Package ‘dils’

July 2, 2014

Type Package

Title Data-Informed Link Strength. Combine multiple-relationship networks into a single weighted network. Impute (fill-in) missing network links.

Version 0.8.1

Date 2013-11-25

Author Stephen R. Haptonstahl

Maintainer Stephen R. Haptonstahl <srh@haptonstahl.org>

Depends R (>= 3.0.1), igraph, Rcpp (>= 0.10.4)

LinkingTo Rcpp

Suggests testthat, Matrix

Description Combine multiple-relationship networks into a single weighted network. The approach is similar to factor analysis in the that contribution from each constituent network varies so as to maximize the information gleaned from the multiple-relationship networks. This implementation uses Principal Component Analysis calculated using 'prcomp' with bootstrap subsampling. Missing links are imputed using the method of Chen et al. (2012).

License MIT + file LICENSE

Collate 'GetSampleFromDataFrame.R' 'RelationStrengthSimilarity.R' 'ScalablePCA.R' 'GetSampleFromFile.R' 'GetSampleFromDb.R' 'RssCell.R' 'RssThisRadius.R' 'EdgelistFill.R' 'EdgelistFromAdjacency.R' 'EdgelistFromIgraph.R' 'GenerateDilsNetwork.R' 'MergeEdgelists.R' 'AdjacencyFromEdgelist.R' 'IgraphFromEdgelist.R' 'MeasureNetworkInformation.R' 'RssSuggestedNetwork.R' 'RelativeNetworkInformation.R'

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-11-25 21:23:02

R topics documented:

`dils-package` *Data-Informed Link Strength. Combine multiple-relationship networks into a single weighted network.*

Description

Combine multiple-relationship networks into a single weighted network. The approach is similar to factor analysys in the that contribution from each constituent network varies so as to maximize the information gleaned from the multimetwork. This implementation uses Principal Component Analysis calculated using 'prcomp' with bootstrap subsampling.

Details

Package:	dils
Type:	Package
Version:	0.8
Date:	2013-10-27
License:	MIT + file LICENSE

Start with a table (data.frame, tab-delimited file, database) where each row/record represents a link between two nodes (a dyad) in a directed or undirected network and each column represents a different relationship between the two nodes, ie. each column is a network. DILS combines these columns/networks into a single network that is a weighted sum of the constituent networks. The resulting DILS network uses information from all of the constituent networks and contains more information than any of the constituent networks. The output is a data.frame of DILS scores for each dyad, therefore is a single network ready for analysis using **igraph** or other social network analysis (SNA) tools.

Workflow synthesizing networks might typically look like this:

1. Start with several networks in igraph, adjacency list, or edgelist form.
2. Is necessary, use `EdgelistFromIgraph` or `EdgelistFromAdjacency` to convert igraph and adjacency list networks to edgelist form.
3. Use `MergeEdgelists` to combine the individual network datasets into a single dataset.
4. Use `GenerateDilsNetwork` to synthesize the networks in the merged data set into a single weighted network.
5. Use `IgraphFromEdgelist` or `AdjacencyFromEdgelist` to convert the edgelist output to the desired output.
6. Use `RelativeNetworkInformation` on input networks and DILS network to see if/how much the information content of the DILS network exceeds the information content of the input networks.

Workflow for imputing edges for a binary network might typically look like this:

1. Start with a binary network as an adjacency matrix (for an igraph use `get.adjacency`).
2. Use `RelationStrengthSimilarity` to Calculate RSS scores for each dyad.
3. Use `RssSuggestedNetwork` on the original network and the RelationStrengthSimilarity output to get a new suggested network with more edges.

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

"Discovering Missing Links in Networks Using Similarity Measures", Hung-Hsuan Chen, Liang Gou, Xiaolong (Luke) Zhang, C. Lee Giles. 2012.

`AdjacencyFromEdgelist` *Convert an edgelist to an adjacency matrix*

Description

Given the adjacency matrix for a network return a data.frame listing all possible edges and the weights for each edge.

Usage

```
AdjacencyFromEdgelist(elist, check.full = TRUE)
```

Arguments

`elist`	data.frame, see 'Details' for formatting assumptions.
`check.full`	logical, if TRUE ensures that all possible edges are in the list exactly once; if FALSE it assumes this is true.

