PCpheno

October 5, 2010

`CoHyperGParams-class`
Class "CoHyperGParams"

Description

A parameter class for representing all parameters needed for running the `hyperGTest` method with multiprotein complexes.

Objects from the Class

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

Slots

`geneIds`**:** Object of class `"ANY"`: A vector of gene identifiers. Numeric and character vectors are probably the only things that make sense. These are the gene ids for the selected gene set.

`universeGeneIds`**:** Object of class `"ANY"`: A vector of gene ids in the same format as `geneIds` defining a subset of the gene ids on the chip that will be used as the universe for the hypergeometric calculation. If this is `NULL` or has length zero, then all gene ids on the chip will be used.

`annotation`**:** A string giving the name of the annotation data package for the chip used to generate the data.

`cateogrySubsetIds`**:** Object of class `"ANY"`: If the test method supports it, can be used to specify a subset of category ids to include in the test instead of all possible category ids.

`categoryName`**:** A string describing the category. Usually set automatically by subclasses. For example "ScISI".

`pvalueCutoff`**:** The p-value to use as a cutoff for significance for testing methods that require it. This value will also be passed on to the result instance and used for display and counting of significant results. The default is 0.01.

`testDirection`**:** A string indicating whether the test should be for overrepresentation (`"over"`) or underrepresentation (`"under"`).

Methods

hyperGTest `signature(p = "HyperGParams")`: Perform hypergeometric tests to assess over-representation of category ids in the gene set. See the documentation for the generic function for details. This method must be called with a proper subclass of `HyperGParams`.

`geneIds(r)`, `geneIds(r) <- value` Accessors for the gene identifiers that will be used as the selected gene list.

codeannotation(object) Accessor for annotation

`ontology(r)` Accessor for GO ontology.

`pvalueCutoff(r)`, `pvalueCutoff(r) <- value` Accessor for the p-value cutoff. When setting, `value` should be a numeric value between zero and one.

`testDirection` Accessor for the test direction. When setting, `value` must be either "over" or "under".

`universeGeneIds(r)` accessor for vector of gene identifiers.

`isConditional(r)` Returns `TRUE` if the instance has its conditional flag set

Author(s)

S. Falcon and N. LeMeur

See Also

`HyperGResult-class` `CoHyperGResult-class` `hyperGTest`

CoHyperGResult-class

Class "CoHyperGResult"

Description

This class represents the results of a test for over-representation of genes in a selected gene set based among protein complexes upon the Hypergeometric distribution.

Objects from the Class

Objects is created by calls to the funtion hyperGTest.

Slots

`pvalues`: `"numeric"` vector: the ordered p-values for each category term tested.

`oddsRatios`: Object of class `"numeric"` Odds ratio for each category term tested

`expectedCounts`: Object of class `"numeric"` The expected number of genes for each gene term tested

`geneCounts`: `"integer"` vector: for each category term tested, the number of genes from the gene set that are annotated at the term.

`universeCounts`: `"integer"` vector: for each category term tested, the number of genes from the gene universe that are annotated at the term.

`catToGeneId`: Object of class `"list"`. The names of the list are category IDs. Each element is a vector of gene IDs annotated at the given category ID and in the specified gene universe.

Extends

Class "HyperGResultBase", directly.

Methods

geneCounts signature(r = "CoHyperGResult"): return an "numeric" vector: for each category term tested, the number of genes from the gene set that are annotated at the term.

pvalues signature(r = "HyperGResult"): return a "numeric" vector: the ordered p-values for each category term tested.

oddsRatios signature(r = "HyperGResult"): return a "numeric" vector: the odds ratio for each category term tested.

expectedCounts signature(r = "HyperGResult"): return a "numeric" vector: the expected number of genes for each GO term tested.

universeCounts signature(r = "HyperGResult"): return an "numeric" vector: for each category term tested, the number of genes from the gene universe that are annotated at the term.

geneIdUniverse signature(r = "CoHyperGResult"): return a list named by the protein Complexes. Each element of the list is a vector of gene identifiers (from the gene universe) annotated at the corresponding protein complex.

summary signature(r = "CoHyperGResult"): Returns a data.frame summarizing the test result. Optional arguments pvalue and categorySize allow specification of minimum p-value and cateogyrSize, respectively. Optional argument htmlLinks is a logical value indicating whether to add HTML links (useful in conjunction with xtables print method with type set to "html").

