

# Package ‘ProCoNA’

October 18, 2017

**Type** Package

**Title** Protein co-expression network analysis (ProCoNA).

**Version** 1.14.0

**Date** 2013-04-28

**Author** David L Gibbs

**Maintainer** David L Gibbs <gibbsd@ohsu.edu>

**Description** Protein co-expression network construction using peptide level data, with statistical analysis. (Journal of Clinical Bioinformatics 2013, 3:11 doi:10.1186/2043-9113-3-11)

**License** GPL (>= 2)

**LazyLoad** yes

**Depends** R (>= 2.10), methods, WGCNA, MSnbase, flashClust

**Imports** BiocGenerics, GOstats

**Suggests** RUnit

**biocViews** GraphAndNetwork, Software, Proteomics

**NeedsCompilation** no

## R topics documented:

procona-package . . . . .	2
Accessors . . . . .	3
bootstrapProconaNetwork . . . . .	4
buildProconaNetwork . . . . .	5
c2i . . . . .	6
compareNetworksWithFishersExactTest . . . . .	7
compareNetworksWithFishersExactTestProcona . . . . .	8
corBootstrap . . . . .	8
correlationWithPhenotypesHeatMap . . . . .	9
getFisherMatrix . . . . .	10
getPeptideNAs . . . . .	11
goStatTest . . . . .	11
hclust-class . . . . .	12
i2c . . . . .	13
i2col . . . . .	13
MMvsPS . . . . .	14

MMvsPSallModules . . . . .	14
moduleMemberCorrelations . . . . .	15
modulePhenotypeCorrelations . . . . .	16
orderMatrixIndex . . . . .	17
peptideConnectivityTest . . . . .	17
peptideCorrelationTest . . . . .	18
plotNet . . . . .	19
ppiPermTest . . . . .	19
printNet . . . . .	20
ProCoNA-Data . . . . .	21
proconaNet-class . . . . .	21
proconaVersionFun . . . . .	22
runningStats . . . . .	23
subsetModCors . . . . .	23
subsetPeptideData . . . . .	24
toPermTest . . . . .	24
utri . . . . .	25

**Index** **26**

---

procona-package      *Peptide co-expression network construction.*

---

**Description**

Peptide co-expression network construction, analysis, and visualization.

**Details**

Package:	procona
Type:	Package
Title:	Peptide co-expression network construction.
Version:	0.13
Date:	2011-08-10
Author:	David L Gibbs
Maintainer:	David Gibbs <gibbsd@ohsu.edu>
License:	GPLv3
LazyLoad:	yes
Depends:	WGCNA, GOstats, multicore

**Author(s)**

David L Gibbs

**Description**

Accessor functions allow access to the object data.

**Methods**

**TOM:** The topological overlap matrix or TOM. "matrix"

**adj:** The adjacency matrix. "matrix"

**networkName:** A name describing the data or experiment used to build the network. "character"

**samples:** The names of samples used in building the network. "character"

**peptides:** The names of peptides used in the network, also the node names. "character"

**pepTree:** The network dendrogram. "hclust"

**dynamicColors:** The module labels on each node (or peptide). "numeric"

**MEs:** The module eigenvectors (or eigen-peptides). "data.frame"

**mergedMEs:** The module eigenvectors after merging similar modules. "data.frame"

**mergedColors:** The module labels after merging similar modules. "numeric"

**colorOrder:** Modules are ordered by size, these labels correspond to that order. "character"

**power:** The soft thresholding power used in scaling the adjacency matrix. "numeric"

**networkType:** Either a signed or unsigned network regarding the method used in computing the initial correlations between nodes. "character"

**permtest:** The results of the permutation test on significance of topological overlap within modules. "matrix"

**proconaVersion:** Returns the version number of the software that built the object. "character"

**Author(s)**

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
tomMatrix <- TOM(net1)
```

---

 bootstrapProconaNetwork

*bootstrapProconaNetwork*


---

### Description

This function returns a peptide co-expression network object based on a bootstrapped correlation matrix.

### Usage