Details

This assumes that `elist` is a data.frame with three columns. Each row is an edge in the network. The first column lists the node the edge is coming from, the second column lists the node the edge is going to, and the third column lists the weight of the edge.

Value

list, containing an adjacency matrix and a vector of node ids identifying the rows and columns.

`adjacency`	The adjacency matrix for the network. The row indicates the node the edge is coming 'from', the column indicat
`nodelist`	The ids of the nodes in the same order as the the rows and columns of the adjacency matrix.

Author(s)

Stephen R. Haptonstahl `<srh@haptonstahl.org>`

References

https://github.com/shaptonstahl/

See Also

`EdgelistFill`

Examples

```
edgelist <- cbind(expand.grid(letters[1:2], letters[1:2]), runif(4))
AdjacencyFromEdgelist(edgelist)
```

`EdgelistFill` *Ensure an edgelist has all dyads and a column of weights.*

Description

Given a matrix or data.frame edgelist, fill in all possible edges not already listed with a weight of 0 or the value of `fillBlanksWith`.

Usage

```
EdgelistFill(elist, fillBlanksWith = 0, nodelist)
```

Arguments

`elist`	data.frame or matrix, see 'Details' for formatting assumptions.
`fillBlanksWith`	numeric, default weight for edges not already listed in elist.
`nodelist`	character, optional list of node names.

Details

The `elist` can be either a data.frame or a matrix with either 2 or 3 columns. Each row is an edge. The first column lists the node the edge is 'from' and the second column lists the node the edge is 'to'. If there is a third column, it lists the weight of the edge.

Value

data.frame, full list of all possible edges with weights for each in third column.

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

https://github.com/shaptonstahl/

Examples

```
g <- erdos.renyi.game(10, 2/10)
EdgelistFill(get.edgelist(g))
EdgelistFill(get.edgelist(g), nodelist=1:10)

E(g)$weight <- runif(ecount(g))
el <- cbind(get.edgelist(g), E(g)$weight)
EdgelistFill(el)
EdgelistFill(el, nodelist=1:10)
```

EdgelistFromAdjacency *Convert an adjacency matrix to filled edgelist.*

Description

Given the adjacency matrix for a network return a data.frame listing all possible edges and the weights for each edge.

Usage

```
EdgelistFromAdjacency(A,
  nodelist = paste("node", 1:nrow(A), sep = ""))
```

Arguments

`A`	matrix, see 'Details' for formatting assumptions.
`nodelist`	character, optional list of node names.

Details

This assumes that the row of the adjacency matrix indicates the node the edge is coming 'from', the column represent the node the edge is going 'to', and the value in the adjacency matrix is the weight given to the edge.

Value

data.frame, full list of all possible edges with weights for each in third column.

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

https://github.com/shaptonstahl/

See Also

EdgelistFromIgraph

Examples

```
n <- 10
A <- matrix(rnorm(n*n), nrow=n)
A
EdgelistFromAdjacency(A)

n <- 100
A <- matrix(rnorm(n*n), nrow=n)
A
EdgelistFromAdjacency(A)

n <- 500
A <- matrix(rnorm(n*n), nrow=n)
A
## Not run: EdgelistFromAdjacency(A)
```

`EdgelistFromIgraph` *Convert an igraph to filled edgelist*

Description

Given an igraph object for a network return a data.frame listing all possible edges and the weights for each edge.

Usage

```
EdgelistFromIgraph(g, useWeight = FALSE)
```

Arguments

`g`	igraph, from igraph package.
`useWeight`	logical, Should E(g)$weight be used as the weights for the edges?

Details

This function is preferred to the igraph function `get.edgelist` because `get.edgelist` only returns rows for edges that have non-zero weight and does not return weights, if present.

Value

data.frame, full list of all possible edges with weights for each in third column.

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

https://github.com/shaptonstahl/

See Also

`EdgelistFromAdjacency`

Examples

```
g <- erdos.renyi.game(10, 2/10)
EdgelistFromIgraph(g)

V(g)$name <- letters[1:vcount(g)]
EdgelistFromIgraph(g)

E(g)$weight <- runif(ecount(g))
EdgelistFromIgraph(g, useWeight=TRUE)
```

GenerateDilsNetwork *Combine multiple networks into a single weighted network.*

Description

Use ScalablePCA to recover optimal weights for each network, then calculate the weighted average across networks for each edge.