Author(s)

S. Falcon and N. LeMeur

See Also

HyperGResultBase-class

Examples

```
data(DudleyPhenoM)
data(ScISIC)

## Select genes sensitive to paraquat
DudleyPhenoL <- apply(DudleyPhenoM,2,function(x) names(which(x==1)))
paraquat <- DudleyPhenoL[["Paraq"]]

## Apply a hypergeometric test
params <- new("CoHyperGParams",
              geneIds=paraquat,
              universeGeneIds=rownames(ScISIC),
              annotation="org.Sc.sgd.db",
              categoryName="ScISIC",
              pvalueCutoff=0.01,
              testDirection="over")
```

```
paraquat.complex <- hyperGTest(params)

## access the p-values
pvalues(paraquat.complex)[1:5]

## Display a summary of the results
summary(paraquat.complex)[,1:4]
```

`DudleyPheno` *List of fitness defect score generated from Dudley et al 2005*

Description

Dudley et al (2005) create a collection of gene-deletion mutants to determine genes that contribute to a particular phenotype in specific environmental conditions. This list is generated from a fitness analysis under 21 different experimental conditions.

Usage

```
data(DudleyGenelist)
data(DudleyPhenoFull)
data(DudleyPhenoM)
data(DudleySign)
```

Format

`DudleyGenelist` is a character vector of length 814 genes. `DudleyPhenoFull` is a dataframe of 814 genes by 23 elements. The column contains the yeast gene name. The 22 remaining columns are the experimental conditions (see details). The values obtained in the different condition are the fitness defect scores for the 814 genes sensitive to the experimental condition, as defined by Dudley et al (2005). `DudleyPhenoM` is a 814 by 22 incidence matrix with rownames corresponding to the genes names and columns to an experimental condition. This matrix contains a 1 in the (i,j) position if the i-th gene is sensitive to the experimental condition of the j-th column; it contains a 0 otherwise.

Yname Yeast systematic gene name

benomyl 15ug/ml benomyl,microtubule function

CaCl2 0.7M calcium chloride, divalent cation

CAD 55uM Cadmium, heavy metal

Caff 2mg/ml Caffeine

cyclohex 0.18ug/ml cycloheximide, protein synthesis

DTT unknown

EtOH YPD + 6% Ethanol

FeLim iron limited, nutrient limited condition

HU 11.4mg/ml Hydroxyurea, DNA replication and repair

HygroB 50ug/ml hygromycin B, aminoglycosides

lowPO4 Low phosphate, nutrient limited condition

MPA 20ug/ml mycophenolic acid, transcriptional elongation

NaCl 1.2M sodium chloride, general stress condition

Paraq 1mM paraquat, oxidative stress

pH3 Low pH, general stress condition

rap 0.1ug/ml rapamycin, protein synthesis

Sorb 1.2M sorbitol, general stress condition

UV 100J/m2 ultra-violet, DNA replication and repair

YPGal 2% galactose, carbon source

YPGly 3% glycerol, carbon source

YPLac 2% lactate, carbon source

YPRaff 2% raffinose, carbon source

`DudleySign` is a list of dataframe that summaryzes in which complexes the gene related to the phenotype are found, the size of the complexes and the associated p-value. This is the result of applying a Hypergeometric test (see `CoHyperGParams-class` for more details) and the `complexStatus` function.

`Dudleyresult` is a data.frame that summaryzes the number of sensitive genes per condition, how many of those genes are present in the ScISI interactome and the associated p-value. This is the result of applying a Hypergeometric test (see `CoHyperGParams-class` for more details) and the `complexStatus` function.

Author(s)

N. LeMeur

Source

Dudley et al (2005), supplementary information: http://arep.med.harvard.edu/pheno/default.htm

References

Aim\'ee Marie Dudley, Daniel Maarten Janse, Amos Tanay, Ron Shamir and George McDonald Church. (2005).A global view of pleiotropy and phenotypically derived gene function in yeast. *Molecular Systems Biology* doi:10.1038/msb4100004

Examples

```
data(DudleyPhenoFull)
data(DudleyPhenoM)
```

GiaeverPheno *List of fitness defect score generated from Giaever et al 2002*

Description

Giaever et al (2002) create a collection of gene-deletion mutants to determine genes that contribute to a particular phenotype in specific environmental conditions. This list is generated from a fitness analysis under six different experimental conditions.

Usage

```
data(GiaeverPheno)
data(GiaeverGene)
data(GiaeverExpCdt)
```

Format

GiaeverPheno is a list with 31 elements. The name of each element is a experimental condition (see details). The value of each element are the fitness defect scores for the genes sensitive to the experimental condition, as defined by Giaever et al (2002).

GiaeverGene Vector of the systematic gene names of the 5898 tested genes. Note that some updates have been made for the list to be consistent with *Saccharomyces Genome Database*.

