ppiStats

April 20, 2011

`assessSymmetry`	*A function that calcualtes some directed degree statistics on graphs using a binomial error model*

Description

This function takes in a bait to prey protein-protein interaction graph (undirected) and calculates the reciprocated degree, the unreciprocated in and out degrees. Using this information and binomial error model, it asses the p-value for the in and out degree of each protein. Lastly, it plots the countour curves for these p-values

Usage

```
assessSymmetry(bpMat, bpGraph = FALSE,
  prob=0.5, pLevels = 1e-4)
```

Arguments

`bpMat`	Either a bait to prey directed graphNEL or its corresponding adjacency matrix.
`bpGraph`	A logical. If TRUE, than bpMat is passed in by the user as a graphNEL.
`pLevels`	A numeric vector. It gives the levels to calculate the countours of the function in p in the (n-in, n-out)-plane
`prob`	A numeric. The bias of the coin used in the function pbinom call.

Value

A list:

`deg`	A 3xn matrix. The rows are indexed by each protein. Column one gives the number of reciprocated edges; column two gives the number of unreciprocated out edges; colunm three gives the number of unreciprocated in-edges
`p`	The p-value for each protein with experimental in and out degrees
`countours`	The countours as a function of p

Author(s)

W Huber

Examples

```
library(ppiData)
assessSymmetry(Ito2001BPGraph, bpGraph=TRUE)
```

`bpMatrix` *This function generates the Bait-Prey Matrix for an protein-protein interaction (ppi) experiment*

Description

This function takes in experimental ppi data and generates the bait to prey adjacency matrix.

Usage

```
bpMatrix(y2h, symMat = TRUE, homodimer = FALSE, baitAsPrey = FALSE,
      unWeighted=TRUE, onlyRecip=FALSE, baitsOnly=FALSE)
```

Arguments

`y2h`	A named list of character vectors. The names of the list correspond to the baits used in the experimental technology. The entries of the character vectors are those proteins found as prey or a character(0) if the bait did not find any prey.
`symMat`	A logical, if TRUE, the matrix will be square with all the proteins documented in the experiment indexing both the row and column; if FALSE, only the baits index the rows, preys the columns.
`homodimer`	A logical. If TRUE, the matrix will record the presence of homodimers; if FALSE, all homodimers data will be deleted.
`baitAsPrey`	A logical; if TRUE, the columns will be indexed by both the bait and prey population while the rows will remain indexed by the baits exclusively.
`unWeighted`	A logical. If TRUE, the entries of the adjacency will be binary (0,1) which records the presence of interactions or not. If FALSE, the entries of the matrix will be a natrual number to record the multiplicity of the interaction found by the experiment.
`onlyRecip`	A logical. If TRUE, the adjacency matrix will be restricted to only those interactions which are symmetric.
`baitsOnly`	A logical. If TRUE, the matrix will be indexed by the baits for both the rows and columns. If baitsOnly is TRUE, then baitsAsPrey must also be TRUE and symMat must be FALSE

Details

It is important to note that the weight of each directed edge is recorded by the number of instances bait b finds prey p.

Value

An adjacency matrix with weighting. The rows are indexed by those proteins sampled as baits (if symMat is true, only those rows with non-trivial row sums were sampled as baits) and the colunms are indexed by proteins detected as baits (et cetera).

Author(s)

T Chiang

Examples

```
library(ppiData)
data(y2hSysGW)
eg <- y2hSysGW[2:4]
lapply(eg, bpMatrix)
```

---

calcInOutDegStats *This function calculates the various degree statistic for a Protein-Protein Interaction (PPI) Graph*

Description

This function takes a graphNEL argument and calculates several degree statistics for which further analysis can be made.

Usage

```
calcInOutDegStats(graphObj, homodimer=FALSE)
```

Arguments

graphObj	The argument graphObj is an instance of the class graphNEL.
homodimer	A logical. If FALSE, the function removes all homodimer relationships. It retains them otherwise.

