to the smaller lambda path.
ub.stars	Should the upper bound be computed after the first $N = 2$ subsamples (should result in considerable speedup and only implemented if stars is selected). If this option is selected, other summary metrics will only be applied to the smaller lambda path. This option is ignored if the lb.stars flag is FALSE.
rep.num	number of random subsamples N to take for graph re-estimation. Default is $N = 20$, but more is recommended for non-StARS criteria or if using edge frequencies as confidence scores.
seed	A numeric seed to force predictable subsampling. Default is NULL. Use for testing purposes only.
wkdir	set the working directory if different than getwd
regdir	directory to store intermediate batch job files. Default will be a tempory directory
init	text string appended to basename of the regdir path to store the batch jobs for the initial StARS variability estimate (ignored if 'regdir' is NA)
conffile	path to or string that identifies a batchtools configuration file. This argument is passed directly to the name argument of the findConfFile function. See that help for detailed explanation.
job.res	named list of resources needed for each job (e.g. for PBS submission script). The format and members depends on configuration and template. See examples section for a Torque example
cleanup	Flag for removing batchtools registry files. Recommended FALSE unless you're sure intermediate data shouldn't be saved.
refit	Boolean flag to refit on the full dataset after pulsar is run. (see also refit)

Value

an S3 object of class `batch.pulsar` with a named member for each stability criterion/metric. Within each of these are:

- summary: the summary criterion over `rep.num` graphs at each value of lambda
- criterion: the stability metric
- merge: the raw criterion merged over the `rep.num` graphs (constructed from `rep.num` subsamples), prior to summarization
- opt.ind: index (along the path) of optimal lambda selected by the criterion at the desired threshold. Will return 0 if no optimum is found or `NULL` if selection for the criterion is not implemented.

If `stars` is included as a criterion then additional arguments include

- lb.index: the lambda index of the lower bound at $N = 2$ samples if `lb.stars` flag is set to TRUE
- ub.index: the lambda index of the upper bound at $N = 2$ samples if `ub.stars` flag is set to TRUE

reg: Registry object. See `batchtools::makeRegistry`

id: Identifier for mapping graph estimation function. See `batchtools::batchMap`

call: the original function call

References

Müller, C. L., Bonneau, R., & Kurtz, Z. (2016). Generalized Stability Approach for Regularized Graphical Models. arXiv https://arxiv.org/abs/1605.07072

Liu, H., Roeder, K., & Wasserman, L. (2010). Stability approach to regularization selection (stars) for high dimensional graphical models. Proceedings of the Twenty-Third Annual Conference on Neural Information Processing Systems (NIPS).

Zhao, T., Liu, H., Roeder, K., Lafferty, J., & Wasserman, L. (2012). The huge Package for High-dimensional Undirected Graph Estimation in R. The Journal of Machine Learning Research, 13, 1059–1062.

Michel Lang, Bernd Bischl, Dirk Surmann (2017). batchtools: Tools for R to work on batch systems. The Journal of Open Source Software, 2(10). URL https://doi.org/10.21105/joss.00135.

See Also

`pulsar` `refit`

Examples

```
## Not run:
## Generate the data with huge:
library(huge)
set.seed(10010)
p <- 400 ; n <- 1200
dat   <- huge.generator(n, p, "hub", verbose=FALSE, v=.1, u=.3)
```

```
lams  <- getLamPath(.2, .01, len=40)
hugeargs  <- list(lambda=lams, verbose=FALSE)

## Run batch.pulsar using snow on 5 cores, and show progress.
options(mc.cores=5)
options(batchtools.progress=TRUE, batchtools.verbose=FALSE)
out <- batch.pulsar(dat$data, fun=huge::huge, fargs=hugeargs,
                rep.num=20, criterion='stars', conffile='snow')
## Run batch.pulsar on a Torque cluster
## Give each job 1gb of memory and a limit of 30 minutes
resources <- list(mem="1GB", nodes="1", walltime="00:30:00")
out.p <- batch.pulsar(dat$data, fun=huge::huge, fargs=hugeargs,
                rep.num=100, criterion=c('stars', 'gcd'), conffile='torque'
                job.res=resources, regdir=file.path(getwd(), "testtorq"))
plot(out.p)
## take a look at the default torque config and template files we just used
file.show(findConfFile('torque'))
file.show(findTemplateFile('simpletorque'))

## End(Not run)
```

`estrada.class` *Estrada class*

Description

Estrada proposes that graphs can be classified into four different classes. We call this the Estrada class. These are: I. Expander-like II. Cluster III. Core-Periphery IV. Mixed.

