Package 'ScISI'

September 24, 2012

Title In Silico Interactome

Version 1.28.0

Author Tony Chiang <tchiang@fhcrc.org>

Description Package to create In Silico Interactomes

Depends R (>= 2.10), GO.db, RpsiXML, annotate, apComplex

Imports AnnotationDbi, GO.db, RpsiXML, annotate, methods,org.Sc.sgd.db, utils

Suggests ppiData, xtable

Maintainer Tony Chiang <tchiang@fhcrc.org>

License LGPL

biocViews GraphsAndNetworks, Proteomics

R topics documented:

`arp23` *Constituent members of Protein Complex Arp 2/3*

Description

A character vector of the constituent members of Arp 2/3 given by the common names.

Usage

```
data(arp23)
```

Format

The format is: chr "arp23"

Examples

```
data(arp23)
```

---

arp23G *The graph of arg 2/3*

Description

An instance of the class graph of the protein complex ARP 2/3

Usage

```
data(arp23G)
```

Format

The format is: chr "arp23G"

Examples

```
data(arp23G)
```

---

arp23Orf *Constituent members of Protein Complex Arp 2/3*

Description

A character vector of the constituent members of Arp 2/3 given by the systematic gene names

Usage

```
data(arp23Orf)
```

Format

The format is: chr "arp23Orf"

Examples

```
data(arp23Orf)
```

arp23Y2HG — *A graph of ARP 2/3 containing only Y2H verified interactions*

Description

An instance of the class graph for the protein complex Arp 2/3's Y2H verified edges.

Usage

```
data(arp23Y2HG)
```

Format

The format is: chr "arp23Y2HG"

Examples

```
data(arp23Y2HG)
```

calcGraphStats — *A function to calculate the various summary statistics for Y2H induced graphs*

Description

This function takes a putative protein complex (given by comp) and a list of bait to prey associations, and from these two data-sets computes various summary statistics on the induced Y2H graph which include: edge proportion of sampled nodes (baits to prey); estimated population mean nodal degree; proportion of sampled baits with hits to baits with no hits; the average out degree of baits; the proteins (bait and prey) which are not isolated; the number of connected components to the y2h graph, etc.

Usage

```
calcGraphStats(comp, bait2PreyL)
```

Arguments

comp	The comp argument is a character vector containing the constituent proteins of some putative protein complex
bait2PreyL	A named list of lists. Each name represents a particular Y2H experiment; within each experimental entry, there is two more lists: 1. bpList and 2. expList. bpList is a named list of character vectors; each name corresponds to a bait protein of that particular experiment, and the enry is a chracater vecotor of the hit proteins found as prey. the expList is a named list concerning experimental data: ActD gives the activation domain; GW is a logical denoting if the prey list is genome wide; and numBaits details the number of baits used in the experiment.

Details

For the bait2PreyL; use the the intactInfo.rda data-set of the package y2hStat.

Value

`complex`	A character vector of the constituent members of the proteinn complex of interest
`complexBait`	A character vector of those protein members which were also tested as baits in the y2h experiments
`b2CompP`	A character vector of all proteins which were tested as baits in the y2h experiments and also found a prey in the complex of interest
`notBait`	A character vector of those protein members which were not tested as baits
`avgDegOut`	The average out-degree of the bait proteins of the complex of interest
`notIsolated`	A character vector of proteins that showed experimental binary affiliation for some other complex member
`y2hGraph`	An instance of the class graph; a graphNEL induceds Y2H graph on the protein complex
`popMeanDegree`	The population mean degree derived from the bait proteins
`edgeProp`	The estimated edge proportions derived between undirected Y2H edges
`degBait`	A named list; each name corresponds to each member of the protein complex with the entry the number of proteins to which it is adjacent

Author(s)

T Chiang

---

cfia *Constituent members of Protein Complex Cleavage Factor IA (CFIA)*

Description

A character vector of the constituent members of Cleavage Factor IA given by the common names.

Usage

```
data(cfia)
```

Format

The format is: chr "cfia"

Examples

```
data(cfia)
```

cfiaOrf *Constituent members of Protein Complex Cleavage Factor IA (CFIA)*

Description

A character vector of the constituent members of Cleavage Factor IA given by the systematic gene names

Usage

```
data(cfiaOrf)
```

Format

The format is: chr "cfiaOrf"

Examples

```
data(cfiaOrf)
```

checkComplex *Function to check a list of protein complexes wrt ScISI*

Description

This function takes a named list of protein complexes (where each protein complex name indexes an item of the list and each list entry is the protein composition of the complex) and an interactome (in the bi-partite matrix representation) and checks to see if each complex of the list is either in the interactome, a sub-complex of some complex(es) of the interactome, or a super-complex of some complex(es) of the interactome.

Usage

```
checkComplex(comps, interactome)
```

Arguments

comps	A named list of protein complexes. The names corresponds the protein complexes and the entries correspond to the complex composition
interactome	A bi-partite graph representation of some interactome

Details

This is another QC function to check the participation of certain protein complexes. The out-put will describe how the various protein complexes in question relate to the protein complexes of the interactome in question.

Value

A named list:

`equal`	A named character vector - the name equals the entry
`subset`	A named char vector - the name is a subset of the entry
`superset`	A named char vector - the name is a superset of the entry
`sameNameNotEqual`	A named char vector - the name is equal to the name of the entry but the composition is not equal

Author(s)

T Chiang

`checkSGN` *A function to check that the protein names are all systematic gene names.*

Description

This function takes the in silico interactome and checks the rownames against the names of the meta-data set org.Sc.sgdALIAS from the meta-data package org.Sc.sgd

Usage

```
checkSGN(ISI)
```

Arguments

`ISI`	The in silico interactome as a bipartite graph

Value

A character vector of potential gene names not found to be a systematic gene name

Author(s)

T. Chiang

compareComplex *A function to compare two bipartite graph matrices*

Description

The compareComplex function takes two bipartite graph matrix representations and calculates three statistics between all pairs of complexes, C-i and K-j: (1) the intersection between C-i and K-j, (2) the proteins in C-i and not in K-j, (3) the proteins in K-j and not in C-i. These stats are later used to calculate Jaccard and Dice-Sorenson Coefficients as well as probability distributions of a protein p in a complex C.

Usage

```
compareComplex(BGMat1, BGMat2)
```

Arguments

BGMat1	The first bipartite graph matrix
BGMat2	The second bipartite graph matrix

Details

The main point to remember is that we don't compare complexes within the same matrix. We only compare pairs of complexes from the BGMat1 and BGMat2. The runCompareComplex function must be called first since all the labelled vertices present in BGMat1 must be present in the BGMat2 and vice versa so a true comparison can be made.

Value

A list with the following entries:

intersect	A matrix of pairwise intersections of the complexes of BGMat1 and BGMat2
cminusk	A matrix of the setdiff of the complexes of BGMat1 and those of BGMat2
kminusc	A matric of the setdiff of the complexes of BGMat2 and those of BGMat1

Author(s)

Tony Chiang

`compBijection` *A recursive function that greedily handles the alignment issue*

Description

This function takes a matrix of similarity measures (e.g. Jaccard Index) between the TSNMat and the estMat and finds the maximal value of this matrix and records its position (i,j). Then it matches C-i to K-j and then deletes row i and colunm j creating a matrix with one less row and one less colunm. The function calls itself recursively using this smaller matrix as the new argument. It stops when there are either no rows left or no columns left or the matrix of similarity measures is reduced to a 0-matrix and breaks from the recursive loop.