GiaeverExpCdt is a 3 columns dataframe with fileID from which the data were extracted, the generation time (growth time) and the condition (media).

gen. generations

rep. replicate

ypg5a,ypg5b yeast/peptone/galactose 5 gen. rep. a and b ==> carbone source

ypg15a ypg15b yeast/peptone/galactose 15 gen. rep. a and b ==> carbone source

sorbitol5a sorbitol5b 1.5M Sorbitol 5 gen. rep. a and b ==> sugar, osmotic stress

sorbitol20a sorbitol15b 1.5M Sorbitol 20 and 15 gen. rep. a and b respectively==> sugar, osmotic stress

NaCl5a NaCl5b 1M NaCl 5 gen. rep. a and b ==> salt, osmotic stress

NaCl15a NaCl15b 1M NaCl 15 gen. rep. a and b ==> salt, osmotic stress

lysM5a lysM5b lysine minus 5 gen. rep. a and b ==> lack of required AA

thM5a threonine minus 5 gen. rep. a ==> lack of required AA

trpM5a trpM5b tritophanee minus 5 gen. rep. a ==> lack of required AA

minimalPlus5a minimalPlus5b minimal + histidine/leuvine/uracile 5 gen. rep. a and b

minimalPlus15a minimalPlus15b minimal + histidine/leuvine/uracile 15 gen. rep. a and b

minimalC5a minimalC5b minimal complete 5 gen. rep. a and b

nystatin5a nystatin5b Nystatin 5 gen. rep. a and b ==> antifungal drug

nystatin15a nystatin15b Nystatin 5 gen. rep. a and b ==> antifungal drug

pH8g5a pH8g5b pH 8 5 gen. rep. a and b ==> alkali stress

pH8g15a pH8g20b pH 8 15 and 20 gen. rep. a and b respectively ==> alkali stress

Note: in their study they confound the 15 and 20 generations.

`Giaeverresult` is a data.frame that summaryzes the number of sensitive genes per condition, how many of those genes are present in the ScISI interactome and the associated p-value. This is the result of applying a Hypergeometric test (see `CoHyperGParams-class` for more details) and the `complexStatus` function.

Author(s)

N. LeMeur

Source

Giaever et al (2002), supplementary information: http://genomics.lbl.gov/YeastFitnessData/websitefiles/cel_index.html Saccharomyces Genome Database (last update 03/17/06): http://www.yeastgenome.org/

References

Giaever G et al (2002) Functional profiling of the Saccharomyces cerevisiae genome. Nature. 418(6896):387-91. PMID: 12140549

Examples

```
data(GiaeverPheno)
data(GiaeverExpCdt)
data(GiaeverGene)
```

HI *Data from Deutshbauer et al. (2005)*

Description

Mechanisms of Haploinsufficiency revealed by Genome-Wide Profiling in Yeast (Deutshbauer et al.,2005)

Usage

```
data(HI)
```

Details

HI stands for haploinsufficient. The dataframe is composed of:

orf Yeast ORF, systematic name

gene Yeast common gene name of the corresponding ORF

go GO terms

Source

http://www.sciencemag.org/cgi/data/303/5659/808/DC1/1

References

Deutschbauer AM, Jaramillo DF, Proctor M, Kumm J, Hillenmeyer ME, Davis RW, Nislow C, Giaever G. (2005) Mechanisms of haploinsufficiency revealed by genome-wide profiling in yeast. Genetics. 169(4):1915-25.

Examples

```
data(HI)
```

`KEGG2SCISI` *Mapping between KEGG and ScISI*

Description

Count the number of genes shared between a KEGG pathway and a protein complex from the ScISI interactome.

Usage

```
KEGG2SCISI(pw, pc, pcMat, pwMat)
```

Arguments

`pw`	list of pathway names
`pc`	list of complex names
`pwMat`	pathway incidence matrix
`pcMat`	complex incidence matrix

Value

matrix

Author(s)

N. LeMeur

See Also

ScISI KEGG

Examples

```
data(ScISIC)
## Mapping from Yeast genes to KEGG pathways.
KeggMat <- PWAmat("org.Sc.sgd")
KEGG2SCISI(pw = colnames(KeggMat)[1:5], pc = colnames(ScISIC)[1:5], pwMat =
KeggMat, pcMat =ScISIC)
```

KastenmayerRaw *Data from Kastenmayer et al. 2006*

Description

Kastenmayer et al. (2006) undertook the first functional studies of small open reading frames (sORFs) in any system, using the model eukaryote Saccharomyces cerevisiae. Phenotypic analyses of the new gene-deletion strains identified 22 sORFs required for haploid growth, growth at high temperature, growth in the presence of a non-fermentable carbon source, or growth in the presence of DNA damage and replication-arrest agents.

Usage

```
data(KastenmayerRaw)
```

Format

`Kastenmayer` is a 5 columns dataframe.

SYSTEMATIC Systematic name of the sORF.

COMMUN Commun name of the sORF.

Length Length of the small ORF sequence in number of amino acids.

Evidence Experimental source of the data

refHomology Bibliographical evidence of reported homology.

Kocollection Bibliographical evidence of reported homology.

ESSENTIAL Indicates if the sORF knockout is essential. A blank in this column indicates that the knockout is not-essential, if available.

GFPTAP "GFP" or "TAP" signifies that sORF was detected by the indicated technique. "both" indicates that sORF was detected both as a TAP-tagged and GFP-tagged protein. "None" indicates that sORF was not detected by either method. Empty field indicates that sORF was not tested

UPTAG Sequence of the upstream primer.