Usage

```
estrada.class(G, evthresh = 0.001)
```

Arguments

`G`	a $p * p$ adjacency matrix of a Graph
`evthresh`	tolerance for a zero eigenvalue

Value

Estrada class $(1 - 4)$

References

Estrada, E. (2007). Topological structural classes of complex networks. Physical Review E - Statistical, Nonlinear, and Soft Matter Physics, 75(1), 1-12. doi:10.1103/PhysRevE.75.016103

`findConfFile` *find config file*

Description

Find a default config file. First calls `batchtools::findConfFile` and then find a pulsar default.

Usage

```
findConfFile(name = "")
```

Arguments

`name`	name of default config or path to config file.

Details

See the batchtools functions `batchtools::findConfFile` and `batchtools::makeRegistry`. When calling `batch.pulsar`, we attempt to use batchtool's default lookup for a config file before calling `pulsar::findConfFile`.

For clusters with a queuing submission system, a template file, for defining worker node resources and executing the batch R code, will need to be defined somewhere on the system. See `findTemplateFile`.

See Also

`findTemplateFile`

Examples

```
## Default config file provided by pulsar runs code in interactive mode
## This is for testing purposes and executes serially.
findConfFile()
## Use the parallel package
## slower than providing the 'ncores' argument to pulsar function, due to
## the overhead of creating the batchtools registry.
findConfFile('parallel')

## Use the snow package to register/execute batch jobs on socket clusters.
findConfFile('snow')
## Use a TORQUE / PBS queing system. Requires brew template file.
findConfFile('torque')
findTemplateFile('simpletorque')
```

findTemplateFile *find template file*

Description

Find a config file from batchtools or default file from pulsar

Usage

```
findTemplateFile(name)
```

Arguments

name	name of default template or path to template file.

Details

See the batchtools functions batchtools::findTemplateFile, batchtools::makeClusterFunctionsTORQUE, batchtools::makeClusterFunctionsSGE, etc, to employ batchtools' default lookup scheme for template files. Supply the output of this function to the template argument to override batchtools' default.

In this case we look for "[name].tmpl" in the pulsar installation directory in the subfolder "templates".

See Also

findConfFile

Examples

```
 ## Not run:
 cluster.functions = batchtools::makeClusterFunctionsTORQUE(
                     template=pulsar::findTemplateFile('simpletorque'))

## End(Not run)
```

gcvec *Graphlet correlation vector*

Description

Compute graphlet correlations over the desired orbits (default is 11 non-redundant orbits of graphlets of size <=4) for a single graph G

Usage

```
gcvec(G, orbind = c(0, 2, 5, 7, 8, 10, 11, 6, 9, 4, 1) + 1)
```

Arguments

G	a $p * p$ adjacency matrix (dense or sparse) of a graph.
orbind	index vector for which orbits to use for computing pairwise graphlet correlations. Default is from Yaveroğlu et al, 2014 (see References), but 1 offset needed for R-style indexing.

References

Hočevar, T., & Demšar, J. (2014). A combinatorial approach to graphlet counting. Bioinformatics (Oxford, England), 30(4), 559–65. doi:10.1093/bioinformatics/btt717

Yaveroğlu, Ö. N., Malod-Dognin, N., Davis, D., Levnajic, Z., Janjic, V., Karapandza, R., … Pržulj, N. (2014). Revealing the hidden language of complex networks. Scientific Reports, 4, 4547. doi:10.1038/srep04547

---

get.opt.index *Get or evaluate an optimal index*

Description

If the optimal index for the lambda path is not already assigned, then use a validated method to select the optimal index of the lambda path for alternate criteria (i.e. other than StARS).

Usage

```
get.opt.index(obj, criterion = "gcd", ...)
```

Arguments

obj	the pulsar/batch.pulsar object to evaluate
criterion	a character argument for the desired summary criterion
...	Ignored

Details

Automated optimal index selection is [currently] only implemented for gcd (graphlet stability).

Criterion:

- gcd: Select the minimum gcd summary score within the lower and upper StARS bounds.

Value

index of the lambda path

See Also

opt.index

getEnvir *Get calling environment*

Description

Generic S3 method for extracting an environment from an S3 object. A getter for an explicitly stored environment from an S3 object or list... probably the environment where the original function that created the object was called from. The default method is a wrapper for `x$envir`.