Usage

```
compBijection(TSNMat, estMat, c2kMatrix, bijMat, counter = 1)
```

Arguments

`TSNMat`	The first bipartite graph matrix
`estMat`	The second bipartite graph matrix
`c2kMatrix`	A matrix of similarity measure
`bijMat`	A recording matrix to keep the alignment
`counter`	A place keeping index

Details

The function creates a greedy matching between two bipartite graphs (where complexes C-i of the first bipartite graph matrix, bg1, is matched to complexes K-j of the second bipartite graph matrix, bg2). The particular greedy algorithm is as follows:

1. When the function is called, the parameter c2kMatrix is parsed and the maximal element, m, is found. If m is not unique in the matrix, the function looks to every position where m occurs: (i,j)...(m,n). To chose a particular position, the size of the complexes are taken into consideration, i.e. the function compares the cardinalities of (C-i + K-j) ... (C-n + K-n). And the pair, wlog (i,j), of complexes with the largest cardinality is selected (if there is again a tie amongst cardinalities, then a random choice is made).

2. The function matches C-i to K-j and records this alignment into bijMat. Row i and colunm j is deleted from c2kMatrix, creating a new matrix called c2kM.

3. If the dimension of this matrix is nonzero or if the matrix itself has some nonzero element, the function recursively calls itself with the new argument, c2kM.

4. Because the dimension is always decreased with every call, this function must terminate in some finite number of steps.

5. bijMat will have recorded the greedily matched complexes between bg1 and bg2.

Value

A matrix with the rows recording the alignment: the first colunm records complexes of TSNMat; the second column records complexes of estMat; and the third column records the similarity measure. The rows will also denote the ordering of the matchings, i.e. row 1 will denote the first match, etc.

Author(s)

Tony Chiang

`createGODataFrame` *A function to create a Dataframe from the GO protein complexes*

Description

This function takes a named list (the names are the GO ID's representing some protein complex and the list elements are character vectors consisting of the members of each particular complex) and an Bipartite Graph Incidence Matrix and creates a dataframe with three columns: complex name, GO ID, and complex description.

Usage

```
createGODataFrame(cMembers, goMat)
```

Arguments

`cMembers`	Named list with GO ID's as names and character vector consisting of proteins of the respective complex
`goMat`	Bipartite Graph Incidence Matrix

Details

This function takes the output parameters of `createGOMatrix` and `getGOInfo` and creates the bipartite graph incidence matrix.

Value

Dataframe:

`names`	The names of the GO Complexes
`ID`	The GO IDs' of the complexes
`description`	The description give by GO

Author(s)

Tony Chiang

References

www.geneontology.org

Examples

```
#go = getGOInfo(wantAllComplexes = FALSE)
#goM = createGOMatrix(go)
#createGODF(go, goM)
```

createGOMatrix *A function to create the bipartite graph (BG) incidence matrix from the protein complexes parsed from the GO Database*

Description

This function takes the output from the getGOInfo function and creates the bipartite graph (BG) incidence matrix from the output.

Usage

```
createGOMatrix(cMembers)
```

Arguments

cMembers	A named list consisting of vectors. Each list item is named by a protein complex GO ID and points to a character vector where each character vector consists of the proteins composing of that protein complex.

Value

A bipartite graph matrix of the corresponding GO protein complexes where the rows are indexed by the protein names and the colunms by GO ID's.

Author(s)

Tony Chiang

Examples

```
#cMembers = getGOInfo(wantAllComplexes = FALSE)
#createGOMatrix(cMembers)
```

createMipsDataFrame *A function that creates a data frame from the MIPS Data*

Description

This function takes two parameters: (1) a named vector that has a description of the MIPS protein complexes (this vector is one of the two outputs from getMipsInfo referenced by DESC); (2) the matrix representation of the bipartite graph. The output of the function is a dataframe with three columns: the names of the complexes in the matrix; the ID of the corresponding complex; the description of the corresponding complex.

Usage

```
createMipsDataFrame(desc, mips)
```

Arguments

desc	A named character vector (where the name is the MIPS ID for each protein complex obtained by getMipsInfo) whose entries describe the corrsponding protein complex.
mips	The matrix representation of the protein complex membership graph for the MIPS protein complexes

Value

The return value is a data frame with three coluumns: the names column records the names of the protein complexes as indexed in the incidence matrix; the ID column records the MIPS ID that corresponds to each protein complex; the Desc column describes each of the protein complexes.

Author(s)

Tony Chiang

References

mips.gsf.de

---

createMipsMatrix *A function to create the bipartite graph incidence matrix from MIPS protein complexes*

Description

This function takes the output from the getMipsInfo function and creates the bipartite graph incidence matrix where the rows are indexed by proteins and colunms by protein complexes (given by the MIPS ID's).

Usage

```
createMipsMatrix(mipsL)
```

Arguments

mipsL	A list consisting of the two items: mipsL$Mips is a named list of character vectors. The names correspond to MIPS protein complexes and the character vector correspond to the proteins within that complex; and mipsL$DESC is a named character vector where the names are the protein complex MIPS ID and the values is the description of the complex.

Value

A bipartite graph incidence matrix of the MIPS protein complexes where rows are indexed by proteins names and colunms by the MIPS ID's pre-fixed with "MIPS-".

Author(s)

Tony Chiang

References

mips.gfs.de

Examples

```
#mips= getMipsInfo(wantSubComplexes = FALSE)
#mipsM = createMipsMatrix(mips)
```

---

createYeastDataObj *Creates an object of class yeastData*

Description

Creates an instance of the class yeastData.

Usage

```
createYeastDataObj(dataFrameISI)
```

Arguments

dataFrameISI	A dataframe with three slots: complex name (from the incidence matrix), complex ID (from some database), complex description.

Value

An object of class yeastData.

Author(s)

Tony Chiang

Examples

```
#mips = createMipsMatrix(wantList=TRUE)
#mipsDF = createMipsDF(mips$DESC, mips$Matrix)
#mipsOb = createYeastDataObj(mipsDF)
```

dataS *A character matrix containing the source data for the ScISI*

Description

This data object is a matrix that gives the description of each of the data repositories used to generate the ScISI.

Usage

```
data(dataS)
```

Format

The format is: chr "dataS"

Examples

```
data(dataS)
```

Desc *A method to return a description of a protein complex*

Description

The `Desc` method takes in the protein complex name of a bi-partite graph incidence matrix (usually an ad hoc name) and returns a description of that protein complex.

The object to be referenced is an instance of the class yeastData.

Usage

```
Desc(object, name)
```

Arguments

`object`	An instance of a subclass of `yeastData`.
`name`	A `character`. It is the name of the protein complex with respect to the bi-partite graph incidence matrix

eAt *An edge attribute data file*

Description

This file contains edge attributes used in the rendering of the graph ARP 2/3 in the ScISI.Rnw file.

Usage

```
data(eAt)
```

Format

The format is: chr "eAt"

Examples

```
data(eAt)
```

eAt2 *A file containing edge attributes*

Description

This file contains edge attributes used in the rendering of the graph ARP 2/3 in the ScISI.Rnw paper.