DOWNTAG Sequence of the downstream primer.

Author(s)

N. LeMeur

Source

Kastenmayer et al. (2006), supplementary information: http://www.genomeresearch.org or ftp://genome-ftp.stanford.edu/pub/yeast/systematic_results/phenotypes

References

Kastenmayer JP, Ni L, Chu A, Kitchen LE, Au WC, Yang H, Carter CD, Wheeler D, Davis RW, Boeke JD, Snyder MA, Basrai MA. (2006) Functional genomics of genes with small open reading frames (sORFs) in S. cerevisiae. Genome Res. 16(3):365-73. PMID: 16510898

Examples

```
data(KastenmayerRaw)
str(KastenmayerRaw)
```

LesageRaw *Data from Lesage et al. 2005*

Description

Lesage et al. (2005) assembled a network of 316 interactions among 163 genes using deletion mutants in CHS1, CHS3, CHS4, CHS5, CHS6, CHS7 and BNI4 in a synthetic genetic array analysis.

Usage

```
data(LesageRaw)
```

Format

LesageRaw is a 5 column dataframe.

SYSTEMATIC Systematic gene names. NOTE: All mutants are isogenic to BY4741 (MATa his3\u0394 leu2\u0394 met15\u0394 ura3\u0394) except anp1\u0394 and mnn9\u0394 that are isogenic to BY4742 (MAT\u03b1 his3\u0394 leu2\u0394 lys2\u0394 ura3\u0394).

COMMUN Commun gene names.

CFW Mutants showing increased, decreased or wild type sensitivity to Calcofluor white are scored s, r, or wt, respectively.

ChitinLevel Chitin level (nmole GlcNAc/mg dry weight). Values are an average of at least three independent determinations. Values statistically higher and lower than wild type ($p < 0.01$) are highlighted in red and green, respectively.

ChitinLevel.SD Standard deviation of the average of at least three independent determinations of Chitin level.

Author(s)

N. LeMeur

Source

Lesage et al. (2005), supplementary information: http://www.biomedcentral.com/1471-2156/6/8/suppl/S2 or ftp://genome-ftp.stanford.edu/pub/yeast/systematic_results/phenotypes

References

Lesage G, Shapiro J, Specht CA, Sdicu AM, Menard P, Hussein S, Tong AH, Boone C, Bussey H. (2005) An interactional network of genes involved in chitin synthesis in Saccharomyces cerevisiae.BMC Genet.6(1):8. PMID: 15715908

Examples

```
data(LesageRaw)
str(LesageRaw)
```

`OsterbergRaw` *Data from Osterberg at al. 2006*

Description

Osterberg at al. (2006) report growth phenotypes in yeast for a strain collection over-expression ~600 C-terminal tagged integral membrane proteins growth both under normal and three different stress conditions.

Usage

```
data(OsterbergRaw)
```

Format

`OsterbergRaw` is a 17 columns dataframe.

SYSTEMATIC Systematic gene names of the studied membrane protein

COMMUN Commun gene names of the studied membrane protein

TMHMM.C The topology predicted by TMHMM (TransMembrane prediction using Hidden Markov Models) using the experimentally assigned C-terminal location for the protein as a constraint. The topology is represented in the format Location of N-terminus TMhelices Location of C-terminus (i and o stand for in and out respectively)

WesternBlot Protein expression levels (arbitrary units), estimated from the band intensity and normalized to the internal standard on each Western blot.

Bands Proteins detected as two distinct bands with different molecular mass on the Western blot analysis. category 1 indicates that both bands were insensitive to Endo H digestion, 2 indicates the higher molecular mass was shifted down upon Endo H digestion on SDS/PAGE and one band was predominant compared to the other, and 3 indicates that a higher molecular mass band shifted down upon Endo H digestion on SDS/PAGE and both bands were equal intensity on Western blot

Toxicity Toxicity index from Spoko et al. (2006). The index varies between 1 and 5, where 1 means the strain is dead, and 5 indicates no difference in growth rate compared with the wild type strain.

sign.norm Over-expression strains that show a significant (P<0.001) growth rate phenotype (LSCrate) in synthetic defined medium conditions (Warringer et al., 2003). An average of two replicates is given. Strains that do not show a significant difference in doubling time compared with the wild-type strain are indicated by 0.

all.norm Phenotypes (significant or not) of over-expression strains (LSCrate) in synthetic defined medium conditions (Warringer et al., 2003). An average of two replicates is given.

sign.NaCl Over-expression strains that show a significant (P<0.001) growth rate phenotype (LPIrate) in NaCl. An average of two replicates is given. Strains that do not show a significant difference in doubling time compared with the wild-type strain under NACL stress are indicated by 0.