```
bootstrapProconaNetwork(networkName = "bootstrap procona", pepdat = NULL,
  pow = NULL, powMax = 20, networkType = "signed", scaleFreeThreshold = 0.8,
  deepSplit = 2, minModuleSize = 30, mergeThreshold = 0.1,
  clusterType = "average", pamRespectsDendro = T, performTOPermtest = TRUE,
  toPermTestPermutates = 100, bootstrapThreshold = 1e-04)
```

### Arguments

networkName	Name of this network
pepdat	This variable is the data set with rows as samples and cols as peptides
pow	The scaling power, NULL if unknown
powMax	The maximum power to be searched.
networkType	Whether the sign is considered in constructing adjacency and TOM
scaleFreeThreshold	The threshold for fitting to scale-free topology.. will use closest power.
deepSplit	Course grain control of module size
minModuleSize	The minimum module size allowed
mergeThreshold	Below this threshold, modules are merged.
clusterType	Clustering option
pamRespectsDendro	When cutting the dendrogram, pay attention to branch membership.
performTOPermtest	Performs permutation testing on modules
toPermTestPermutates	Number of permutations to do.
bootstrapThreshold	When to stop resampling...

### Value

returns the procona network object

### Author(s)

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
net <- bootstrapProconaNetwork("peptide network", peptideData,
performTOPermtest=FALSE, bootstrapThreshold=0.1)
```

---

buildProconaNetwork    *buildProconaNetwork*

---

**Description**

This function returns a peptide co-expression network object.

**Usage**

```
buildProconaNetwork(networkName = "ProCoNA", pepdat, pow=1,
  powMax = 20, networkType = "signed", pearson = FALSE, scaleFreeThreshold = 0.8,
  deepSplit = 2, minModuleSize = 30, mergeThreshold = 0.1,
  clusterType = "average", pamRespectsDendro = TRUE, performTOPermtest = TRUE,
  toPermTestPermutates = 100)
```

**Arguments**

networkName	Name of this network
pepdat	This variable is the data set with rows as samples and cols as peptides
pow	The scaling power, NULL if unknown
powMax	The maximum power to be searched.
networkType	Should the sign be considered in constructing adjacency and TOM ("signed" or "unsigned")
pearson	use Pearson's cor or the robust bi-weight correlation
scaleFreeThreshold	The threshold for fitting to scale-free topology.. will use closest power.
deepSplit	Course grain control of module size
minModuleSize	The minimum module size allowed
mergeThreshold	Below this threshold, modules are merged.
clusterType	Clustering option
pamRespectsDendro	When cutting the dendrogram, pay attention to branch membership.
performTOPermtest	Performs permutation testing on modules
toPermTestPermutates	Number of permutations to do.

**Details**

The procona network object contains a number of slots which store information relevant to the construction of the network. Accessor functions provide direct access to the slots. See `getSlots("proconaNet")` for a complete list.

**Value**

returns the procona network object

**Author(s)**

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
net <- buildProconaNetwork("peptide network", peptideData)
```

---

c2i

*c2i*

---

**Description**

coordinates to index

**Usage**

```
c2i(nrows, x, y)
```

**Arguments**

nrows	number of rows in the matrix
x	the row coordinate
y	the col coordinate

**Value**

the index into the matrix

**Author(s)**

David L Gibbs

---

```
compareNetworksWithFishersExactTest
      compareNetworksWithFishersExactTest
```

---

## Description

Fisher's exact test is used pairwise on modules to compare two networks. The arguments to Fisher's exact test are given below.

n == number of entities in the network

m == number of entities in intersection of two modules

d1 == number of entities in module A but not in module B

d2 == number of entities in module B but not in module A

2x2 matrix for the test is then: m d1 d2 n-d1-d2-m

## Usage

```
compareNetworksWithFishersExactTest(peps1, peps2, colors1, colors2,
      title = "", net1label = "", net2label = "")
```

## Arguments

peps1	Nodes in network 1, character vector
peps2	Nodes in network 2, character vector
colors1	modules for net 1
colors2	modules for net 2
title	Plot title
net1label	xlabel
net2label	ylabel

## Value

Returns fishers exact test -log pvalues and overlap matrix showing the number of shared members for each pair of modules.

## Author(s)

David L Gibbs

## Examples