Usage

```
data(eAt2)
```

Format

The format is: chr "eAt2"

Examples

```
data(eAt2)
```

edgeProp *A function to estimate the edge proportion of a y2h induced graph on a protein complex*

Description

This function first creates the Y2H symmetric adjacency matrix on a protein complex (as we ignore directionality). Then it calculates an estimate for the proportion of edges the complex might have with high probability (if the sampling is unbaised and without error). In addition to this statistic, it also generates an instance of graph, specifically a graphNEL, on the adjcency matrix.

Usage

```
edgeProp(comp, compB2P, sampled)
```

Arguments

comp	A character vector of the constitutent members of the protein complex of interest
compB2P	A named list; the names correspond to constituent proteins used as baits and the entries are the found hits (prey)
sampled	A character vector of constituent members used as baits

Value

eProp	The estimated edge proportion of the complex
y2hGraph	A graphNEL instance of graph

Author(s)

T Chiang

egEBI16112 *A graph example mapping an IntAct ID to Systematic Gene Names*

Description

This file is an instance of the class graph showing the mapping of the IntAct ID EBI-16112 to multiple Systematic Names.

Usage

```
data(egEBI16112)
```

Format

The format is: chr "egEBI16112"

Examples

```
data(egEBI16112)
```

expStats — *A data file containing the experimental statistics*

Description

This file is a matrix that details the number of protein complexes found within each data repository (MIPS, GO, Gavin, Ho, Krogan) as well as the number of genes present (pre-processing).

Usage

```
data(expStats)
```

Format

The format is: chr "expStats"

Examples

```
data(expStats)
```

findSubComp — *A function that looks for either equality between two complexes or complete containment of one complex in another*

Description

This function determines if either a complex C-i of bg1 is equal to some complex K-j of bg2 or if a complex C-i (or K-j) is a subcomplex to K-j (or C-i).

Usage

```
findSubComp(bg1, bg2, interSectMat, simMat)
```

Arguments

bg1	The first bipartite graph matrix
bg2	The second bipartite graph matrix; bg2 may be equal to bg1
interSectMat	A matrix whose rows and columns are indexed by protein complexes: the rows are indexed by complexes of bg1 (colnames of bg1) and the columns indexed by complexes of bg2 (colnames of bg2). The (i,j) entries is the cardinality of the intersection between the i-th complex of bg1 and the j-th complex of bg2.
simMat	A matrix of similarity measures (e.g. Jaccard Index) between two bipartite graphs where the rows are indexed by complexes of the first bipartite graph, bg1, and the columns are indexed by the complexes of the second bipartite graph, bg2. The (i,j) entry of this matrix would be the similarity between C-i of bg1 and K-j of bg2.

Details

This function uses interSectMat and simMat to determine which complexes C-i of bg1 is equal to which complexes K-j of bg2. When the function finds an equality, C-i = K-j, it produces 5 statistics:

1. The name of C-i 2. The name of K-j 3. The cardinality of C-i 4. The cardinality of K-j 5. The cardinality of (C-i intersect K-j)

When all the equality of complexes have been found, this function also uses interSectMat and simMat to find which complexes C-i (or K-j) completely contain complexes K-n (or C-m respectively), i.e. which complexes are subcomplexes. It also returns the 5 statistics listed above.

NB - from the 5 statistics above, it will be clear if the relationship between the complexes is that of equality or sub-ordination.

Value

`record1`	A list of lists. The value of each sublist is: 1. a complex of bg1, 2. a complex of bg2, 3. the cardinality of the complex of bg1, 4. the cardinality of the complex of bg2, 5. the cardinality of these two complexes mutual intersection.
`record3`	A list of lists. The value of each sublist is: 1. a complex of bg1, 2. a complex of bg2, 3. the cardinality of the complex of bg1, 4. the cardinality of the complex of bg2, 5. the cardinality of these two complexes mutual intersection.
`toBeRm1`	A character vector of the names of complexes in bg1 that equal some complex in bg2. These complexes should be deleted to avoid redundancy if these two bipartite graphs are merged.
`toBeRm2`	A character vector of the names of complexes from either bg1 or bg2 which are sub-complexes of another protein complex.

Author(s)

Tony Chiang

Examples

```
#go = getGOInfo()
#mips = getMipsInfo()
#goM = createGOMatrix(go)
#mipsM = createMipsMatrix(mips)
#cc = runCompareComplex(mipsM, goM, byWhich = "ROW")
```

---

gavin2mergeMG *A data file containing the pre-merged Gavin to (merged) MIPS-GO data*

Description

This file contains the pre-processed Gavin data before merging to the joint MIPS-GO interactome.

Usage

```
data(gavin2mergeMG)
```

Format

The format is: chr "gavin2mergeMG"

Examples

```
data(gavin2mergeMG)
```

`getAPMSData` *A function to get the estimated complexes from high throughput data determined by the package apComplex.*

Description

This function will generate the bipartite graph (BG) incidence matrix from either the experiments of Gavin, Ho, or Krogan.

Usage

```
getAPMSData(author = NULL)
```

Arguments

`author`	A character - name of the lead experimentor. It can be either "Gavin", "Ho", or "Krogan"

Details

This function is called to gain access to the protein complex co-membership bipartite graph matrices stored in the apComplex package.

The matrix that is returned will be an incidence matrix with yeast standard gene name indexing the rows and ad hoc protein complex names indexing the columns.

The matrix that is obtained from apComplex database will be the multi-bait, multi-edge (MBME) estimate from each of the experiment. We have elected to take the MBME estimates rather than the cumulative estimates because the MBME's are generated from more statistically significant data sets.

Value

BG Incidence Matrix with rows indexed by proteins and columns by complexes

Author(s)

Tony Chiang

Examples

```
getAPMSData("Gavin")
```

`getGOInfo` *A function that parses through the GO database; it agreps for the term "complex" and greps suffixes "-ase" and "-some" and returns nodes whose description contains such terms.*

Description

This function parses through the Cellular Component ontology for the GO nodes and searchs for the term "complex" or the suffix "-ase" (e.g. RNA Polymerase) or "-some" (e.g. ribosome) and (or) other user defined phrases in the description of these nodes.

Usage

```
getGOInfo(wantDefault = TRUE, toGrep = NULL,
parseType=NULL, eCode = NULL, wantAllComplexes = TRUE,
includedGOTerms=NULL, not2BeIncluded=NULL)
```

Arguments

`wantDefault`	A logical. If TRUE, the default parameters ("complex", "\Base\b" and "\Bsome\b") are used.
`toGrep`	A character vector of the phrases (with Perl regular expressions) for which the function will parse through the GO database and search.
`parseType`	A character vector.This vector is in one to one correspondence with toGrep; it takes in the parse type such as "grep", "agrep", etc.
`eCode`	A character vector of evidence codes (see "http://www.geneontology.org/GO.evidence.shtml" for details). The function will disallow any protein inclusion in the protein complexes if they are not indexed by evidence code other than those found in eCode.
`wantAllComplexes`	A logical. If TRUE, the function will incorporate all GO children of the nodes found by term searches. In addition, children of node GO:0043234 (the protein complex node) will also be incorporated.
`includedGOTerms`	A character vector of GO terms that will be parsed regardless of the default and parseType parameters. In essence, these GO terms forced to be included.
`not2BeIncluded`	A character vector of GO terms that should not be parsed nor included in the output.