Value

The return value is a list of various degree statistics of the PPI directed graph:

inDegree	A named numeric vector. The names corresponds to the particular yeast Open Reading Frame (ORF), and each entry details how many edges flow towards that node, i.e. how many times each node was detected as a prey.
outDegree	A named numeric vector. The names corresponds to the particular yeast Open Reading Frame (ORF), and each entry details how many edges flow out from that node, i.e. how many prey was detected by each node as a bait.
inDegreeMinusOutDegree	A named numeric vector. This entry is the inDegree minus the outDegree entries, or the signed difference for each node as a prey to each node as a bait.
outDegreeMinusInDegree	A named numeric vector. This entry is the outDegree minus the inDegree entries, or the signed difference for each node as a bait to each node as a prey.
recipInDegree	A named numeric vector. Again the names represents the ORFs, while each entry corresponds to how many times each node was found as a prey in a reciprocated interaction.

recipOutDegree
: A named numeric vector. Again the names represents the ORFs, while each entry corresponds to how many times each node was found as a bait in a reciprocated interaction.

totalRecipDegree
: A numeric. For the PPI graph given, this argument gives the total number of reciprocated interactions found.

unrecipInDegree
: A named numeric vector. The names corresponds ORF and each entry details how many edges flow towards each node modulo the reciprocated interactions, i.e. for each node p, how many baits b detected p as a prey with the condition that p never detects b as a prey.

unrecipOutDegree
: A named numeric vector. The names corresponds ORF and each entry details how many edges flow out from each node modulo the reciprocated interactions, i.e. for each node b, how many prey p does b detect with the condition that b is never detected by p.

totalUnrecipDegree
: A numeric. The total number of unreciprocated interactions for the PPI graph given.

...

Author(s)

T Chiang

Examples

```
library(ppiData)
degStat <- calcInOutDegStats(Cagney2001BPGraph)
degStat$recipOutDeg
```

---

createSummaryTables

A function to create summary statistic tables for directed graphs

---

Description

This function takes a list of directed graph objects and creates a summary table based on the directed connectivity.

Usage

```
createSummaryTables(dataGraphs)
```

Arguments

dataGraphs	A named list of directed graphNELs

Value

A dataframe with the rows indexed by the names of each directed graphNEL and the columns indexed by viable baits (VB), viable prey (VP), viable bait/prey (VBP), the ratio of viable bait/prey by viable baits (VBP/VB), the ratio of viable prey by viable baits (VP/VB), the total number of directed interactions (TI), and the ratio of total intractions by viable baits (TI/VB).

Author(s)

T Chiang

Examples

```
graphs = lapply(bpExperimentNames, function(x) get(x))
names(graphs) = bpExperimentNames
createSummaryTables(graphs)
```

---

degreeEstimates *Estimate per-node degree in viable bait-prey (VBP) graph using maximum likelihood.*

Description

Estimate per-node degree in viable bait-prey (VBP) graph using maximum likelihood.

Usage

```
degreeEstimates(m,pTP,pFP)
findDegree(rd,ud,pTP,pFP,N)
degreePMF(deltahat,rd,ud,pTP,pFP,N)
```

Arguments

m	Square incidence matrix for VBP graph.
pTP	True positive probability.
pFP	False positive probability.
rd	Observed number of reciprocated edges.
ud	Observed number of unreciprocated edges.
deltahat	Estimate of node degree.
N	Number of proteins which were tested twice (e.g. both as viable bait and as viable prey.) Should equal the number of rows of m.

Details

`degreeEstimates` returns per-node degree estimates using the maximum likelihood method of Scholtens et al. (Submitted). It takes arguments `m`, which is an incidence matrix of bait-prey relationships, typically a VBP graph filtered for proteins prone to systematic bias, as well as pTP and pFP values, globally applicable to the entire graph.

`degreeEstimates` calls the function `findDegree` which estimates degree for a single node, given its observed number of reciprocated and unreciprocated incident edges. `findDegree` takes

an argument N which is the number of proteins in the graph that were tested twice. When degreeEstimates calls findDegree, N is set to the first dimension of the incidence matrix m.

degreePMF calculates the value of the pmf for an estimated degree, given observed numbers of reciprocated and unreciprocated edges, as well as pTP, pFP, and N. It is not generally called directly by the user. It is used to locate the maximum likelihood estimator for degree.

Value

degreeEstimates returns a named vector of degree estimates for each node in the graph.

findDegree returns a single degree estimate for one node.

degreePMF returns the value of the pmf for the multinomial model at a specified estimate of node degree, given the number of observed reciprocated and unreciprocated edges incident on the node and pTP and pFP.

Author(s)

Denise Scholtens, dscholtens@northwestern.edu

References

Scholtens D, Chiang T, Huber W, Gentleman R. Estimating node degree in bait-prey graphs. (Submitted).

Examples

```
findDegree(rd=2,ud=2,pTP=0.75,pFP=0.001,N=1000)
```

---

estErrProbMethodOfMoments

Estimate false positive and false negative error probabilities by method moments.