```
## Not run:
data(ProCoNA_Data)
#net1 <- buildProconaNetwork("peptide network", peptideData, pow=12)
#net2 <- buildProconaNetwork("peptide network", peptideData, pow=6)
compareNetworksWithFishersExactTest(peptides(net1), peptides(net2),
mergedColors(net1), mergedColors(net2), "network comparison", "net1", "net2")

## End(Not run)
```

---

`compareNetworksWithFishersExactTestProcona`*compareNetworksWithFishersExactTestProcona*

---

**Description**

Convenience function for calling the `compareNetworksWithFishersExactTest` using only two procona objects.

**Usage**

```
compareNetworksWithFishersExactTestProcona(net1, net2,  
      title)
```

**Arguments**

<code>net1</code>	procona object for network 1
<code>net2</code>	procona object for network 2
<code>title</code>	plot title

**Value**

Returns a list of fisher -log pvalues, and overlaps between modules.

**Author(s)**

David L Gibbs

**Examples**

```
## Not run:  
data(ProCoNA_Data)  
#net1 <- buildProconaNetwork("peptide network", peptideData)  
#net2 <- buildProconaNetwork("peptide network", peptideData)  
compareNetworksWithFishersExactTestProcona(net1, net2, "new comparison")  
  
## End(Not run)
```

---

`corBootstrap`*corBootstrap*

---

**Description**

Boostraps a correlation matrix. In order to bootstrap a large correlation matrix, several thousand samplings may be necessary. To avoid storing thousands of matrices, a running mean is kept for each pairwise correlation. In addition, a running standard deviation is computed so that for each pairwise correlation, we can estimate the distribution of values across resamplings. After each resampling, a new correlation matrix is computed. A difference is taken between this new matrix and the running mean. If all differences are less than the specified threshold, then the bootstrapped matrix has converged to a final state.



**Usage**

```
corBootstrap(dataMatrix, networkType = "signed", threshold = 1e-04,
             tmpSaveFile = TRUE)
```

**Arguments**

dataMatrix	Matrix with samples in rows and peptides (or other data type) in columns.
networkType	Whether the sign is considered in constructing adjacency and TOM
threshold	Maximum difference allowed between running mean bootstrap correlation matrix, and new resampled cor matrix. Defines how soon we consider the bootstrap to have converged.
tmpSaveFile	Should temporary saves be done?

**Value**

Returns a list of the bootstrapped matrix, standard deviation matrix, and the number of resamplings done.

**Author(s)**

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
x <- peptideData[,1:10]
y <- corBootstrap(dataMatrix=x, networkType="unsigned", threshold=0.1, tmpSaveFile=FALSE)
```

---

correlationWithPhenotypesHeatMap

*correlationWithPhenotypesHeatMap*

---

**Description**

Plots a heatmap showing the Pearson correlation of modules with phenotypes.

**Usage**

```
correlationWithPhenotypesHeatMap(net, phenotypes, modules,
                                 plotName, title, textSize)
```

**Arguments**

net	The ProCoNA network object.
phenotypes	Matrix of phenotypic traits, can include character strings (converted to factors).
modules	Vector of modules to plot. Default is all modules.
plotName	Name of the saved plot, NULL to show on screen.
title	Plot title.
textSize	The font size of the correlations shown in each module-phenotype pair.

**Value**

the module eigenvector correlations

**Author(s)**

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
#net1 <- buildProconaNetwork("pepnet", peptideData, pow=12)
n <- length(samples(net1))
phenotypes <- matrix(rnorm(10*n), nrow=60)
moduleCors <- correlationWithPhenotypesHeatMap(net1, phenotypes, modules = 1:7,
  plotName = "Phenotype Associations", title = "Module-trait relationships", textSize = 0.5)
```

---

`getFisherMatrix`

*getFisherMatrix*

---

**Description**

Fisher's exact test pairwise on modules.

**Usage**