Details

This function's generic operation is to parse the GO database and search for pre-determined or chosen terms. It returns a named list of chracter vectors where the names are GO id's from the CC ontoloy and the vectors consist of proteins corresponding to that particular GO id. Running this function has multiple combinations:

1. If the wantDefault parameter is TRUE, the function will agrep for "complex" and grep for "\Base\b" and "\Bsome\b".

2. If toGrep is not NULL, it will be a character vector with terms and perl regular expressions that are intended for searching in the GO database. NB - it toGrep is not NULL, then parseType should also not be NULL as the parseType indicates how each term should be searched.

3. parseType needs to be supplied if toGrep is not NULL. It is a character vector, either a single entry or of length equal to the length of toGrep, detailing how each term in toGrep will be parsed in the GO database. If only one term is supplied for parseType, then all the terms in toGrep will be parsed identically. Otherwise, the i-th term in parseType will reflect the parsing of the i-th term in toGrep.

4. The eCode argument is a user determined refining mechanism. It takes in a vector of evidence codes (as detailed by the GO website). The function will dis-allow proteins if and only if these proteins are only indexed by evidence codes found within eCodes.

5. If wantAllComplexes parameter is True, the function will also return the children of nodes found by parsing terms. In addition, the children of GO ID GD:0043234 (the protein complex ID) will be returned. The union of complexes is then returned.

Value

The return value is a list of size n (n depends on the current status of the GO database) where the name of each list element is a GO ID and each list element itself is a character vector consisting of the proteins corresponding to a particular GO ID:

`"GO:XXXXXXX"`	A character vector containing proteins (not indexed by only eCode evidence codes) which make up protein complex "GO:XXXXXXXX"

Author(s)

Tony Chiang

References

www.geneontology.org

Examples

```
#go = getGOInfo(wantAllComplexes = FALSE)
#goCoded = getGOInfo(code = c("IPI","ND","IDA"))
#goPhrase = getGOInfo(wantDefault = FALSE, toGrep = "\Bsomal\b",
#parseType = "grep", wantAllComplexes = FALSE)
#nam1 = names(go)
#nam2 = names(goCoded)
#if(length(nam1) == length(nam2) && nam1 == nam2){
#sapply(nam1, function(x) setdiff(go[[x]], goCoded[[x]]))
#}
```

---

getLocOrfs *A function to obtain ORFs for the ScISI*

Description

This function takes a list of bi-partite graph incidence matrices as well as a vector of GO nodes, and for each protein complex of each bi-partite graph, finds those constituent member proteins which are annotated to each GO node.

Usage

```
getLocOrfs(imList, goNode, pathToSave = NULL, name = NULL)
```

Arguments

imList	A list of bi-partite graph incidence matrices
goNode	A character vector of GO ID's
pathToSave	A character denoting where to save the resultant data
name	The name by which the resulting R-object as well as the file will be named

Details

Currently, this function will take a protein complex and find those constituent protein members which have also been annotated to the user specified GO ID. For instance, the GO ID "" is the nucleus ID. This function will find all proteins within an interactome which has been annotated with the ID "". NB - if n GO ID is supplied, the function returns n different interactomes, each is has proteins uniquely annotated with each GO ID.

Value

The return value is a list of lists of lists:

1. The top level list contains entries named by each GO ID 2. The second tier level contains entries after each entry of imList 3. The third tier level contains three entries: i. A bi-partite graph incidence matrix which only has protein complexes restricted to the respective GO ID annotation ii. A bi-partite graph incidence matrix which only has protein complexes listed in (i) but contains all the original proteins iii. A vector of proportions. We look at the column sums of (i) divided by the column sums fo (ii). This gives the ratio of proteins which are annotated by the GO ID over all other proteins in that complex

restrictedOrfsComp	A bi-partite graph incidence matrix which only has protein complexes restricted to the respective GO ID annotation
restrictedOrfsOnly	A bi-partite graph incidence matrix which only has protein complexes listed in (i) but contains all the original proteins
ratio	A vector of proportions. We look at the column sums of (i) divided by the column sums fo (ii). This gives the ratio of proteins which are annotated by the GO ID over all other proteins in that complex

Author(s)

TC

References

~put references to the literature/web site here ~

getMipsInfo *A function that reads the downloaded text file from the MIPS repository and generates a named list of protein complexes.*

Description

This function reads the downloaded text file from the MIPS database and parses the file for those collection of proteins either referred to as a "complex", an "-ase" (e.g. RNA Polymerase), or a "-some" (e.g. ribosome) and (or) user supplied terms as the protein complex of interest. It returns a list containing two items: a named list of protein complexes and a character vector (of the same length as the named list) describing each protein complex.

Usage

```
getMipsInfo(wantDefault = TRUE, toGrep = NULL,
parseType = NULL, eCode = c("901.01.03", "901.01.03.01", "901.01.03.02",
                 "901.01.04", "901.01.04.01", "901.01.04.02",
                 "901.01.05", "901.01.05.01", "901.01.05.02",
                 "902.01.09.02", "902.01.01.02.01.01",
                 "902.01.01.02.01.01.01", "902.01.01.02.01.01.02",
                 "902.01.01.02.01.02", "902.01.01.02.01.02.01",
                 "902.01.01.02.01.02.02", "902.01.01.04",
                 "902.01.01.04.01", "902.01.01.04.01.01",
                 "902.01.01.04.01.02", "902.01.01.04.01.03",
                 "902.01.01.04.02", "901.01.09.02"), wantSubComplexes=TRUE,
 ht=FALSE, dubiousGenes=NULL)
```

Arguments

wantDefault	A logical. If true, the default parameters "complex", "\Base\b" and "\Bsome\b" are grepped.
toGrep	A character vector. Each entry is a term with perl regular expressions which are intended to be searched in the Mips text file.
parseType	A character vector. Each entry is a term that tells how each entry of toGrep should be parsed; e.g. "grep" or "agrep"
eCode	A character vector. The evidence code is given in the file evidence.scheme found in the inst/extdata section of the package.
wantSubComplexes	A logical.If FALSE, the function only returns aggregate protein complexes. If TRUE, the function will also return subcomplexes as well.
ht	A logical. If FALSE, the function will not extract protein complex estimates obtained from high throughput analysis.
dubiousGenes	A character vector of genes that will be removed when parsing the MIPS repository.

Details

This function's generic operation is to parse the Mips protein complex database (as given by the downloaded text file) and search for pre-determined or chosen terms. It returns a named list of

chracter vectors where the names are MIPS id's from the protein complex sub-category and the vectors consist of proteins corresponding to that particular MIPS id. Running this function has multiple combinations:

1. If the wantDefault parameter is TRUE, the function will grep for "complex", "\Base\b", and "\Bsome\b".

2. If toGrep is not NULL, it will be a character vector with terms and perl regular expressions that are intended for searching in the MIPS database. NB - it toGrep is not NULL, then parseType should also not be NULL as the parseType indicates how each term should be searched.