all.NaCl.LSC Phenotypes (significant or not) of over-expression strains (LSCrate) in NaCl (Warringer et al., 2003). An average of two replicates is given.

all.NaCl.LPI Phenotypes (significant or not) of over-expression strains (LPIrate) in NaCl (Warringer et al., 2003). An average of two replicates is given.

sign.caff Over-expression strains that show a significant (P<0.001) growth rate phenotype (LPIrate) in caffeine. An average of two replicates is given. Strains that do not show a significant difference in doubling time compared with the wild-type strain under caffeine stress are indicated by 0.

all.caff.LSC Phenotypes (significant or not) of over-expression strains (LSCrate) in caffeine (Warringer et al., 2003). An average of two replicates is given

all.caff.LPI Phenotypes (significant or not) of over-expression strains (LPIrate) in caffeine (Warringer et al., 2003). An average of two replicates is given.

sign.paraq. Over-expression strains that show a significant (P<0.001) growth rate phenotype (LPIrate) in paraquat. An average of two replicates is given. Strains that do not show a significant difference in doubling time compared with the wild-type strain under paraquat stress are indicated by 0.

all.paraq.LSC Phenotypes (significant or not) of overexpression strains (LSCrate) in paraquat (Warringer et al., 2003). An average of two replicates is given.

all.paraq.LPI Phenotypes (significant or not) of overexpression strains (LPIrate) in paraquat (Warringer et al., 2003). An average of two replicates is given.

Author(s)

N. LeMeur

Source

Osterberg et al (2006), supplementary information: http://www.pnas.org/content/vol0/issue2006/images/data/0604078103/DC1/04078Table1.xls or ftp://genome-ftp.stanford.edu/pub/yeast/systematic_results/phenotypes

References

Osterberg M, Kim H, Warringer J, Melen K, Blomberg A, von Heijne G. (2006) Phenotypic effects of membrane protein overexpression in Saccharomyces cerevisiae. PNAS. 103(30):11148-53. PMID: 16847257

Examples

```
data(OsterbergRaw)
str(OsterbergRaw)
```

`PCpheno-package` *Linkage between Protein Complexes, Pathways, and Phenotypes*

Description

Tools to integrate, annotate and search for associations between phenotypes, protein complexes, and pathways.

Details

Package:	PCpheno
Type:	Package
Version:	1.3.1
Date:	2006-03-09
License:	The Artistic License, Version 2.0

Author(s)

N. LeMeur and R. Gentleman

Maintainer: N. LeMeur <nlemeur@fhcrc.org>

References

Giaever G, et al.(2002) Functional profiling of the Saccharomyces cerevisiae genome. Nature. 418(6896):387-91. PMID: 12140549

Deutschbauer AM, Jaramillo DF, Proctor M, Kumm J, Hillenmeyer ME, Davis RW, Nislow C, Giaever G. (2005) Mechanisms of haploinsufficiency revealed by genome-wide profiling in yeast. Genetics. 169(4):1915-25.

Byrne KP, Wolfe KH. (2005) The Yeast Gene Order Browser: combining curated homology and syntenic context reveals gene fate in polyploid species. Genome Res. 15(10):1456-61. PMID: 16169922

See Also

ScISI,SLGI

`SGDphenoL` *Saccharomyces Genome Database list of phenotypic data*

Description

Saccharomyces Genome Database list of phenotypes and associated genes from several published experiments (last update 2006).

Usage

```
data(SGDphenoL)
```

Format

`SGDphenoL` is a list of phenotypes. Under each phenotype is listed the genes that potentially induce that phenotype. A binary matrix can be built from that list using the `list2Matrix` function from the `Rintact` package.

Author(s)

N. LeMeur

Source

SGD, supplementary information: http://www.yeastgenome.org/

Examples

```
data(SGDphenoL)
```

YEASTOHNOLOG *List of ohnolog gene pairs from Byrne, K.P and Wolfe, K.H (2005)*

Description

List of 551 paralogous *Saccharomyces cerevisiae* gene pairs formed by Whole Genome Duplication (WGD) or ohnolog pairs.

Usage

```
data(YEASTOHNOLOG)
```

Format

YEASTOHNOLOG is a dataframe of 551 paired genes. The first two columns are the ohnolog gene pairs (systematic gene names). The third column is an index (numeric) of the rate of sequence evolution. The last two columns define the chromosome location.

Details

Gene1 Gene2 Systematic gene names of the ohnolog pairs

Ka Coefficient that represents the extent of non-synonymous sequence divergence between each ohnolog pairs (Yang and Nielsen, 2000). The highest is the coefficient the fastest the 2 elements of a pair have diverged.

ChrG1 ChrG2 Chromosome location of the each element of a pair. Note that repeat of the same chromosome locations shared by a set of pairs define a block of duplication.