Usage

```
getEnvir(x)

## Default S3 method:
getEnvir(x)
```

Arguments

`x`	S3 object to extract the environment

See Also

`getCall`, `environment`, `parent.env`, `eval`

getLamPath *Lambda path*

Description

Generate a lambda path sequence in descending order, equally or log-spaced.

Usage

```
getLamPath(max, min, len, log = FALSE)
```

Arguments

`max`	numeric, maximum lambda value
`min`	numeric, minimum lambda value
`len`	numeric/int, length of lambda path
`log`	logical, should the lambda path be log-spaced

Value

numeric vector of lambdas

See Also

getMaxCov

Examples

```
## Generate the data with huge:
library(huge)
set.seed(10010)
p <- 40 ; n <- 100
dat   <- huge.generator(n, p, "hub", verbose=FALSE, v=.1, u=.3)

## Theoretical lamda max is the maximum abs value of the empirical covariance matrix
maxCov <- getMaxCov(dat$data)
lams   <- getLamPath(maxCov, 5e-2*maxCov, len=40)
```

---

getMaxCov *Max value of cov*

Description

Get the maximum [absolute] value of a covariance matrix.

Usage

```
getMaxCov(x, cov = isSymmetric(x), abs = TRUE, diag = FALSE)
```

Arguments

x	A matrix/Matrix of data or covariance
cov	Flag if x is a covariance matrix, Set False is x is an nxp data matrix. By default, if x is symmetric, assume it is a covariance matrix.
abs	Flag to get max absolute value
diag	Flag to include diagonal entries in the max

Details

This function is useful to determine the theoretical value for lambda_max - for Gaussian data, but may be a useful starting point in the general case as well.

See Also

getLamPath

graph.diss *Graph dissimilarity*

Description

Dissimilarity matrix of a graph is here defined as the number of neighbors shared by any two nodes.

Usage

```
graph.diss(G, sim = FALSE, loops = FALSE)
```

Arguments

G	a $p * p$ adjacency matrix (dense or sparse) of a graph.
sim	Flag to return Graph similarity instead (1-dissimilarity)
loops	Flag to consider self loops

Value

a $p * p$ dissimilarity matrix

References

Bochkina, N. (2015). Selection of the Regularization Parameter in Graphical Models using a Priori Knowledge of Network Structure, arXiv: 1509.05326.

natural.connectivity *Natural Connectivity*

Description

Compute the natural connectivity of a graph

Usage

```
natural.connectivity(G, eig = NULL, norm = TRUE)
```

Arguments

G	a $p * p$ adjacency matrix (dense or sparse) of a graph. Ignored if eig is given
eig	precomputed list of eigen vals/vectors (output from eigen). If NULL, compute for G.
norm	should the natural connectivity score be normalized

Details

The natural connectivity of a graph is a useful robustness measure of complex networks, corresponding to the average eigenvalue of the adjacency matrix.

Value

numeric natural connectivity score

References

Jun, W., Barahona, M., Yue-Jin, T., & Hong-Zhong, D. (2010). Natural Connectivity of Complex Networks. Chinese Physics Letters, 27(7), 78902. doi:10.1088/0256-307X/27/7/078902

`opt.index` *Optimal index*

Description

Get or set the optimal index of the lambda path, as determined by a given criterion. `value` must be a numeric/int.

Usage

```
opt.index(obj, criterion = "gcd")

opt.index(obj, criterion = names(value)) <- value
```

Arguments

`obj`	a pulsar or batch.pulsar object
`criterion`	a summary statistic criterion for lambda selection. If value is not named, default to gcd.
`value`	Integer index for optimal lambda by criterion

See Also

`get.opt.index`

plot.pulsar *Plot a* pulsar *S3 object*

Description

Plot a pulsar S3 object

Usage

```
## S3 method for class 'pulsar'
plot(x, scale = TRUE, invlam = FALSE, loglam = FALSE, legends = TRUE, ...)
```

Arguments

x	a pulsar or batch.pulsar object
scale	Flag to scale non-StARS criterion to max StARS value (or 1)
invlam	Flag to plot 1/lambda
loglam	Flag to plot log[lambda]
legends	Flag to plot legends
...	ignored

Details

If both invlam and loglam are given, log[1/lambda] is plotted

print.pulsar *Print a* pulsar *and* batch.pulsar *S3 object*

Description

Print information about the model, path length, graph dimension, criterion and optimal indices, if defined.