3. parseType needs to be supplied if toGrep is not NULL. It is a character vector, either a single entry or of length equal to the length of toGrep, detailing how each term in toGrep will be parsed in the GO database. If only one term is supplied for parseType, then all the terms in toGrep will be parsed identically. Otherwise, the i-th term in parseType will reflect the parsing of the i-th term in toGrep.

4. The eCode argument is a character vector consistin of MIPS evidence codes. A protein will be removed from the protein complex is ALL the evidence codes used to annotate the protein are supplied in the eCode argument; otherwise, it is left in the complex.

5. If wantSubComplexes parameter is True, the function will return the sub-groupings (sub-complexes or sub-structures) as given by the clusterings in the MIPS protein complex database.

6. If ht parameter is True, the function will return the will return those protein complex estimates obtained from high throughput analysis as well.

Value

The return value is a list -

`Mips`	A named list of the protein complexes. Each list entry is denoted by some particlar MIPS ID (with the pre-fix "MIPS-") attachedand points to a character vector which are the members of that protein complex
`DESC`	A named chracter vector describing each protein complex parsed by the function. (The names are the MIPS ID)

Author(s)

Tony Chiang

References

mips.gsf.ed

Examples

```
#mips = getMipsInfo(wantSubComplexes = FALSE)
#mipsPhrase = getMipsInfo(wantDefault = FALSE, toGrep = "\Bsomal\b",
#parseType = "grep", wantSubComplexes=FALSE)
```

getURL *A method to return an url location of a protein complex*

Description

The getURL method takes in the protein complex name of a bi-partite graph incidence matrix (usually an ad hoc name) and returns the MIPS, GO, etc url containing all information of that protein complex.

The object to be referenced is an instance of the class yeastData.

Usage

```
getURL(object, name)
```

Arguments

object	An instance of a subclass of yeastData.
name	A character. It is the name of the protein complex with respect to the bi-partite graph incidence matrix

graphSumStats *An initiation function to generate graph statistics*

Description

This function takes an in silico interactome and a list of Y2H experimental data (see details) and generates various graph summary statistics.

Usage

```
graphSumStats(ISI, bait2PreyL)
```

Arguments

ISI	An incidence matrix representation of a bi-partite graph of an in silico interactome
bait2PreyL	A list of y2h information. The data structure is that of intactInfo.rda of the y2hStat package

Value

complex	A character vector of the constituent members of the proteinn complex of interest
complexBait	A character vector of those protein members which were also tested as baits in the y2h experiment
notBait	A character vector of those protein members which were not tested as baits
avgDegOut	The average out-degree of the bait proteins of the complex of interest

notIsolated	A character vector of proteins that showed experimental binary affiliation for some other complex member
y2hGraph	An instance of the class graph; a graphNEL induceds Y2H graph on the protein complex
popMeanDegree	The population mean degree derived from the bait proteins
edgeProp	The estimated edge proportions derived between undirected Y2H edges
degBait	A named list; each name corresponds to each member of the protein complex with the entry the number of proteins to which it is adjacent

Author(s)

T Chiang

---

ho2mergeMGG *A data file containing the pre-merged Ho to the (merged) MIPS-GO-Gavin data*

Description

This file contains the pre-processed Ho data before merging to the joint MIPS-GO-Gavin interactome.

Usage

```
data(ho2mergeMGG)
```

Format

The format is: chr "ho2mergeMGG"

Examples

```
data(ho2mergeMGG)
```

ID *A method to return the ID of a protein complex*

Description

The ID method takes in the protein complex name of a bi-partite graph incidence matrix (usually an ad hoc name) and returns its MIPS, GO, etc id. The object to be referenced is an instance of the class yeastData.

Usage

```
ID(object, name)
```

Arguments

object	An instance of a subclass of yeastData.
name	A character. It is the name of the protein complex with respect to the bi-partite graph incidence matrix

---

JaccardCoef *A function to calculate the Jaccard similarity index between two sets*

Description

The JaccardCoef function takes the return values of compareComplex function and calculates, for each pair of complexes C-i and K-j (where C-i is in first bipartite graph matrix and K-j is second), the similarity coefficient of Jaccard.

Usage

```
JaccardCoef(dataMat)
```

Arguments

dataMat	A list which is the output from compareComplex, which is a list of three matrices: intersect, cminusk, and kminusc which are explained in the details.

Details

The argument of this function is a list of three matrices all of whom are indexed exactly in the same manner - the rows of each of the matrix is indexed by the complexes, {C-i}, of the first bipartite graph, bg1, and the colunms are indexed by the complexes, {K-j} of the second bipartite graph, bg2.

The first matrix of the list is the intersect matrix, I. The (i,j) entry of I is the cardinality of complex C-i of bg1 and K-j of bg2.

The second matrix of the list is the cminusk matrix, Q. The (i,j) entry of Q is the cardinality of the set difference between C-i and K-j.

The third matrix of the list is the kminusc matrix, P. The (i,j) entry of P is the cardinality of the set difference between K-j and C-i.

The Jaccard Coefficient between two sets (here between two complexes) C-i and K-j is given by the quotient of cardinality(C-i intersect K-j) and cardinality(C-i union K-j). Note that cardinality(C-i intersect K-j) is the (i,j) entry of I, and that cardinality(C-i union K-j) is the sum of the (i,j) entry of I, Q, P.

Value

The return value is a matrix consisting of the Jaccaard coefficient for each pair of complexes C-i and K-j with rows in indexed by C-i and columns indexed by K-j.

Author(s)

Tony Chiang

Examples

```
#go = getGOInfo(wantAllComplexes=FALSE)
#mips = getMipsInfo(wantSubComplexes=FALSE)
#goM = createGOMatrix(go)
#mipsM = createMipsMatrix(mips)
#cc = runCompareComplex(mipsM, goM, byWhich = "ROW")
#cc$simInd
```

`krogan2mergeMGGH` *A data file containing the pre-merged Krogan to the (merged) MIPS-GO-Gavin-Ho data*

Description

This file contains the pre-processed Krogan data before merging to the joint MIPS-GO-Gavin-Ho interactome

Usage

```
data(krogan2mergeMGGH)
```

Format

The format is: chr "krogan2mergeMGGH"

Examples

```
data(krogan2mergeMGGH)
```

`locScISI` *A data file used to estimate the location of the complexes of the ScISI*

Description

This is a three tier-ed list. The top list contains two sub-lists which have either the nucleus GO term or the cytoplasm GO terms as reference. For each sub-list, we have three entries: 1. A matrix indexed by only nuclear proteins in the rows and only non-trivial complexes with respect to the row indices; 2. A matrix with all proteins of the original ScISI indexing the rows, but the same complexes of (1) indexing the columns; 3. a numeric vector with the ratio of the column sums of (1) by the column sums of (2). This is used to derive nuclear protein complexes - those complexes with 75 percent or more of its constituent members belonging to the nucleus.