Description

Estimate false positive and false negative error probabilities by method moments.

Usage

```
estErrProbMethodOfMoments(nint, nrec, nunr, ntot)
```

Arguments

nint	Integer vector. True number of interactions. Typically, the function is called for a range of these, returning all possible solutions for that range.
nrec	Integer scalar. Observed number of reciprocated edges.
nunr	Integer scalar. Observed number of unreciprocated edges.
ntot	Integer scalar. Number of proteins which were tested twice (e.g. both as viable bait and as viable prey).

Details

The model is described in the vignette *Stochastic and systematic errors in PPI data, by looking at unreciprocated in- or out-edges* by W. Huber, T. Chiang and R. Gentleman.

Value

Matrix with 5 columns `nint` (a copy of the input argument), `pfp1`, `pfn1`, `pfp2` and `pfn2`, and as many rows as the length of `nint`.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

Examples

```
est = estErrProbMethodOfMoments(nint=seq(8000, 40000, by=100), nrec=9722, nunr=15856, ntc
if(interactive()) {
  plot(est[, c("pfp2", "pfn2")], type="l", col="blue", lwd=2,
       xlab=expression(p[FP]), ylab=expression(p[FN]))
  abline(h=0, v=0, lty=2)
}
```

---

estimateCCMErrorRates

Estimate false positive and false negative error probabilities

---

Description

Estimate false positive and false negative error probabilities for complex comembership edges using a protein complex interactome gold standard

Usage

```
estimateCCMErrorRates(m,GS,filterSystematic=TRUE,
          obsPropThresh=1,SystematicpThresh=.01)
```

Arguments

`m`	The bait to prey data adjacency matrix. Baits index the rows and prey index the columns.
`GS`	A gold standard protein complex interaction incidence matrix. Proteins index the rows and protein complexes index the columns.
`filterSystematic`	A logical. If TRUE, all baits with with highly uneven directed degree will be filtered out of the data.
`obsPropThresh`	A numeric between 0 and 1. The proportion of tested proteins found within a protein complex needed to keep that protein complex within the gold standard set.
`SystematicpThresh`	A numeric between 0 and 1. The p-value threshold by which systematic errors are filtered.

Details

The model is described in the manuscript *Estimating node degree in bait-prey graphs.* by D. Scholtens et al.

Value

A list:

`globalpTP`	A numeric between 0 and 1. Estimate of pTP.
`globalpTPSE`	A numeric. Estimate of standard error of globalpTP estimate.
`globalpFP`	A numeric between 0 and 1. Estimate of pFP.
`pTP95CI`	A vector of length 2. 95 percent confidence interval upper and lower bounds for globalpTP estimate.
`pFP95CI`	A vector of length 2. 95 percent confidence interval upper and lower bounds for globalpFP estimate.
`nEligComplexes`	A numeric. Number of complexes from GS that met obsPropThresh criteria.
`nEligBaits`	A numeric. Total number of eligible baits in GS set.
`nEligEdges`	A numeric. Total number of eligible edges in GS set.
`nBaitsInComplexes`	A vector. Number of baits in each eligible complex.
`complexSizes`	A vector. Size of each complex in GS set.

Author(s)

T. Chiang and D. Scholtens

References

Scholtens D, Chiang T, Huber W, Gentleman R. Estimating node degree in bait-prey graphs. *Bioinformatics*. To appear.

Examples

```
data(Ho2002BPGraph)
data(ScISIC)
Ho2002mat = as(Ho2002BPGraph,"matrix")
estimateCCMErrorRates(Ho2002mat,ScISIC)$globalpTP
```

`estimatePPIErrorRates`

Estimate false positive and false negative error probabilities for direct protein interactions

---

Description

Estimate false positive and false negative error probabilities for direct protein interactions using a protein interaction graph gold standard

Usage

```
estimatePPIErrorRates(matList,GSPos=NULL,GSNeg=NULL)
```

Arguments

matList	A named list of list. The names corresponds to a particular publication. Each element of the top list is a list of two matrices: 1. The adjacency matrix of the positive interaction datapoints. 2. The adjacency matrix of the tested interaction datapoints. The dimension names of these two matrices should be identical (an error will be thrown if this is not the case) where the rows are indexed by the bait proteins and the columns are indexed by the prey. The adjacency matrices are 0,1 matrices where an 1 entry signifies either a positive interaction relationship or a positive tested relationship. A 0 entry is then well defined.
GSPos	A positive gold standard protein interaction graph given by its adjacency matrix. The dimension names of this matrix must be the same identifying system used for the dimension names for the matrices found within the matList parameter. An entry of one in this matrix indicates a positive interaction while an entry of zero is inconclusive.
GSNeg	A negative gold standard protein interaction graph given by its adjacency matrix. The dimension names of this matrix must be the same identifying system used for the dimension names for the matrices found within the matList parameter. An entry of one in this matrix indicates a negative interaction while an entry of zero is inconclusive.