Usage

```
## S3 method for class 'pulsar'
print(x, ...)

## S3 method for class 'batch.pulsar'
print(x, ...)
```

Arguments

x	a fitted pulsar or batch.pulsar object
...	ignored

print.pulsar.refit *Print a* pulsar.refit *S3 object*

Description

Print information about the model, path length, graph dimension, criterion and optimal indices and graph sparsity.

Usage

```
## S3 method for class 'pulsar.refit'
print(x, ...)
```

Arguments

x	a pulsar.refit. output from refit
...	ignored

pulsar *pulsar: serial or parallel mode*

Description

Run pulsar using StARS' edge stability (or other criteria) to select an undirected graphical model over a lambda path.

Usage

```
pulsar(
  data,
  fun = huge::huge,
  fargs = list(),
  criterion = c("stars"),
  thresh = 0.1,
  subsample.ratio = NULL,
  rep.num = 20,
  seed = NULL,
  lb.stars = FALSE,
  ub.stars = FALSE,
  ncores = 1,
  refit = TRUE
)
```

Arguments

`data`	A $n * p$ matrix of data matrix input to solve for the $p * p$ graphical model
`fun`	pass in a function that returns a list representing $p * p$ sparse, undirected graphical models along the desired regularization path. The expected inputs to this function are: a data matrix input and a sequence of decreasing lambdas and must return a list or S3 object with a member *named* `path`. This should be a list of adjacency matrices for each value of `lambda`. See `pulsar-function` for more information.
`fargs`	arguments to argument `fun`. Must be a named list and requires at least one member `lambda`, a numeric vector with values for the penalty parameter.
`criterion`	A character vector of selection statistics. Multiple criteria can be supplied. Only StARS can be used to automatically select an optimal index for the lambda path. See details for additional statistics.
`thresh`	threshold (referred to as scalar β in StARS publication) for selection criterion. Only implemented for StARS. `thresh=0.1` is recommended.
`subsample.ratio`	determine the size of the subsamples (referred to as $b(n)/n$). Default is 10*sqrt(n)/n for n > 144 or 0.8 otherwise. Should be strictly less than 1.
`rep.num`	number of random subsamples N to take for graph re-estimation. Default is $N = 20$, but more is recommended for non-StARS criteria or if using edge frequencies as confidence scores.
`seed`	A numeric seed to force predictable subsampling. Default is NULL. Use for testing purposes only.
`lb.stars`	Should the lower bound be computed after the first $N = 2$ subsamples (should result in considerable speedup and only implemented if stars is selected). If this option is selected, other summary metrics will only be applied to the smaller lambda path.
`ub.stars`	Should the upper bound be computed after the first $N = 2$ subsamples (should result in considerable speedup and only implemented if stars is selected). If this option is selected, other summary metrics will only be applied to the smaller lambda path. This option is ignored if the lb.stars flag is FALSE.
`ncores`	number of cores to use for subsampling. See `batch.pulsar` for more parallelization options.
`refit`	Boolean flag to refit on the full dataset after pulsar is run. (see also `refit`)

Details

The options for `criterion` statistics are:

- stars (Stability approach to regularization selection)
- gcd (Graphet correlation distance, requires the **orca** package) see `gcvec`
- diss (Node-node dissimilarity) see `graph.diss`
- estrada (estrada class) see `estrada.class`
- nc (natural connectivity) see `natural.connectivity`
- sufficiency (Tandon & Ravikumar's sufficiency statistic)

Value

an S3 object of class `pulsar` with a named member for each stability metric run. Within each of these are:

- summary: the summary statistic over `rep.num` graphs at each value of lambda
- criterion: the stability criterion used
- merge: the raw statistic over the `rep.num` graphs, prior to summarization
- opt.ind: index (along the path) of optimal lambda selected by the criterion at the desired threshold. Will return 0 if no optimum is found or `NULL` if selection for the criterion is not implemented.

If `stars` is included as a criterion then additional arguments include

- lb.index: the lambda index of the lower bound at $N = 2$ samples if `lb.stars` flag is set to TRUE
- ub.index: the lambda index of the upper bound at $N = 2$ samples if `ub.stars` flag is set to TRUE

call: the original function call

References

Müller, C. L., Bonneau, R., & Kurtz, Z. (2016). Generalized Stability Approach for Regularized Graphical Models. arXiv. https://arxiv.org/abs/1605.07072

Liu, H., Roeder, K., & Wasserman, L. (2010). Stability approach to regularization selection (stars) for high dimensional graphical models. Proceedings of the Twenty-Third Annual Conference on Neural Information Processing Systems (NIPS).