Usage

```
GenerateDilsNetwork(x, subsample = 10000,
  n.subsamples = 1000, ignore.cols, use.cols,
  progress.bar = FALSE)
```

Arguments

`x`	data.frame, data over which to run PCA
`subsample`	numeric or logical, If an integer, size of each subsample. If FALSE, runs PCA on entire data set.
`n.subsamples`	numeric, number of subsamples.
`ignore.cols`	numeric, indices of columns not to include
`use.cols`	numeric, indices of columns to use
`progress.bar`	logical, if TRUE then progress in running subsamples will be shown.

Value

A list

`dils`	vector, named vector of component weights for first dimension of principal component analysis (see example for comparison to `prcomp`).
`dils.edgelist`	Unused columns of `x` bound with the DILS scores on the right. Forms an edgelist if there were two unused columns and they containted the ids for the dyads.
`coefficients`	named vector, weights that genereate `dils` by taking dot-product with network data.
`weights`	named vector, raw.weights scaled by standard deviations of network edges, then scaled to sum to 1.

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

https://github.com/shaptonstahl/

See Also

prcomp

Examples

```
data(iris)        # provides example data
GenerateDilsNetwork(iris, subsample=10, use.cols=1:4)
GenerateDilsNetwork(iris, subsample=10, ignore.cols=5)
```

GetSampleFromDataFrame

Randomly select rows from a data.frame.

Description

Randomly select n rows from data.frame x.

Usage

```
GetSampleFromDataFrame(n, x)
```

Arguments

n	numeric, size of sample.
x	data.frame, data whose rows will be sampled.

Value

data.frame, size n random subset of the rows of x

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

https://github.com/shaptonstahl/

See Also

ScalablePCA, GetSampleFromFile, GetSampleFromFile

Examples

```
data(iris)   # provides example data
x <- dils:::GetSampleFromDataFrame(10, iris)
```

GetSampleFromDb — *Sample from the rows of a (possibly large) database table (NOT IMPLEMENTED)*

Description

Access a database table directly. Return a data.frame whose rows are the sample.

Usage

```
GetSampleFromDb(n, db)
```

Arguments

n	numeric, size of sample to be taken.
db	connection, connection to the database table containing the data.

Value

data.frame, size n random subset of the rows of filename

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

https://github.com/shaptonstahl/

See Also

ScalablePCA, GetSampleFromDataFrame, GetSampleFromFile

Examples

```
## Not run: x <- dils:::GetSampleFromDb(10, my.db)
```

GetSampleFromFile *Sample from the rows of a (possibly large) text file (NOT IMPLEMENTED)*

Description

Read a large text file in batches, keeping the rows to be included in the sample. Return a data.frame whose rows are the sample.

Usage

```
GetSampleFromFile(n, out.of, filename)
```

Arguments

n	numeric, size of sample to be taken.
out.of	numeric, number of rows in the data set not including the header.
filename	character, name of the file containing the data. This must be a tab-delimited file with a header row formatted per the default options for read.delim.

Value

data.frame, size n random subset of the rows of filename

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

https://github.com/shaptonstahl/

See Also

ScalablePCA, GetSampleFromDataFrame, GetSampleFromDb

Examples

```
## Not run: x <- dils:::GetSampleFromFile(10, 150, "folder/containing/data.txt")
```

IgraphFromEdgelist — *Convert an edgelist to an igraph*

Description

Given the adjacency matrix for a network return a data.frame listing all possible edges and the weights for each edge.

Usage

```
IgraphFromEdgelist(elist, directed = TRUE)
```

Arguments

`elist`	data.frame, see 'Details' for formatting assumptions.
`directed`	logical, If TRUE, the returned igraph is directed.

Details

This assumes that `elist` is a data.frame with three columns. Each row is an edge in the network. The first column lists the node the edge is coming from, the second column lists the node the edge is going to, and the third column lists the weight of the edge.

Value

igraph, If the edgelist third column has values other than 0, 1 then the weights are stored in E(returned graph)$weight.