```
getFisherMatrix(peps1, peps2, colors1, colors2)
```

**Arguments**

<code>peps1</code>	Names of entities in the network (nodes of network 1)
<code>peps2</code>	Names of entities in the network (nodes of network 2)
<code>colors1</code>	the module assignments for network 1
<code>colors2</code>	the module assignments for network 2

**Value**

Returns the fisher test pvalues and count of overlapping peptides.

**Author(s)**

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
#net1 <- buildProconaNetwork("peptide network", peptideData, pow=12)
#net2 <- buildProconaNetwork("peptide network", peptideData + 0.3*rnorm(length(peptideData)), pow=12)
getFisherMatrix(peptides(net1), peptides(net2), mergedColors(net1), mergedColors(net2))
```

---

getPeptideNAs	<i>getPeptideNAs</i>
---------------	----------------------

---

**Description**

This function returns the number of NAs for each peptide.

**Usage**

```
getPeptideNAs(pepdat)
```

**Arguments**

pepdat            the peptide data.

**Value**

returns a list of counts of NAs for each peptide.

**Author(s)**

David L Gibbs

---

goStatTest	<i>goStatTest</i>
------------	-------------------

---

**Description**

Wrapper function to run the hyperGTest from package GOstats, after mapping each peptide to an entrez ID.

**Usage**

```
goStatTest(pnet, module, pepinfo, pepColName, protColName, universe,
           onto, annot, pvalue, cond)
```

**Arguments**

pnet	Procona network object.
module	Module of interest (numeric)
pepinfo	The mass tag info, mapping peptides to proteins.
pepColName	Column name in mass tag info for peptides
protColName	Column name in mass tag info for proteins
universe	Table mapping protein IDs to entrez IDs
onto	The ontology category (bp etc)..
annot	The annotation database to use
pvalue	pvalue cutoff
cond	conditional parameter, see GOstats.

**Value**

Returns the results of the hyper geometric test.

**Author(s)**

David L Gibbs

**Examples**

```
## Not run:
data(ProCoNA_Data)
#net1 <- buildProconaNetwork("peptide network", peptideData, pow=12)
goStatTest(net1, 1, masstagdb, "Mass_Tag_ID", "Reference", universe, "BP", "org.Mm.eg.db", 0.005, FALSE)

## End(Not run)
```

---

hclust-class

*Class "hclust"*

---

**Description**

From the OneHandClapping package. Thanks! Dummy class to permit object of S3 class hclust in S4 class definition of Screening

**Objects from the Class**

Objects can be created by calls of the form `new("hclust", ...)`.

**Slots**

.Data: Object of class "list" ~~

**Extends**

Class "list", from data part. Class "vector", by class "list", distance 2.

**Methods**

No methods defined with class "hclust" in the signature.

**Warning**

This class is just defined as a dummy class. No objects should be instantiated.

**Note**

This class is just defined as a dummy class. No objects should be instantiated.

**Examples**

```
showClass("hclust")
```

---

*i2c**i2c*

---

**Description**

Index to coordinates

**Usage**

`i2c(nrows, i)`

**Arguments**

`nrows`            number of rows in the matrix  
`i`                    the index into the matrix

**Value**

the row col coordinates into the matrix

**Author(s)**

David L Gibbs

---

*i2col**i2col*

---

**Description**

index to column

**Usage**

`i2col(nrows, i)`

**Arguments**

`nrows`            number of rows in matrix  
`i`                    the index

**Value**

returns the column of the matrix

**Author(s)**

David L Gibbs

---

MMvsPS                      *Module members vs Peptide Significance*

---

### Description

Plots the module membership (correlation to eigenvector) against the peptide significance (correlation to phenotype) for a given trait and module

### Usage

```
MMvsPS(pnet, pepdat, phenoVec, mod)
```

### Arguments

pnet	The procona network
pepdat	the peptide data, with rows as samples and columns as peptides
phenoVec	the phenotypic trait, vector
mod	the module of interest

### Value

returns a list of module memberships and peptide significances.

### Author(s)

David L Gibbs

### Examples

```
data(ProCoNA_Data)
#net1 <- buildProconaNetwork("peptide network", peptideData, pow=13)
MMvsPS(net1, peptideData, phenotypes[,5], 1)
```

---

MMvsPSallModules            *MMvsPSallModules*

---

### Description

Call MMvsPS, producing plots for all modules.