Usage

```
data(locScISI)
```

Format

The format is: chr "locScISI"

Examples

```
data(locScISI)
```

`mapping2SysG` *A example graph of the mapping from IntAct to Systematic Names*

Description

A instance of class graph used in the ScISI to show an example of the mapping from IntAct to Systematic Names

Usage

```
data(mapping2SysG)
```

Format

The format is: chr "mapping2SysG"

Examples

```
data(mapping2SysG)
```

`mappingsG` *A example graph of the mapping from IntAct to Systematic Names*

Description

A instance of class graph used in the ScISI to show an example of the mapping from IntAct to Systematic Names

Usage

```
data(mappingsG)
```

Format

The format is: chr "mappingsG"

Examples

```
data(mappingsG)
```

`maximizeSimilarity` *A function compares two bipartite graph matrices and finds the most similar matches between the clusters*

Description

This function takes a matrix of similarity indices between two bipartite graph matrices and determines, for each complex of the first bipartite graph matrix (bg1), the most similiar complexes of the second bipartite graph matrix (bg2).

Usage

```
maximizeSimilarity(simMat, bywhich = "ROW", zeroSim = "NO")
```

Arguments

`simMat`	A matrix of similarity coefficients between two bipartite graph matrices (bg1 and bg2) where the rows are indexed by the first bipartite graph matrix and the colunms are indexed by the second bipartite graph matrix. The (i,j)th entry is the similarity index between complex i of bg1 and complex j of bg2.
`bywhich`	Takes one of these three arguments: "ROW", "COL", "BOTH"
`zeroSim`	Takes either one of the following arguments: "NO", "YES"

Details

This function's purpose is to take one (or both) bipartite graph matrix, wlog we take bg1, and, for each complex, C-i, of bg1, finds the complex(es) of bg2 that are the most similiar to C-i based on the similarity index. Since the complexes of bg1 is indexed by the rows of simMat argument, finding the complexes of bg2 that are the most similar to C-i means finding the maximal value, m, of row i and then the complexes, K-j, that index the colunm for which m belongs.

If byWhich argument is set to "ROW", the function parses through each complex of bg1 and finds the complex(es) of bg2 which are most similar. If byWhich is set to "COL", the function parses through each complex of bg2 and finds the complex(es) of bg1 which are most similar. If byWhich is set to "BOTH", the function parses through both the complexes of bg1 and bg2. Since this matrix is not symmetric (this matrix is usually not square) this maximizing is different between row and column.

If zeroSim argument is set to "NO", the only maximal matching occurs if the similarity index is nonzero; e.g. if we want to maximize the match for complex C-i of bg1, but row i is comprised only of 0, C-i will not be matched to any complex of bg2.

Value

The return value is a list consisting of a vector and a list:

`maximize`	A named numeric vector. The name is the complex, C-i, for which the function is trying to find a maximal match. The entries of the vector is the maximal similarity index between C-i and all of the complexes of the other bipartite graph matrix, i.e. the maximal entry row i in `simMat`.
`maxComp`	A named list of named vectors. The name is the complex, C-i, for which the function is trying to find a maximal match. The named vector consists of the positions of the maximal matches (either which row or which column) and the names correspond to the conmplex of maximal matching.

Author(s)

Tony Chiang

Examples

```
#go = getGOInfo(wantAllComplexes = FALSE)
#mips = getMipsInfo(wantSubComplexes = FALSE)
#goM = createGOMatrix(go)
#mipsM = createMipsMatrix(mips)
#cc = runCompareComplex(mipsM, goM, byWhich="ROW")
#m = maximizeSimilarity(cc$JC, byWhich = "ROW")
#m$maximize
#m$maxComp
```

---

meanDeg — *A function to estimate the population mean nodal degree of a protein complex of interest*

Description

This function uses the nodal degree from Y2H data and estimates the overall mean nodal degree for all proteins of a particular complex

Usage

```
meanDeg(comp, degBait, sampled)
```

Arguments

comp	A character vector of the constituent members of some protein complex of interest
degBait	A named list; each name is a constituent member and the entry is the out deg
sampled	A character vector of those constituent members tested as baits in the Y2H experiments

Value

A scalar - the estimate of the population mean nodal degree

Author(s)

TC

`mergeBGMat` *A function that merges two bipartite graph (BG) incidence matrices into one.*

Description

This function takes the union of the row names of mat1 and mat2 for the row names of the aggregate matrix, and takes the union of the complexes of mat1 and mat2. The resulting matrix is also an incidence matrix so an entry of unity impiles protein p is a member of complex C.

Usage

```
mergeBGMat(mat1, mat2, toBeRm)
```

Arguments

`mat1`	The first bipartite graph incidence matrix
`mat2`	The second bipartite graph incidence matrix
`toBeRm`	A character vector of complexes to be removed

Details

This function takes two bipartite graph matrices and merges them into one aggregate incidence matrix where informational redundancy is removed.

The rows of the aggregate matrix is indexed by the union of the rownames of mat1 and mat2. It is important that the rownames of mat1 and the rownames of mat2 are from the same name set (e.g. for yeast, only the standard gene names should be used).

The columns will be indexed by different protein complexes. If two protein complexes are identical, say C-i = K-j, then either C-i or K-j should be listed in toBeRm argument (given as an argument). When the matrices are merged, only one of the two will be kept. The vector toBeRm is generated by calling either `runCompareComplex` or `findSubComp`.

Value

An aggregate bipartite graph incidence matrix.

Author(s)

Tony Chiang

Examples

```
#go = getGOInfo()
#mips = getMipsInfo()
#goM = createGOMatrix(go)
#mipsM = createMipsMatrix(mips)
#cc = runCompareComplex(mipsM, goM, byWhich = "ROW")
#merged = mergeBGMat(mipsM, goM, cc$toBeRm)
```

mips2go — *A data file containing the pre-merged GO to the MIPS data*

Description

This file contains the pre-processed GO data before merging to the MIPS data.

Usage

```
data(mips2go)
```

Format

The format is: chr "mips2go"

Examples

```
data(mips2go)
```

nAt — *A file containing node attributes*

Description

This file contains node attributes used in the rendering of the graph ARP 2/3 in the ScISI.Rnw paper.

Usage

```
data(nAt)
```

Format

The format is: chr "nAt"

nAtMap — *A file containing node attributes*

Description

This file contains node attributes used in the rendering of the graph ARP 2/3 in the ScISI.Rnw paper.

Usage

```
data(nAtMap)
```

Format

The format is: chr "nAtMap"

Examples

```
data(nAtMap)
```

---

nonGenes *Genes found in MIPS which are not gene locus names*

Description

A character vector of the genes found in MIPS which are not gene locus names.

They include: RNA_TLC1, SNRNA_NME1, and RNA_RNASE-P.

Usage

```
data(nonGenes)
```

Format

The format is: chr "nonGenes"

Examples

```
data(nonGenes)
```

---

nucComp *A data file containing the nuclear complexes*

Description

This file is a character vector of the names of those protein complexes which have at least 75 percent of its constituent members annotated with the nucleus GO term.

Usage

```
data(nucComp)
```

Format

The format is: chr "nucComp"

Examples

```
data(nucComp)
```

recCompSize — *A function that records the relative sizes of complex C-i from one bipartite graph with complex K-j from a different bipartite graph.*

Description

This function takes two bipartite graph matrices, bg1 and bg2. For each complex C-i of bg1, we find the relative size of C-i for every complex K-j of bg2. A matrix of these ratios is returned with all cardinalities of C-i as the numerators and K-j as denominators. A second matrix is calculated where the cardinality of K-j is the numerator and C-i is the denominator.