Details

The model is described in the manuscript *Estimating node degree in bait-prey graphs.* by D. Scholtens et al.

Value

A matrix.

The return value is an kx2 matrix where k is the length of matList. The first colum returns the estimated false positive error rates and the second column returns the estimated false negative error rates.

Author(s)

T. Chiang and L. Wang

References

Scholtens D, Chiang T, Huber W, Gentleman R. Estimating node degree in bait-prey graphs. *Bioinformatics*.

Examples

```
load(system.file("extdata", "intacty2hppMatrix.rda", package="ppiStats"))
load(system.file("extdata", "PPIpos", package="ppiStats"))
x = intacty2hppMatrix[c("EBI-389903","EBI-783101")]
estimatePPIErrorRates(x, GSPos=PPIpos)
```

`findAdjacent` *A function that returns the set of prey detected by a bait in a bait-prey experiment.*

Description

This function returns a list of the unique prey detected by a bait in a bait-prey experiment.

Usage

```
findAdjacent(x,bait)
```

Arguments

`x`	A graph object containing the set of bait-prey interaction data.
`bait`	A character specifying tjhe bait node of interest.

Value

A vector of prey that are detected by the bait in the graph, or NULL if the supplied bait is not a node in the graph.

Author(s)

Denise Scholtens

`genBPGraph` *A function to generate the protein-protein interaction (ppi) induced (un)directed graph*

Description

This function will take the ppi data and generate instance of the class graph.

Usage

```
genBPGraph(bpMat, directed=TRUE, bp=TRUE)
```

Arguments

`bpMat`	An adjacency matrix of PPI. If the matrix is obtained by empirical data, or bait to prey, then the rows are indexed by the baits and the columns indexed by the preys. If the rownames are not the same as the column names (i.e. a generic bait to prey matrix) the argument bp must be set to TRUE.
`directed`	A logical - if TRUE, the object will be a directed graph rather than an ordinary graph. For bait to prey interactions, this parameter must always be set to TRUE.
`bp`	A logical - if TRUE, it signifies that the adjacency matrix is a bait to prey empirically derived matrix so that the bait population (rownames) is usually different from the prey population (colnames).

Value

An instance of the class graph.

Author(s)

T Chiang

Examples

```
library(ppiData)
library(graph)
data(y2hSysGW)
eg=y2hSysGW[[3]]
egMat = bpMatrix(eg)
genBPGraph(egMat)
```

---

hgParams *A wrapper function to build a parameter class for the input of the HyperGTest.*

Description

This function takes a gene set and conducts test for either over or under representation of some category using the Hypergeometric distribution.

The two differences when building the parameter classes are that a conditional test can be performed on the GO dag but not on PFAM categories and an ontology can be assigned to GO but not to PFAM.

Usage

```
ppiBuildParams4GO(geneSet, universe, direction="over", annot="org.Sc.sgd",
                            ontology = "CC", cond=TRUE, pThresh = 0.01)
ppiBuildParams4PFAM(geneSet, universe, annot ="org.Sc.sgd",
                               direction = "over", pThresh=0.01)
```

Arguments

geneSet	A character vector of genes given by the gene locus name.
universe	The set of genes by which the geneSet is tested against for over/under representation. The genes are also given by the gene locus names.
direction	A character. This parameter can be either set to over or under when testing for GO categories.
annot	A character. The annotation package used.
ontology	A character: either CC, MF, or BP to describe the GO ontology.
cond	A logical. To test within the CO dag, a conditional hypergeometric test can be conducted.
pThresh	A numeric. A p-value threshold by which the null hypothesis is rejected.

Value

A object of class hyperGParams.

Author(s)

T Chiang

`hgTests` *A wrapper function to implement the Hypergeometric test, HyperGTest found withing the Category and GOstats packages.*

Description

This function takes the instances of the hyperGParams and conducts a (conditional) test for over/under representation of some category under the Hypergeometric distribution.