### Usage

```
MMvsPSallModules(net, peptable, phenoVec, prefixName)
```

### Arguments

net	The procona network object
peptable	The peptide data
phenoVec	The phenotypic trait, as a numeric vector
prefixName	The plot files prefix name. Writes pdfs.

**Value**

nothing returned

**Author(s)**

David L Gibbs

**Examples**

```
## Not run:  
# This function outputs a set of pdfs.  
data(ProCoNA_Data)  
#net1 <- buildProconaNetwork("peptide network", peptideData, pow=13)  
MMvsPSallModules(net1, peptideData, phenotypes[,5], 1)  
  
## End(Not run)
```

---

moduleMemberCorrelations

*moduleMemberCorrelations*

---

**Description**

Computes the relation between peptides and eigenvector summaries and also peptides and phenotypes.

**Usage**

```
moduleMemberCorrelations(pnet, pepdat, phenotypes)
```

**Arguments**

pnet	The peptide net object
pepdat	The peptide data matrix
phenotypes	The matrix of traits

**Value**

Matrix of Pearson correlations with peptides in rows.

**Author(s)**

David L Gibbs

**Examples**

```

data(ProCoNA_Data)
#net1 <- buildProconaNetwork("peptide network", peptideData)
n <- length(samples(net1))
phenotypes <- matrix(rnorm(10*n), nrow=60)
pepcor <- moduleMemberCorrelations(net1, peptideData, phenotypes)

# To plot the heatmap:
# moduleCors <- correlationWithPhenotypesHeatMap(net1, phenotypes, modules = 1:5,
#   plot = NULL, title = "Module-trait relationships", textSize = 0.5)

#####
# quick function to write out the tables for specific modules.
#moduleData <- function(pepnet, pepcors, module, pepinfo, fileprefix) {
#   moduleX <- pepnet@peptides[which(pepnet@mergedColors==module)]
#   moduleInfo <- pepinfo[which(pepinfo$Mass_Tag_ID %in% moduleX),]
#   moduleCors <- pepcors[which(pepcors$Module==module),]
#   corname <- paste(fileprefix, "_correlations.csv", sep="")
#   write.table(moduleCors, file=corname, sep=",", row.names=F)
#   infoname <- paste(fileprefix, "_peptide_info.csv", sep="")
#   write.table(moduleInfo, file=infoname, sep=",", row.names=F)
#}
#####

# WRITE OUT A TABLE WITH THE BELOW FUNCTION CALL :)#
# moduleData(peptideNetwork, pepcor, 1, masstagdb, "Module_1")

```

---

modulePhenotypeCorrelations

*modulePhenotypeCorrelations*

---

**Description**

Computes the relation between the modules and the phenotypes.

**Usage**

```
modulePhenotypeCorrelations(pnet, phenotypes)
```

**Arguments**

pnet	The peptide net object
phenotypes	The matrix of traits

**Value**

returns a matrix of correlations between modules and phenotypes.

**Author(s)**

David L Gibbs



**Examples**

```
data(ProCoNA_Data)
#net1 <- buildProconaNetwork("peptide network", peptideData, pow=13)
n <- length(samples(net1))
phenotypes <- matrix(rnorm(10*n), nrow=60)
m <- modulePhenotypeCorrelations(net1, phenotypes)

# To plot the heatmap:
# moduleCors <- correlationWithPhenotypesHeatMap(net1, phenotypes, modules = 1:5,
#   plot = NULL, title = "Module-trait relationships", textSize = 0.5)
```

---

orderMatrixIndex	<i>orderMatrixIndex</i>
------------------	-------------------------

---

**Description**

Order the the matrix by upper diag in a greedy fashion

**Usage**

```
orderMatrixIndex(mat)
```

**Arguments**

mat	A matrix
-----	----------

**Value**

returns a matrix in order of greatest in upper diagonal direction.

**Author(s)**

David L Gibbs

---

peptideConnectivityTest	<i>peptideConnectivityTest</i>
-------------------------	--------------------------------

---

**Description**

This function will compare the connectivity between peptides mapped to a given protein, against a randomly drawn, similarly sized, selection of peptides. The hypothesis is that peptides from a given protein should be more connected than random.