Author(s)

Stephen R. Haptonstahl `<srh@haptonstahl.org>`

References

https://github.com/shaptonstahl/

See Also

EdgelistFill

Examples

```
edgelist <- cbind(expand.grid(letters[1:2], letters[1:2]), runif(4))
g <- IgraphFromEdgelist(edgelist)
get.edgelist(g)
E(g)$weight
plot(g, edge.width=5*E(g)$weight, edge.curved=TRUE)
```

`MeasureNetworkInformation` *Measure how much a network informs a particular network measure*

Description

Given an `igraph` network, repeatedly perturb the graph and take some measure of the network to see how much the measure varies, then return a measure that increases as the precision of the function values increases.

Usage

```
MeasureNetworkInformation(g, FUN = betweenness,
  remove.share = 0.2, sample.size = 100,
  progress.bar = FALSE)
```

Arguments

`g`	igraph, graph to measure
`FUN`	function, a function that takes an igraph and returns a value for each node in the network.
`remove.share`	numeric, fraction of the edges that are removed randomly when perturbing the network.
`sample.size`	numeric, number of perturbed graphs to generate
`progress.bar`	logical, if TRUE then a progress bar is shown.

Details

This function can vary tremendously based on the network measure being considered and the other parameters. It is only recommended that this be used for comparing the informativeness of two networks on the same set of nodes, keeping all the parameters the same.

Here information is measured as 1 / mean across and perturbed graphs nodes of the relative error of a network node measure.

Specifically, `FUN` is applied to the graph `g` to generate reference values. Some `sample.size` copies of the igraph are generated. For each, `round(remove.share * n.edges)` randomly selected edges are dropped to generate a perturbed graph. For each perturbed graph `FUN` is applied, generating a value for each node in the network. For each node the relative error

$$|\frac{measure of perturbed g - measure of g}{measure of g}|$$

is calculated, then the mean of these across nodes and perturbed graphs is calculated, generating a mean relative error for the network. This value is reciprocated to get a measure of precision.

This measure appears to be very sensitive to the choice of `FUN`.

Value

numeric, mean precision of the measure FUN across the network

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

https://github.com/shaptonstahl/dils

Examples

```
g.rand <- random.graph.game(100, 5/100)
m.rand <- MeasureNetworkInformation(g.rand)
m.rand

pf <- matrix( c(.8, .2, .3, .7), nr=2)
g.pref <- preference.game(100, 2, pref.matrix=pf)
m.pref <- MeasureNetworkInformation(g.pref)
m.pref

m.pref / m.rand  # Relative informativeness of this preference graph
                 # to this random graph with respect to betweenness
## Not run:
prob.of.link <- c(1:50)/100
mnis <- sapply(prob.of.link, function(p)
  MeasureNetworkInformation(random.graph.game(100, p)))
plot(prob.of.link, mnis,
     type="l",
     main="Network Information of random graphs",
     xlab="probability of link formation",
     ylab="information")
mtext("with respect to betweenness measure", line=0.5)
## End(Not run)
```

MergeEdgelists   *Combine edgelists into a single data.frame*

Description

Given two or more edgelists, create a single edgelist with multiple columns, two for the from and to nodes and one for the weights from each constituent network.

Usage

```
MergeEdgelists(...)
```

Arguments

`...`	data.frames, edgelists to be merged.

Value

data.frame, single multinetwork edgelist

Author(s)

Stephen R. Haptonstahl `<srh@haptonstahl.org>`

References

http://www.haptonstahl.org/R

See Also

`EdgelistFill`

Examples

```
edgelist1 <- data.frame(expand.grid(letters[1:2], letters[1:2]),
                        uniform=runif(4))
edgelist2 <- data.frame(v1=c("a", "a"), v2=c("a", "b"), manual=c(.3, .5))
MergeEdgelists(edgelist1, edgelist2)
```

RelationStrengthSimilarity

Calculate the RSS from one node to another.

Description

For a single pair of nodes, implement the RSS algorithm of Chen et al. (2012).

Usage

```
RelationStrengthSimilarity(xadj, v1, v2, radius = 3,
  directed = TRUE,
  method = c("Rcpp", "BetterR", "NaiveR"))
```

Arguments

`xadj`	numeric matrix, then description of `arg1`.
`v1`	numeric Object type, then description of `arg2`.
`v2`	numeric Object type, then description of `arg2`.
`radius`	numeric, length of longest path examined from `v1` to `v2`.
`directed`	logical, if TRUE returns a symmetric RSS matrix.
`method`	character, choose the method of calculation.