Author(s)

N. LeMeur

Source

Byrne,KP and Wolfe KH (2005), Table2 of supplementary information and Scerevisiae_genome.tab file, chromosome location, from YGOB http://wolfe.gen.tcd.ie/ygob/ (last update 03/20/06)

References

Byrne KP, Wolfe KH. (2005) The Yeast Gene Order Browser: combining curated homology and syntenic context reveals gene fate in polyploid species. Genome Res. 15(10):1456-61. PMID: 16169922

Examples

```
data(YEASTOHNOLOG)
str(YEASTOHNOLOG)
```

`buildFDMat` *Build fitness defect contingency matrix*

Description

Function to build a fitness defect contingency matrix where rows correspond to tested genes and columns to experimental conditions.

Usage

```
buildFDMat(data,genenames,condition)
```

Arguments

`data`	List of 'significant' fitness defect scores and the associated genes at different experimental conditions.
`condition`	Character vector of the different experimental conditions tested
`genenames`	Character vector of all the tested genes for fitness defect.

Value

Contingency matrix of genes that present significant fitness defect in different experimental conditions.

Author(s)

N. LeMeur

Examples

```
data(GiaeverPheno)
data(GiaeverExpCdt)
data(GiaeverGene)
fitnessData <- getFDgene(GiaeverPheno,condition=GiaeverExpCdt,cutoff=c(20,100,100),mode=
GiaeverPhenoM <- buildFDMat(data=fitnessData,genenames=GiaeverGene,condition=GiaeverExpC
```

categoryToEntrezBuilder *Return a list mapping multi-protein complexes IDs to YEAST ids*

Description

Return a list mapping multi-protein complexes (category) IDs to the YEAST ids annotated at the category id.

Usage

```
## S4 method for signature 'CoHyperGParams':
categoryToEntrezBuilder(p)
```

Arguments

p	A subclass of HyperGParams-class

Details

End users **should not** call this directly. This method gets called from hyperGTest. To add support for a new category, a new method for this generic must be defined. Its signature should match a subclass of HyperGParams-class appropriate for the new category.

Value

A list mapping category IDs to YEAST identifiers.

Author(s)

S. Falcon and N. LeMeur

See Also

hyperGTest CoHyperGParams-class

Examples

```
data(ScISIC)
data(essglist)
essential <- names(essglist)

params <- new("CoHyperGParams",
              geneIds=essential,
              universeGeneIds=rownames(ScISIC),
              annotation="org.Sc.sgd.db",
              categoryName="ScISIC",
              pvalueCutoff=0.01,
              testDirection="over")

categoryToEntrezBuilder(params)[1:2]
```

`complexStatus` *Complex Status*

Description

Categorize the complex whether or not a complex is composed of a significant number of genes involved in a particular phenotype than expected by chance.

Usage

```
complexStatus(data, phenotype, interactome, threshold=0.05)
```

Arguments

`data`	Output from CoHyperG test
`phenotype`	List of gene names inducing an observed phenotype, e.g., list of essential gene names (see package *SLGI*)
`interactome`	A binary matrix composed of genes (rows) and biological complexes (columns) (see package *ScISI*)
`threshold`	pvalue threshold (default 0.05)

Details

We form four distinct categories from A to D to characterize how a complex might be involved in a particular phenotype (according to the number of genes it contains and that are involved in a particular phenotype - see also hyperGTest function)

Value

The returned value is a list with components:

`A`	"interesting" complexes, complexes with a significant number of interesting genes, i.e., genes that participate to a particular phenotype (at a given p-values threshold)
`B`	complexes with a NON significant number of interesting genes BUT that SHARE genes with complexes from the A status
`C`	complexes with a NON significant number of interesting genes AND that DON'T SHARE interesting genes with complexes from cat A
`D`	complexes WITHOUT interesting genes, i.e. the one involved in the studied phenotype

Author(s)

N. LeMeur

Examples

```
data(ScISI)
data(essglist)
essential <- names(essglist)

CoparamsESS <- new("CoHyperGParams",
                   geneIds=essential,
                   universeGeneIds=rownames(ScISI),
                   annotation="org.Sc.sgd.db",
                   categoryName="ScISI",
                   pvalueCutoff=0.01,
                   testDirection="over")

sign<- hyperGTest(CoparamsESS)
test05 <-complexStatus(data=sign, phenotype=essential,
interactome=ScISI, threshold=0.05)
```

---

deResult-class *A class for representing the result of a densityEstimate test.*

Description

A class for representing the result of a `densityEstimate` test.

Slots

`Size`**:** Object of class `"numeric"` representing the size of the cellular organizational unit tested

`Observed`**:** Return a `"numeric"` vector: the observed number of interactions between genes inducing a specific phenotype and each cellular organizational units

`Expected`**:** Return a matrix: the expected number of interactions between genes inducing a specific phenotype and each cellular organizational units

Extends

Class `"testResult"`, directly.