**Usage**

```
peptideConnectivityTest(pnet, pepInfo, pepCol, protCol, repsPerProt)
```

**Arguments**

<code>pnet</code>	The peptide net object
<code>pepInfo</code>	The peptide information table, mapping peptides to proteins
<code>pepCol</code>	The string identifying the column in the <code>pepInfo</code> table with peptide ID
<code>protCol</code>	String identifying column in <code>pepInfo</code> with Protein ID.
<code>repsPerProt</code>	number of repetitions for the null

**Value**

Returns a list of the connected peptides and the random samples.

**Author(s)**

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
#net1 <- buildProconaNetwork("peptide network", peptideData, pow=12)
p <- peptideConnectivityTest(net1, masstagdb, "Mass_Tag_ID", "Reference", 200)
```

---

`peptideCorrelationTest`

*peptideCorrelationTest*

---

**Description**

Take the data, and a mapping of peptides to proteins, and compute the mean correlation between peptides linked to a given protein. Compare a similar number of random correlations.

**Usage**

```
peptideCorrelationTest(dat, pepinfo, pepCol, protCol)
```

**Arguments**

<code>dat</code>	The data with samples as rows and peptides as columns
<code>pepinfo</code>	The mapping of peptides to proteins as a data frame
<code>pepCol</code>	The column name of peptide info table containing peptide IDs
<code>protCol</code>	The column name of pepinfo info table containing protein IDs

**Value**

return a t-test comparing protein correlations to random correlations.

**Author(s)**

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
net1 <- buildProconaNetwork("peptide network", peptideData, pow=12)
peptideCorrelationTest(peptideData, masstagdb, "Mass_Tag_ID", "Reference")
```

---

plotNet

*plotNet*

---

**Description**

Plots the dendrogram and module colors. See `?plotDendroAndColors`

**Usage**

```
plotNet(object)
```

**Arguments**

object            The procona network object.

**Value**

None.

**Author(s)**

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
#net1 <- buildProconaNetwork("peptide network", peptideData)
plotNet(net1)
```

---

ppiPermTest

*ppiPermTest*

---

**Description**

Performs a permutation test for enrichment of PPI edges given a database. Peptides are selected from each module and mapped to potential protein parents in the mass tag database. We check if these proteins are found in the PPI network, and record any edges between them. This is compared to edges found using randomly selected proteins (taken from the mass tag database). A p-value is computed as the number of times the randomly sampled proteins incurred more edges than the observed proteins, divided by the number of iterations.

**Usage**

```
ppiPermTest(pnet, pepdat, pepinfo, pepColName, pi_colName, pi_edges,
            threshold, iterations)
```

**Arguments**

pnet	procona network object
pepdat	the data matrix with peptides as columns.
pepinfo	Maps peptides to proteins ... same format as in ppiTable
pepColName	The column in pepinfo with peptide IDs... as in pepdat (the peptide data matrix)
pi_colName	The column in pepinfo that maps peptides to unit found in pi_edges
pi_edges	Must be two columns A-B ... sort out evidence levels (in vivo or in vitro) in advance
threshold	Minimum peptide correlation with module eigenvector.
iterations	Number of repetitions

**Value**

returns list of test results.

**Author(s)**

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
#net1 <- buildProconaNetwork("peptide network", peptideData, pow=12)
ppis <- data.frame(A=sample(masstagdb$Reference, 50), B=sample(masstagdb$Reference, 50))
ppiPermTest(net1, peptideData, masstagdb, "Mass_Tag_ID", "Reference", ppis, 0.33, 100)
```

---

printNet

*printNet*

---

**Description**

Prints general information about the network object.

**Usage**

```
printNet(object)
```

**Arguments**

object            The procona network object.

**Value**

None.

**Author(s)**

David L Gibbs

**Examples**

```
## Not run:
data(ProCoNA_Data)
net1 <- buildProconaNetwork("peptide network", peptideData)
printNet(net1)

## End(Not run)
```

ProCoNA-Data

*A simulated mass tag data base***Description**

The mass tag database, which would be used to identify peptides, simply maps peptide IDs to peptide sequences and protein matches.