Details

If v1 and v2 are specified, this returns the RSS from v1 to v2. If not, it calculates the RSS scores for all dyads in the network.

Value

numeric, Relation Strength Similarity score(s).

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

"Discovering Missing Links in Networks Using Similarity Measures", Hung-Hsuan Chen, Liang Gou, Xiaolong (Luke) Zhang, C. Lee Giles. 2012.

https://github.com/shaptonstahl/

See Also

ScalablePCA

Examples

```
g1 <- graph.atlas(128)
## Not run: plot(g1)

M1 <- as.matrix(get.adjacency(g1))
M1
RelationStrengthSimilarity(xadj=M1, v1=5, v2=6, radius=1)
RelationStrengthSimilarity(xadj=M1, v1=5, v2=6, radius=2)
RelationStrengthSimilarity(xadj=M1, v1=5, v2=6, radius=3)
RelationStrengthSimilarity(xadj=M1, v1=5, v2=6, radius=4)

RelationStrengthSimilarity(xadj=M1, radius=2)

TestUndirectedNetwork <- function(n) {
  M <- matrix(runif(n*n), nrow=n)
  M <- (M + t(M)) / 2
  diag(M) <- 0
  return(M)
}
M2 <- TestUndirectedNetwork(75)
system.time(RelationStrengthSimilarity(xadj=M2, directed=FALSE, method="BetterR"))  # all R
system.time(RelationStrengthSimilarity(xadj=M2, directed=FALSE))                 # Rcpp
```

`RelativeNetworkInformation`
Compare how much two networks inform a particular network measure

---

Description

Given two igraph networks, use MeasureNetworkInformation to gauge the informativeness of each network and then return the ratio. If greater than 1, then the first network specified is more informative.

Usage

```
RelativeNetworkInformation(g1, g2, FUN = betweenness,
  remove.share = 0.2, sample.size = 100,
  progress.bar = FALSE)
```

Arguments

g1	igraph, graph to measure
g2	igraph, graph to measure
FUN	function, a function that takes an igraph and returns a value for each node in the network.
remove.share	numeric, fraction of the edges that are removed randomly when perturbing the network.
sample.size	numeric, number of perturbed graphs to generate
progress.bar	logical, if TRUE then a progress bar is shown.

Details

This measure appears to be very sensitive to the choice of FUN. See MeasureNetworkInformation for details.

Value

list, containing the following

g1.over.g2	numeric	informativeness of the first network over the second
winner	character	either g1 or g2
g1.measure	numeric	MeasureNetworkInformation(g1, ...)
g2.measure	numeric	MeasureNetworkInformation(g2, ...)

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

https://github.com/shaptonstahl/dils

Examples

```
g.rand <- random.graph.game(100, 5/100)

pf <- matrix( c(.8, .2, .3, .7), nr=2)
g.pref <- preference.game(100, 2, pref.matrix=pf)

RelativeNetworkInformation(g.rand, g.pref)
```

RssCell *Calculate the RSS from one node to another.*

Description

This is a helper function for RelationStrengthSimilarity that returns the RSS for a single directed dyad.

Usage

```
RssCell(xadj, v1, v2, radius)
```

Arguments

xadj	numeric matrix, adjacency matrix where the [i,j] entry gives the strength of the link from node i to node j.
v1	numeric, index of the 'from' node.
v2	numeric, index of the 'to' node.
radius	numeric, length of longest path examined from v1 to v2.

Details

This is an internal function. There are no guardians and it assumes that the adjacency matrix xadj has had zeros entered on the diagonal and then each row divided by the row mean.

Value

numeric, the Relation Strength Similarity score from v1 to v2.

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

"Discovering Missing Links in Networks Using Similarity Measures", Hung-Hsuan Chen, Liang Gou, Xiaolong (Luke) Zhang, C. Lee Giles. 2012.

https://github.com/shaptonstahl/

See Also

RelationStrengthSimilarity

Examples

```
M <- as.matrix(get.adjacency(graph.atlas(128)))
M
M <- sweep(M, 1, rowMeans(M), "/")
M
dils:::RssCell(xadj=M, v1=5, v2=6, radius=1)
dils:::RssCell(xadj=M, v1=5, v2=6, radius=2)
dils:::RssCell(xadj=M, v1=5, v2=6, radius=3)
dils:::RssCell(xadj=M, v1=5, v2=6, radius=4)
```

RssSuggestedNetwork *Suggest a network with imputed links*

Description

A longer description of the function. This can be perhaps a paragraph, perhaps more than one.