Usage

```
recCompSize(bg1, bg2)
```

Arguments

bg1	The first bipartite graph as an incidence matrix
bg2	The second bipartite graph as an incidence matrix

Value

The return value is a list:

OneOverTwo	The matrix where the cardinalities of complexes from bg1 are numerators.
TwoOverOne	Matrix where cardinalities of complexes from bg2 are numerators.

Author(s)

Tony Chiang

Examples

```
#go = getGOInfo(wantAllComplexes = FALSE)
#goM = createGOMatrix(go)
#mips = getMipsInfo(wantSubComplexes = FALSE)
#mipsM = createMipsMatrix(mips)
#recCompSize(goM, mipsM)
```

redundantM — *A matrix of redundant complex summary statistics*

Description

This matrix contains the redundant protein complex summary statistics for the ScISI. The rows and columns are indexed by the individual data-set from where the protein complex estimates orginated. The (i,j) entry details how many protein complexes of the index i are identical to those with respect to index j. This matrix is symmetric.

Usage

```
data(redundantM)
```

Format

Matrix

Examples

```
data(redundantM)
```

`rmByEvi` *A function that parses through each GO protein complex and removes proteins based on evidence codes.*

Description

The set of proteins protKept argument is a filtered set of proteins based on evidence codes given by the users. If complex contains any protein not in protKept, that protein will be deleted.

Usage

```
rmByEvi(protKept, complex)
```

Arguments

`protKept`	A character vector of filtered proteins. Only the proteins listed in this vector should be kept in the protein complexes
`complex`	A list of the GO protein complexes

Details

When `getGOInfo` is called, the user can specify a set of evidence codes in the argument (see getGOInfo.Rd). The function `getGOInfo` then generates a set of proteins, protKept (which are indexed by evidence codes other than the user specified set of evidence codes), that are valid proteins.

This function takes this set of proteins and compares it with each GO protein complex. If any protein is found within a GO protein complex and does not belong to protKept, it is deleted from that protein complex.

Value

The return value is the "complex" list with a protein removed from the protein complex if that protein is not found in protKept.

Author(s)

Tony Chiang

`runAlignment` *A function to establish preliminaries for the compBijection function.*

Description

This function creates the record keeping bijMat and calls the `compBijection` function.

Usage

```
runAlignment(TSNMat, estMat, c2kMat)
```

Arguments

`TSNMat`	The first bipartite graph matrix
`estMat`	The second bipartite graph matrix
`c2kMat`	The matrix of similarity index between TSNMat and estMat

Value

The matrix bijMat with an alignment calculated by the function `compBijection`

Author(s)

Tony Chiang

Examples

```
#go = getGOInfo()
#mips = getMipsInfo()
#goM = createGOMatrix(go)
#mipsM = createMipsMatrix(mips)
#cc = runCompareComplex(mipsM, goM, byWhich = "ROW")
#align = runAlignment(mipsM, goM, cc$simInd)
```

`runCompareComplex` *A function that calls all other types of comparison functions*

Description

This function begins by acquiring the same vertex set for both BGMat1 and BGMat2. Then it calls the `compareComplex` function to calculate three statistics (see `compareComplex` man page for details). Next it calls calculates a similarity measure (default is Jaccard). Next it calls the `runAlignment` function to greedily calculate an alignment between BGMat1 and BGMat2.

Usage

```
runCompareComplex(BGMat1, BGMat2, index = "Jaccard", byWhich)
```

Arguments

BGMat1	The first bipartite graph matrix
BGMat2	The second bipartite graph matrix
index	The type of similarity index used
byWhich	A parameter for the maximizeSimilarity function. It can either be "ROW", "COL", or "BOTH"

Value

The return value of runCompareComplex is a list with the following statistics:

simInd	The matrix of similarity measures between two bipartite graphs.
maxIntersect	The output of maximizeSimilarity. A list. The first value is the largest similarity indices between complexes of the row and complexes of the columns if byWhich is "ROW" and vice versa if byWhich="COL". The second value returns the actual complex of highest similarity (column complexes if byWhich="ROW" and row complexes if byWhich="COL"
equal	A list of complexes of BGMat1 equal to complexes of BGMat2. The values of the list are the complex, C, of BGMat1, the complex, K, of BGMat2, the cardinality of C, the cardinality of K, and the cardinality of the intersection of C and K
toBeRm	A character vector of complexes that should be removed if two bipartite graphs are merged since they would represent redundant complexes. If the function is called on one bi-partite graph, then this vector looks for redundancies within one database and those complexes needs to be removed as well. These complexes are derived from BGMat1 only.
subcomplex	A list of complexes of BGMat1 which is contained in complexes of BGMat2 or vice versa. The values of the list are the complex, C, of BGMat1, the complex, K, of BGMat2, the cardinality of C, the cardinality of K, and the cardinality of the intersection of C and K
toBeRmSubC	A character vector. The same as toBeRm but instead of redundant protein complexes, this is a compilation of all the sub-complexes between two bi-partite graph matrices (bi-partite graph matrices need not be different)

Author(s)

Tony Chiang

Examples

```
#go = getGOInfo()
#mips = getMipsInfo()
#goM = createGOMatrix(go)
#mipsM = createMipsMatrix(mips)
#cc = runCompareComplex(mipsM, goM, byWhich = "ROW")
```

ScISI *The In Silico Interactome for Saccharomyces cerevisiae*

Description

This is the incidence sparse-matrix representatin for the bi-partite graph for the in silico interactome given by Saccharomyces cerevisiae. The rows are indexed by the systematic gene names and the colunmns are indexed by the protein complexes. This matrix contains a 1 in the (i,j) postion if the protein in indexed in the i-th row is a member of the protein complex of the j-th column; it contains a 0 otherwise.

ScISIC is a sub-interactome of ScISI which consists of protein complexes derived from small scale experiments and have been curated by Gene Ontology (GO) or MIPS. While these complexes have been curated, not all of them have been completely verified to be true. Periodic changes to GO and MIPS will percolate through the protein complexes or protein complex composition and so the ScISIC will need to be rebuilt. A script located in the inst/Scripts/ entitled createScISIC.R creates an updated ScISIC if the GO and MIPS's files are up to date. GO 2.0.0 and complexcat-data-2006 (MIPS) were used to build ScISIC for Bioconductor 2.1 release of the package ScISI. Both are the most up to date versions of the repository as of 9 October 2007. In addition to MIPS and GO, we have also incorporated protein complexes curated by the IntAct repository. The IntAct protein complexes are obtained from the complex data XML file obtained from IntAct via Rintact.

In addition to the protein complexes found within MIPS and GO, ScISIC also contains manually curated protein complexes obtained from IntAct. Because IntAct has yet to version its release, the protein complexes were obtained on 25 May 2007.

ScISIC replaces ScISIverified which has been deprecated.

ScISI combines ScISIC with protein complex estimates on the datasets derived by Gavin et al (2002), Ho et al (2002), and Krogan et al (2004) using the penalized algorithm found with apComplex developed by Scholtens et al (2004).

Usage

```
data(ScISI)
```

Format

The format of both the ScISI and ScISIC is a binary incidence matrix. The rows are indexed by the gene locus names and the columns are indexed by the identification codes for the protein complexes based on the repository from where they are obtained.