This simulated peptide dataset was generated using OpenMS's MSSimulator. A set of proteins was randomly sampled, and used to generate a likely set of observed peptides. Then data for a co-expression network was simulated with WGCNA's simulation function, and columns were named with simulated peptides.

The mass tag database, which would be used to identify peptides, simply maps peptide IDs to peptide sequences and protein matches. This represents a mapping to Entrez IDs.

The matrix annotates the biological samples according to ... phenotypic observations!

The two network objects are included to avoid rebuilding them in the other man page examples.

proconaNet-class

*proconaNet S4 class***Description**

The main ProCoNA object - holder of data.

**Objects from the Class**

Objects can be created by calls of the form `new(proconaNet ...)`

**Slots**

**networkName:** A name describing the data or experiment used to build the network. "character"

**samples:** The names of samples used in building the network. "character"

**adj:** The adjacency matrix. "matrix"

**TOM:** The topological overlap matrix or TOM. "matrix"

**peptides:** The names of peptides used in the network, also the node names. "character"

**pepTree:** The network dendrogram. "hclust"

**dynamicColors:** The module labels on each node (or peptide). "numeric"

**MEs:** The module eigenvectors (or eigen-peptides). "data.frame"

**mergedMEs:** The module eigenvectors after merging similar modules. "data.frame"

**mergedColors:** The module labels after merging similar modules. "numeric"

**colorOrder:** Modules are ordered by size, these labels correspond to that order. "character"

**power:** The soft thresholding power used in scaling the adjacency matrix. "numeric"

**networkType:** Either a signed or unsigned network regarding the method used in computing the initial correlations between nodes. "character"

**permtest:** The results of the permutation test on significance of topological overlap within modules. "matrix"

**proconaVersion:** Returns the version number of the software that built the object. "character"

### Methods

**show** signature(x = "proconaNet"): Shows info about the network.

**print** signature(x = "proconaNet"): Prints info about the network.

### Author(s)

David L Gibbs

---

proconaVersionFun      *Procona Software Version*

---

### Description

Returns the current version of the software.

### Usage

```
proconaVersionFun()
```

### Value

returns the version

### Author(s)

David L Gibbs

---

runningStats	<i>runningStats</i>
--------------	---------------------

---

**Description**

Computing the running mean and variance

**Usage**

```
runningStats(newMat, runningMean, Mk1, Sk1, k)
```

**Arguments**

newMat	The matrix from resampled data
runningMean	The running mean matrix
Mk1	Matrix used in calculation of mean
Sk1	Matrix used in calculation of sd
k	Current resampling iteration

**Value**

returns the list of runningMean, runningSD, Mk, Sk

**Author(s)**

David L Gibbs

---

subsetModCors	<i>subsetModCors</i>
---------------	----------------------

---

**Description**

subsets the module-phenotype correlation matrix which has funny rownames

**Usage**

```
subsetModCors(modCors, modules)
```

**Arguments**

modCors	The matrix of module-phenotype correlations
modules	Which modules are desired.

**Author(s)**

David L Gibbs

---

subsetPeptideData	<i>subsetPeptideData</i>
-------------------	--------------------------

---

### Description

Given a matrix of peptide data, omit columns with excess missing data, specified by NAs.

### Usage

```
subsetPeptideData(pepdatt, numNAsAllowed = NULL, percentageNAsAllowed = 0.05)
```

### Arguments

pepdatt	The peptide matrix, with peptides in columns and samples in rows.
numNAsAllowed	The maximum count of missing values for each peptide (counts NAs).
percentageNAsAllowed	The percentage of missing data allowed for each peptide over samples.

### Value

Returns a matrix.

### Author(s)

David L Gibbs

### Examples

```
data(ProCoNA_Data)
subsetPeptideData(peptideData, percentageNAsAllowed=0.2)
```

---

toPermTest	<i>toPermTest</i>
------------	-------------------

---

### Description

Uses the procona network object, the data with peptides as columns, samples in rows. And the power that the net was built at the number of permutations to do... Modules are permuted and mean topological overlap is recorded, constructing the null. The number of random permutations with mean TO greater than observed provides the p-value.