Zhao, T., Liu, H., Roeder, K., Lafferty, J., & Wasserman, L. (2012). The huge Package for High-dimensional Undirected Graph Estimation in R. The Journal of Machine Learning Research, 13, 1059–1062.

See Also

`batch.pulsar` `refit`

Examples

```
## Not run:
## Generate the data with huge:
library(huge)
p <- 40 ; n <- 1200
dat   <- huge.generator(n, p, "hub", verbose=FALSE, v=.1, u=.3)
lams  <- getLamPath(getMaxCov(dat$data), .01, len=20)

## Run pulsar with huge
hugeargs <- list(lambda=lams, verbose=FALSE)
out.p <- pulsar(dat$data, fun=huge::huge, fargs=hugeargs,
                rep.num=20, criterion='stars')
```

```
## Run pulsar in bounded stars mode and include gcd metric:
out.b <- pulsar(dat$data, fun=huge::huge, fargs=hugeargs,
                rep.num=20, criterion=c('stars', 'gcd'),
                lb.stars=TRUE, ub.stars=TRUE)
plot(out.b)

## End(Not run)
```

`pulsar-function` *Graphical model functions for pulsar*

Description

Correctly specify a function for graphical model estimation that is compatible with the pulsar package.

Details

It is easy to construct your own function for penalized model estimation that can be used with this package. The R function must have correctly specified inputs and outputs and is passed into the `fun` argument to `pulsar` or `batch.pulsar`. Any function that does not follow these rules will fail to give the desired output and may trigger an error.

These packages on CRAN have functions that work out of the box, so you won't need to construct a wrapper:

~function~	~package~
huge	huge
sugm	flare

Inputs:

The function may take arbitrary, named arguments but the first argument must be the data $n * p$ data matrix with the n samples in rows and p features in the columns. At least one argument must be named "lambda", which is expected to be a decreasing numeric vector of penalties. The non-data arguments should be passed into `pulsar` or `batch.pulsar` as a named list (the names must match function arguments exactly) to the `fargs` argument.

Outputs:

The output from the function must be a list or another S3 object inherited from a list. At least one member must be named `path`. This `path` object itself must be a list of $p * p$ adjacency matrices, one for each value of lambda. Each cell in the adjacency matrix contains a 1 or TRUE if there is an edge between two nodes or 0/FALSE otherwise. It is highly recommended (though not enforced by **pulsar**) that each adjacency matrix be a column-oriented, compressed, sparse matrix from the **Matrix** package. For example, `dgCMatrix`/`dsCMatrix` (general/symmetric numeric Matrix) or the 1-bit `lgCMatrix`/`lsCMatrix` classes. The function may return other named outputs, but these will be ignored.

References

Müller, C. L., Bonneau, R. A., & Kurtz, Z. D. (2016). Generalized Stability Approach for Regularized Graphical Models. arXiv: https://arxiv.org/abs/1605.07072.