Details

This is the working in silico interactome built for computational experimentation. The data from which this interactome is built is from the Intact, Gene Ontology, Mips, and estimated protein complexes from apComplex.

Source

http://www.geneontology.org

References

http://www/bioconductor.org

Examples

```
data(ScISIC)
```

`ScISI2html` *A function that generates an html page for the GO and MIPs protein complexes*

Description

This function takes a vector or url's and a vector of the protein complex description names (known names of the protein complexes) and creates an html file that lets the user link to each of the protein complex description site based on the particular url.

Usage

```
ScISI2html(urlVect, linkName, filename, title,
othernames, table.head, table.center = TRUE, compSize=NULL)
```

Arguments

`urlVect`	A character vector containing the url for each speficied protein complex
`linkName`	A character vector containing a description name for each of the url's in urlVect that will be the anchor for the url
`filename`	The output file; an html file written to the home directory
`title`	The title given to the html file
`othernames`	Other titles that are needed; sub-titles
`table.head`	The title for the table
`table.center`	Logical; to center the table
`compSize`	A numeric vector of the complex sizes for the various protein complexes of GO and MIPS

Value

An html file written to the user's home directory. The file contains a page of links (given by the url's) where each link is anchored by a description name for each url.

Author(s)

Tony Chiang

Examples

```
#go = getGOInfo(wantAllComplexes = FALSE)
#goM = createGOMatrix(go)
#goDF = createGODataFrame(go, goM)
#goOb = createYeastDataObj(goDF)
#goNames = colnames(goM)
#url = vector(length = length(goNames))
#for(i in 1:length(goNames)){
#   url[i] = getURL(goOb, goNames[i])
#}
#ScISI2html(url, goNames, test, GO Complexes)
```

subCompM *A matrix of sub-complex summary statistics*

Description

This matrix contains the sub-complex summary statistics for the ScISI. The rows and columns are indexed by the individual data-set from where the protein complex estimates orginated. The (i,j) entry details how many protein complexes of the index i are sub-complexes of the index j.

Usage

```
data(subCompM)
```

Format

Matrix

Examples

```
data(subCompM)
```

sumStats *A function to calculate some summary statistics between an two interactomes*

Description

This function takes in a named list of in silico interactomes (by its incidence matrix representation of the bi-partite graph). The function compares each interactome pairwise (with itself as well as with each other interactome) and generates some summary statistics: e.g. the number of redundant protein complexes, the number of protein sub-complexes one interactome may posses with respect to some other interactome (possibly itself), etc.

Usage

```
sumStats(imList, pathToSave = NULL)
```

Arguments

imList	A named list of in silic interactomes (incidence matrix)
pathToSave	A character vector of a path location to where the summary statistics will be saved

Value

redundantM	A symmetric matrix with the row and column names named by the interactome names. The shows the number of redundancies (i.e. the number of repeated protein complexes) within two interactomes
subM	A matrix with the row and column names named by the interactome names. Each entry details how many protein sub-complexes are found within the interactome indexed by the row with respect to the interactome indexed by the column

Author(s)

TC

Examples

```
#gavin <- getAPMSData("Gavin")
#krogan <- getAPMSData("Krogan")
#imList <- vector("list", length=2)
#imList[[1]] <- gavin
#imList[[2]] <- krogan
#names(imList) <- c("Gavin", "Krogan")

#sumStats(imList)
```

---

unwanted *GO terms that are parsed but not protein complexes*

Description

A character vector of the GO terms parsed but known not to be protein complexes in GO.

The include: GO:0000262, GO:0000228, GO:0000775, GO:0010008, GO:0005792, GO:0005768, GO:0005769, GO:0005770, GO:0005777, GO:0005844, and GO:0001400.

Usage

```
data(unwanted)
```

Format

The format is: chr "unwanted"

Examples

```
data(unwanted)
```

unWantedComp *A function to manually remove protein complexes from some in silico interactome*

Description

This function takes in an in silico interactome by its bipartite graph representation and a character vector of complex ID's. The function parses through the column names of the in silico interactome and removes those columns whose names are found in the character vector.

Usage

```
unWantedComp(ISI, unwantedComplex = c("GO:0000262", "GO:0000228",
"GO:0000775", "GO:0010008", "GO:0005792", "GO:0005768", "GO:0005769",
"GO:0005770", "GO:0005777", "GO:0005844", "GO:0001400"), unwantedGenes =
c("RNA_TLC1", "SNRNA_NME1","RNA_RNASE-P"))
```

Arguments

ISI	The in silico interactome as a bipartite graph
unwantedComplex	A character vector. Each entry is a protein complex id for which should be removed from the interactome.
unwantedGenes	A character vector of gene names that will be deleted from the row indexing set of the in silico interactome.

Value

An incidence matrix. A new matrix with those columns indexed by unwantedComplex removed. It will also remove any protein which is now no longer contained in any protein complex.

Note

The entries of unwantedComplex must be found in the column names of ISI, otherwise, they will not be removed. This is especially important since there are many id's that index the same protein complex.

Author(s)

T. Chiang

Examples

```
#mips = getMipsInfo()
#go = getGOInfo()
#mipsM = createMipsMatrix(mips)
#goM = createGOMatrix(go)
#mips2go = runCompareComplex(mipsM, goM, byWhich = "ROW")
#merged = mergeBGMat(mipsM, goM, mips2go$toBeRm)
#ISI = unWantedComp(merged)
```

xtraGO *A character vector of hand selected GO protein complexes*

Description

Hand selected protein complexes that will be used to see if any subset should be incorporated into the ScISI

Usage

```
data(xtraGO)
```

Format

The format is: chr "xtraGO"

Examples

```
data(xtraGO)
```

xtraGONodes *A function to check manually curated GO nodes*

Description

This function takes any manually curated GO nodes and checks to see if those nodes belong should be incorporated to the ScISI

Usage

```
xtraGONodes(xtraGO, goM)
```

Arguments

xtraGO	A character vector of the GO nodes to be checked
goM	A bi-partite graph incidence matrix of the complexes selected by the getGOInfo function and put together by the createGOMatrix function.

Value

A bi-partite graph incidence matrix with whichever xtraGO nodes checked and deemed appropriate added to the goM matrix.

Author(s)

T. Chiang

yeastData-class *Class "yeastData"*

Description

A class representing an interactome of yeast. The object is in the form of a dataframe with three generic methods. The methods take two arguments: the first is the instance of the class yeastData; the second is the name of a protein complex with respect as given by the bi-partite graph incidence matrix.

Objects from the Class

Objects can be created by calls of the form `new("yeastData", reference = dataFrameISI)`.

Slots

reference The yeastData slot is merely a data frame for some in silico interactome.

Methods

ID `signature(object = "yeastData", name = "character")`: ...

Desc `signature(object = "yeastData", name = "character")`: ...

getURL `signature(object = "yeastData", name = "character")`: ...

Author(s)

Tony Chiang

Examples

```
#go = getGOInfo(wantAllComplexes = FALSE)
#goM = createGOMatrix(go)
#cNames = colnames(goM)
#goDF = createGODataFrame(go, goM)
#goOb = createYeastObj(goDF)
#ID(goOb, cNames[5])
#Desc(goOb, cNames[5])
#getUrl(goOb, cNames[5])
```

Index