### Usage

```
toPermTest(pnet, numPermutates)
```

### Arguments

pnet	ProCoNA network object.
numPermutates	The number of permutations to perform



**Value**

returns the network obj with the perm test

**Author(s)**

David L Gibbs

**Examples**

```
data(ProCoNA_Data)
#net1 <- buildProconaNetwork("peptide network", peptideData, pow=12)
toPermTest(net1, 100)
```

---

utri

*utri*

---

**Description**

The upper triangle of a matrix

**Usage**

```
utri(mat)
```

**Arguments**

mat                    A matrix

**Value**

Returns a vector

**Author(s)**

David L Gibbs

**Examples**

```
m <- matrix(rnorm(9), nrow=3, ncol=3)
utri(m)
```



- net1 (ProCoNA-Data), 21
- net2 (ProCoNA-Data), 21
- networkName (Accessors), 3
- networkName, proconaNet-method (Accessors), 3
- networkName<- (Accessors), 3
- networkName<-, proconaNet-method (Accessors), 3
- networkPower (Accessors), 3
- networkPower, proconaNet-method (Accessors), 3
- networkPower<- (Accessors), 3
- networkPower<-, proconaNet-method (Accessors), 3
- networkType (Accessors), 3
- networkType, proconaNet-method (Accessors), 3
- networkType<- (Accessors), 3
- networkType<-, proconaNet-method (Accessors), 3
- orderMatrixIndex, 17
- pepnet (Accessors), 3
- pepnet-class (Accessors), 3
- peptideConnectivityTest, 17
- peptideConnectivityTest, proconaNet, data.frame, character, numeric-method (peptideConnectivityTest), 17
- peptideCorrelationTest, 18
- peptideCorrelationTest, matrix, data.frame, character, proconaNet-method (peptideCorrelationTest), 18
- peptideData (ProCoNA-Data), 21
- peptides (Accessors), 3
- peptides, proconaNet-method (Accessors), 3
- peptides<- (Accessors), 3
- peptides<-, proconaNet-method (Accessors), 3
- pepTree (Accessors), 3
- pepTree, proconaNet-method (Accessors), 3
- pepTree<- (Accessors), 3
- pepTree<-, proconaNet-method (Accessors), 3
- permtest (Accessors), 3
- permtest, proconaNet-method (Accessors), 3
- permtest<- (Accessors), 3
- permtest<-, proconaNet-method (Accessors), 3
- phenotypes (ProCoNA-Data), 21
- plotNet, 19
- power (Accessors), 3
- power, proconaNet-method (Accessors), 3
- power<- (Accessors), 3
- power<-, proconaNet-method (Accessors), 3
- ppiPermTest, 19
- ppiPermTest, proconaNet, matrix, data.frame, character, character-method (ppiPermTest), 19
- print, proconaNet-method (proconaNet-class), 21
- printNet, 20
- ProCoNA-Data, 21
- procona-package, 2
- proconaNet (proconaNet-class), 21
- proconaNet-class, 21
- proconaVersion (Accessors), 3
- proconaVersion, proconaNet-method (Accessors), 3
- proconaVersion<- (Accessors), 3
- proconaVersion<-, proconaNet-method (Accessors), 3
- proconaVersionFun, 22
- runningStats, 23
- samples (Accessors), 3
- samples, proconaNet-method (Accessors), 3
- samples, proconaNet-methods (Accessors), 3
- samples<- (Accessors), 3
- samples<-, proconaNet-method (Accessors), 3
- samples, proconaNet-methods (Accessors), 3
- show, proconaNet-method (proconaNet-class), 21
- subsetModCors, 23
- subsetPeptideData, 24
- TOM (Accessors), 3
- TOM, proconaNet-method (Accessors), 3
- TOM<- (Accessors), 3
- TOM<-, proconaNet-method (Accessors), 3
- toPermTest, 24
- toPermTest, proconaNet, numeric-method (toPermTest), 24
- universe (ProCoNA-Data), 21
- utri, 25
- vector, 12