See Also

pulsar, batch.pulsar, **huge**, **Matrix**

Examples

```
## Generate a hub example
 dat <- huge::huge.generator(100, 40, 'hub', verbose=FALSE)

## Simple correlation thresholding
corrthresh <- function(data, lambda) {
  S <- cor(data)
  path <- lapply(lambda, function(lam) {
    tmp <- abs(S) > lam
    diag(tmp) <- FALSE
    as(tmp, 'lMatrix')
  })
  list(path=path)
}

## Inspect output
lam <- getLamPath(getMaxCov(dat$sigmahat), 1e-4, 10)
out.cor  <- pulsar(dat$data, corrthresh, fargs=list(lambda=lam))
out.cor

## Not run:
## Additional examples
## quic
library(QUIC)
quicr <- function(data, lambda, ...) {
    S    <- cov(data)
    est  <- QUIC(S, rho=1, path=lambda, msg=0, tol=1e-2, ...)
    est$path <-  lapply(seq(length(lambda)), function(i) {
                   ## convert precision array to adj list
                   tmp <- est$X[,,i]; diag(tmp) <- 0
                 as(tmp!=0, "lMatrix")
    })
    est
}
## clime
library(clime)
climer <- function(data, lambda, tol=1e-5, ...) {
     est <- clime(data, lambda, ...)
     est$path <- lapply(est$Omegalist, function(x) {
                     diag(x) <- 0
                     as(abs(x) > tol, "lMatrix")
                 })
     est
```

```
}

## inverse cov shrinkage Schafer and Strimmer, 2005
library(corpcor)
icovshrink <- function(data, lambda, tol=1e-3, ...) {
     path <- lapply(lambda, function(lam) {
                     tmp <- invcov.shrink(data, lam, verbose=FALSE)
                     diag(tmp) <- 0
                     as(abs(tmp) > tol, "lMatrix")
                 })
     list(path=path)
}

## Penalized linear model, only
library(glmnet)
lasso <- function(data, lambda, respind=1, family="gaussian", ...) {
         n <- length(lambda)
         tmp <- glmnet(data[,-respind], data[,respind],
                                 family=family, lambda=lambda, ...)
         path <-lapply(1:n, function(i) as(tmp$beta[,i,drop=FALSE], "lMatrix"))
         list(path=path)
}

## alternative stability selection (DIFFERENT from hdi package)
out <- pulsar(dat$data, lasso, fargs=list(lambda=lam))
mergmat <- do.call('cbind', tmp$stars$merge)
image(mergmat)

## End(Not run)
```

`refit` *Refit pulsar model*

Description

Run the supplied graphical model function on the whole dataset and refit with the selected lambda(s)

Usage

```
refit(obj, criterion)
```

Arguments

`obj`	a fitted `pulsar` or `batch.pulsar` object
`criterion`	a character vector of criteria for refitting on full data. An optimal index must be defined for each criterion or a message will displayed. If missing (no argument is supplied), try to refit for all pre-specified criteria.

Details

The `refit` call is evaluated in the environment specified by the `pulsar` or `batch.pulsar` object, so if any variables were used for arguments to the original call, unless they are purposefully updated, should not be altered. For example, if the variable for the original data is reassigned, the output of `refit` will not be on the original dataset.

Value

a `pulsar.refit` S3 object with members:

- est: the raw output from the graphical model function, `fun`, applied to the full dataset.
- refit: a named list of adjacency matrices, for each optimal criterion in `obj` or specified in the `criterion` argument.
- fun: the original function used to estimate the graphical model along the lambda path.

See Also

`pulsar` `batch.pulsar`

Examples

```
## Generate the data with huge:
## Not run:
library(huge)
set.seed(10010)
p <- 40 ; n <- 1200
dat   <- huge.generator(n, p, "hub", verbose=FALSE, v=.1, u=.3)
lams  <- getLamPath(getMaxCov(dat$data), .01, len=20)

## Run pulsar with huge
hugeargs <- list(lambda=lams, verbose=FALSE)
out.p <- pulsar(dat$data, fun=huge::huge, fargs=hugeargs,
                rep.num=20, criterion='stars')

fit  <- refit(out.p)

## End(Not run)
```

`update.pulsar` *Update a pulsar call*

Description

Update a pulsar model with new or altered arguments. It does this by extracting the call stored in the object, updating the call and (by default) evaluating it in the environment of the original `pulsar` call.

Usage

```
## S3 method for class 'pulsar'
update(object, ..., evaluate = TRUE)
```

Arguments

object	a n existing pulsar or batch.pulsar object
...	arguments to pulsar to update
evaluate	Flag to evaluate the function. If FALSE, the updated call is returned without evaluation

Details

The update call is evaluated in the environment specified by the pulsar or batch.pulsar object, so if any variables were used for arguments to the original call, unless they are purposefully updated, should not be altered. For example, if the variable for the original data is reassigned, the output of update will not be on the original dataset.

Value

If evaluate = TRUE, the fitted object - the same output as pulsar or batch.pulsar. Otherwise, the updated call.

See Also

eval, update, pulsar, batch.pulsar

Examples

```
## Not run: p <- 40 ; n <- 1200
dat   <- huge.generator(n, p, "hub", verbose=FALSE, v=.1, u=.3)
lams  <- getLamPath(getMaxCov(dat$data), .01, len=20)

## Run pulsar with huge
hugeargs <- list(lambda=lams, verbose=FALSE)
out.p <- pulsar(dat$data, fun=huge::huge, fargs=hugeargs,
                rep.num=20, criterion='stars')

## update call, adding bounds
out.b <- update(out.p, lb.stars=TRUE, ub.stars=TRUE)

## End(Not run)
```

Index