Usage

```
RssSuggestedNetwork(g, rss, q.impute.above = 0.8)
```

Arguments

g	Object type, then description of arg1.
rss	Object type, then description of arg2.
q.impute.above	Object type, then description of arg3.

Value

list

g.imputed	igraph contatining the original and the new links
g.new	igraph containing only the new links
g.original	original graph
q.impute.above	quantile of RSS scores above which links should be imputed
frac.filled	fraction of potential links that were actually filled with a new link

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

http://www.haptonstahl.org/R

See Also

RelationStrengthSimilarity

Examples

```
g <- graph.atlas(128)
## Not run: plot(g)

suggested <- RssSuggestedNetwork(g, q.impute.above=.6)
## Not run: plot(suggested$g.imputed)
suggested$frac.filled
```

RssThisRadius *Calculate part of the RSS from one node to another.*

Description

This is a helper function for RelationStrengthSimilarity that returns the component of RSS contributed by paths of one particular length r.

Usage

```
RssThisRadius(x, v1, v2, r, prepped = FALSE)
```

Arguments

x	numeric matrix, adjacency matrix where the [i,j] entry gives the strength of the link from node i to node j.
v1	numeric, index of the 'from' node.
v2	numeric, index of the 'to' node.
r	numeric, length of paths examined from v1 to v2.
prepped	logical, whether or not the adjacency matrix x has had zeros entered on the diagonal and each row divided by the row sum.

Value

numeric, the part of the Relation Strength Similarity score from v1 to v2 contributed by paths of length r.

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

"Discovering Missing Links in Networks Using Similarity Measures", Hung-Hsuan Chen, Liang Gou, Xiaolong (Luke) Zhang, C. Lee Giles. 2012.

https://github.com/shaptonstahl/

See Also

RelationStrengthSimilarity

Examples

```
M <- as.matrix(get.adjacency(graph.atlas(128)))
M
dils:::RssThisRadius(x=M, v1=5, v2=6, r=1)
dils:::RssThisRadius(x=M, v1=5, v2=6, r=2)
dils:::RssThisRadius(x=M, v1=5, v2=6, r=3)
dils:::RssThisRadius(x=M, v1=5, v2=6, r=4)
```

ScalablePCA *Perform Principal Component Analysis on a large data set*

Description

Run prcomp on subsamples of the data set and compile the results for the first dimension.

Usage

```
ScalablePCA(x, filename = NULL, db = NULL,
  subsample = 10000, n.subsamples = 1000, ignore.cols,
  use.cols, return.sds = FALSE, progress.bar = FALSE)
```

Arguments

x	data.frame, data over which to run PCA
filename	character, name of the file containing the data. This must be a tab-delimited file with a header row formatted per the default options for read.delim.
db	Object type, database connection to table containing the data (NOT IMPLEMENTED).
subsample	numeric or logical, If an integer, size of each subsample. If FALSE, runs PCA on entire data set.
n.subsamples	numeric, number of subsamples.
ignore.cols	numeric, indices of columns not to include.

`use.cols`	numeric, indices of columns to use.
`return.sds`	logical, if TRUE return the standard deviations of each network's edge weights.
`progress.bar`	logical, if TRUE then progress in running subsamples will be shown.

Details

Scales the function `prcomp` to data sets with an arbitrarily large number of rows by running `prcomp` on repeated subsamples of the rows.

Value

If `return.sds` is FALSE, return named vector of component weights for first dimension of principal component analysis (see example for comparison to `prcomp`).

If `return.sds` is TRUE, return a list.

coefficients	named vector of the component weights for first dimension of principal component analysis (see example for co
sds	named vector of the standard deviations of each network's edge weights.

Author(s)

Stephen R. Haptonstahl <srh@haptonstahl.org>

References

https://github.com/shaptonstahl/

See Also

`prcomp`

Examples

```
data(iris)        # provides example data
prcomp(iris[,1:4], center=FALSE, scale.=FALSE)$rotation[,1]
ScalablePCA(iris, subsample=10, use.cols=1:4)
ScalablePCA(iris, subsample=10, ignore.cols=5)
```

Index