Methods

`plot` Graphical representation of the test result

Author(s)

N. LeMeur

See Also

`testResult`,`gtResult`,`densityEstimate`, `plot`

Examples

```
## apply a densityEstimate test
data( DudleyPhenoM)
data(ScISIC)

DudleyPhenoL <- apply(DudleyPhenoM,2,function(x) names(which(x==1)))
pH3 <- DudleyPhenoL[["pH3"]]

perm <- 20
pH3Density <- densityEstimate(genename=pH3, interactome=ScISIC, perm=perm)

## access results
pH3Density@Observed[1:5]

## use of the plot method
plot(pH3Density)
```

---

densityEstimate *Observed versus Expected Ratios*

Description

Function to calculate the ratio of genes that characterize a phenotype (observed) among the genes that characterize a biological complex versus the ratio of a set of randomly sampled genes (expected) among the genes that characterize a biological complex.

Usage

```
densityEstimate(genename,interactome,perm)
```

Arguments

genename	Character vector of the gene names that characterize a specific phenotype.
interactome	Contingency matrix of genes (rows) and biological complexes (columns) (see package *ScISI*)
perm	Numeric vector indicating the number of simulations to run to compute the expected ratios.

Value

List of observed and simulated ratios.

Author(s)

N. LeMeur

Examples

```
data(ScISI)
data(essglist)
essential <- names(essglist)
ScISI <- as.matrix(ScISI)
 ratio<- densityEstimate(genename=essential,interactome=ScISI,perm=50)
```

getDescr *Get formatted annotation data*

Description

Function to retrieve the annotation of multi-protein complexes or pathways via GO, MIPS or KEGG.

Usage

```
getDescr(x, database="GO.db")
```

Arguments

x	Vector of multi-protein complexes or pathways IDs to be described
database	Source of annotation. The database currently available are MIPS, GO.db and KEGG.db

Author(s)

N. LeMeur

Examples

```
xx <- getDescr(c("MIPS-220","MIPS-260.20","04111"),c("MIPS","KEGG.db"))
```

getFDgene *Get fitness defect genes*

Description

Function to select genes that present a significant growth defect according to the condition(media) or generation time.

Usage

```
getFDgene(data,condition,cutoff,mode="generation",subset)
```

Arguments

data	List of fitness defect scores for genes tested at different experimental conditions.
condition	Dataframe of experimental conditions
cutoff	Numerical vector of length one or more, defining the threshold of 'significance' for the fitness defect score
mode	Character string defining the base of the selection either 'condition' (media) or 'generation' time, Default=generation.
subset	Numerical vector or list to which apply the different cutoffs.

Value

Reduced list of gene fitness scores per experimental condition according to the experimental condition or the generation time.

Author(s)

N. LeMeur

References

Giaever G. et al. (2002) Functional profiling of the Saccharomyces cerevisiae genome. Nature. 418(6896):387-91. PMID: 12140549

Examples

```
 data(GiaeverPheno)
 data(GiaeverExpCdt)
 ##Select all the genes, in the different experimental conditions, that present a fitness
 fitnessGen <- getFDgene(GiaeverPheno,condition=GiaeverExpCdt,cutoff=c(20,100,100),mode="
 ##Select all the genes, that present a fitness score above 15 and 100
 ##in the condition set A and B respectively, independently of the generation time
fitnessCondt <- getFDgene(GiaeverPheno,condition=GiaeverExpCdt,cutoff=c(100,15),mode="con
```

---

graphTheory *Graph theory to test associations between two or more relationships*

Description

Graph theory approach associated with a permutation test to evaluate whether the number of associations is unexpectedly large.

Usage

```
graphTheory(genename, interactome, perm)
```

Arguments

genename	A vector a gene names that are associated with a particular phenotype
interactome	A binary matrix composed of genes (rows) and biological complexes (columns) (see package *ScISI*)
perm	Numeric, number of permutation run

Details

We form two distinct graphs where the set of nodes are the proteins (genes) in the organism. In one graph we create edges between genes if the two genes are members of one, or more, protein complexes. In the second graph we create an edge between all genes that are associated to a particular phenotype. We then construct a third graph on the same node set, but where there is an edge in this graph only if there is an edge in both of the first to graphs. We count the number of edges in the third and test by permutation whether the number of edges is unexpectedly large.

Value

The returned value is a list with components:

`edgeCount`	Number of associations observed between the genes that are linked to a particular phenotype and the given interactome.
`edgeSimul`	Number of associations if the genes that are linked to a particular phenotype are randomly distributed across the given interactome
`p.value`	Returned p.value

Author(s)

R. Gentleman and N. LeMeur

References

Balasubramanian, R., LaFramboise, T., Scholtens, D., Gentleman, R. (2004) A graph-theoretic approach to testing associations between disparate sources of functional genomics data.Bioinformatics,20(18),3353-3362.

Examples

```
data(ScISI)
data(essglist)
ans <- graphTheory(names(essglist), ScISI, perm=3)
```

---

gtResult-class *A class for representing the result of a graphTheory test.*

Description

A class for representing the result of a `graphTheory` test.

Slots

`Pvalue`**:** Object of class `"numeric"`

`Observed`**:** Return a `"numeric"` vector: the observed number of interactions between genes inducing a specific phenotype and each cellular organizational units

`Expected`**:** Return a matrix: the expected number of interactions between genes inducing a specific phenotype and each cellular organizational units

Extends

Class `"testResult"`, directly.

Methods

`plot` Graphical representation of the test result

Author(s)

N. LeMeur

See Also

testResult,deResult,plot

Examples

```
## apply a densityEstimate test
data(DudleyPhenoM)
data(ScISIC)

DudleyPhenoL <- apply(DudleyPhenoM,2,function(x) names(which(x==1)))
NaCl <- DudleyPhenoL[["NaCl"]]

perm <- 20
NaClGraph <- graphTheory(genename=NaCl, interactome=ScISIC,
perm=perm)


## access results
slotNames(NaClGraph)
NaClGraph@Pvalue[1:5]

## use of the plot method
plot(NaClGraph)
```

---

overlap *Count the number of proteins shared by protein complexes*

Description

Count the number of proteins shared by protein complexes

Usage

```
overlap(interactome)
```

Arguments

interactome	Binary matrix composed of genes (rows) and biological complexes (columns) (see package *ScISI*)

Value

The returned value is a data frame with components:

C1	Name of the first biological complex
C2	Name of the second biological complex
nbSharedProt	Number of proteins in common

Author(s)

N. LeMeur

See Also

ScISI

Examples

```
xx = cbind("a"=c(0,1,1,1),"b"=c(1,1,0,1))
overlap(xx)
```

`plot` *Graphical method to represent the result of the density or graph test.*

Description

a plot method for `deResult` and `gtResult` objects.

Usage

```
## S4 method for signature 'deResult':
plot(x,...)
## S4 method for signature 'gtResult':
plot(x,...)
```

Arguments

`x`	the `deResult` or `gtResult` object to plot.
`...`	general commands to be sent to plot.

Details

The plot generated from a `deResult` object is a set of density plots.

The plot generated from a `gtResult` object is a histogram.

Author(s)

N. LeMeur

See Also

ScISI

Examples

```
data(ScISI)
data(essglist)
essential <- names(essglist)
ScISI <- as.matrix(ScISI)
ratio<- densityEstimate(genename=essential,interactome=ScISI,perm=50)
plot(ratio)
```

`ppiInteract` *Test the association between AP-MS data and phenotype*

Description

Test the association between AP-MS data and phenotype data via a graph and permutation model.

Usage

```
ppiInteract(genename, expGraph, bait, prey, perm=10)
```

Arguments

`genename`	Genes associated to a phenotype
`expGraph`	A graphNEL object (a direct graph instance of class`graph`). The nodes are the union of viable baits (VB) and viable prey (VP) of the experiment (see package *ScISI*)
`bait`	Proteins which was sampled as a bait in the binary relationship
`prey`	Proteins which was sampled as a prey in the binary relationship
`perm`	Number of permutation

Value

The returned value is a list:

`Observed`	Observed values
`Expected`	Expected values after each permutation

Author(s)

R. Gentleman and N. LeMeur

See Also

ScISI

Examples

```
data(ScISI)
data(essglist)
s1 <- ppiInteract(names(essglist), Gavin2002BPGraph, viableBaits[[8]],
       viablePrey[[8]], perm=10)
```

reduceM *Resize a matrix*

Description

Resize a matrix to the number of rows commun to a vector.

Usage

```
reduceM(x, mat, threshold=0)
```

Arguments

x	Character or numeric vector.
mat	Matrix sharing rownames with the supplied vextor x.
threshold	Threshold upon column. Only the columns with a colSums above the threshold are kept.

Value

Resized matrix.

Author(s)

N. LeMeur

Examples

```
mat <- matrix(c(1:25), nrow = 5, ncol = 5, dimnames = list(c(LETTERS[1:5]), c(1:5)))
xx <- LETTERS[c(2, 4, 5)]
reduceM(xx, mat)
```

testResult-class *A virtual class for representing the result of a test.*

Description

The testResult class is the virtual base class for all result objects of the densityEstimate and graphTheory tests proposed in PCpheno.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

Observed**:** Return a "numeric" vector: the observed number of genes or interactions within each cellular organizational units

Expected**:** Return a numeric or a matrix: the expected number of genes or interactions within each cellular organizational units

Methods

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

Author(s)

N. LeMeur

See Also

gtResult,deResult

---

truncName *Truncate character strings*

Description

Truncate character strings

Usage

```
truncName(x,n)
```

Arguments

x	Character string
n	Maximum length (in characters)

Value

Character string

Author(s)

N. LeMeur

Examples

```
xx <- "Anticonstitutionnelement is a family name"
truncName(xx,5)
```

Index