Usage

```
ppiHGTest4GO(parameter, filename, append=TRUE,
                      label = "Experiment name here",
                      typeGeneSet = "Describe the gene set here",
                      cs=50)

ppiHGTest4PFAM(parameter, filename, append = TRUE,
                           label = "Experiment Name Here",
                           typeGeneSet = "Describe the Gene Set Here",
                           cs = 50)
```

Arguments

`parameter`	An object of hyperGParams.
`filename`	A character vector. The name given to the .html file produced.
`append`	A logical. If multiple tests are conducted using the same filename, then the .html file will be appended if TRUE or over-written if FALSE.
`label`	A character. The should give a description of the experiment used to obtain the gene set.
`typeGeneSet`	A character: this character vector should adequately describe the gene set itself.
`cs`	A numeric. This gives a cut-off for the size of the categories in which we conduct the test.

Value

A instance of the class hyperGTest as well as an .html file.

Author(s)

T Chiang

idProteinErrorType *A function to identify those proteins affected by either stochastic or systematic errors*

Description

This function takes in either a bait to prey Graph (matrix) and, based on a binomial error model, partitions proteins identified as either affected by systematic or stochastic error. It is a wrapper function that will eventually call the qbinom function.

Usage

```
idSystematic(bpMat, viable, bpGraph = FALSE, pThresh = 0.01, pLevels =
1e-4, prob=0.5)
idStochastic(bpMat, bpGraph = FALSE, pThresh = 0.01, pLevels =
1e-4, prob=0.5)
```

Arguments

bpMat	Either a bait to prey directed graphNEL or its corresponding adjacency matrix.
viable	This is a character vector of viable proteins. It is only used in the idSystematic function.
bpGraph	A logical. If TRUE, than bpMat is passed in by the user as a graphNEL.
pThresh	The p-value threshold for which to partition stochastic or systematic errors
pLevels	A numeric. It gives the levels to calculate the countours of the function in p in the (n-in, n-out)-plane
prob	A numeric. The probability parameter in the call to the qbinom function.

Value

A character vector of proteins either affected by systematic or stochastic errors.

Author(s)

T Chiang

References

~put references to the literature/web site here ~

Examples

```
library(ppiData)
idSystematic(Ito2001BPGraph, viableBaits[[1]], bpGraph=TRUE)
```

idProteinType *A function to determine viable baits, viable preys, or homodimers within experimental data-sets.*

Description

These functions take a bait to prey directed graphNEL and returns either a character vector of all proteins which participates in homodimer relationships or a list of three character vectors: a vector of viable baits, a vector of viable prey, and a vector of the viable bait/prey.

Usage

```
idViableProteins(bpGraph, homomer=TRUE)
idHomodimers(bpGraph)
```

Arguments

bpGraph	A directed graphNEL
homomer	A logical. If True, homomer relationships will also be used to characterize proteins as viable baits, viable prey, and also as viable bait/prey simultaneously. If False, those proteins whose only intraction is a homomer relationship will not be characterized as viable in any context.

Value

The return value for idHomodimers is a character vector of those proteins which participates in homomer relationships.

The return value for idViableProteins is a list of two character vectors:

VB	A vector of baits that finds at least one prey in the experimental graphNEL, i.e. the nodes with out-degree at greater than 0. If homomer = FALSE, those proteins which participate in only homomer relationships will not be returned.
VP	A vector of prey which is found by at least one bait in the experimental graphNEL, i.e. the nodes with in-degree at greater than 0. If homomer = FALSE, those proteins which participate in only homomer relationships will not be returned.
VBP	A vector of proteins that are both viable baits and viable prey in an experimental dataset as defined above. If homomer = FALSE, those proteins which participate in only homomer relationships will not be returned.

Author(s)

T Chiang

Examples

```
library(ppiData)
idViableProteins(ItoCore2001BPGraph)
idHomodimers(ItoCore2001BPGraph)
```

`inOutScatterCharts` *A function to create scatter plots of the in-degree and out-degree for each vertex of a directed graph*

Description

This function takes a list of directed graph objects and plots the out-degree against the in-degree for each vertex. A binomial test is used to determine if there is a highly disproportionate in-degree or out-degree given a p-value threshold (the binomial test presumes p = 0.5).

Usage

```
inOutScatterCharts(dataGraphs, pThresh=0.01, pLevels=1e-4)
```

Arguments

`dataGraphs`	A named list of directed graphNELs
`pThresh`	The two-sided p-value threshold for the binomial testing for potentially biased nodes
`pLevels`	Figure out what this does

Value

For each directed data graph, this function generates 3 pdf scatter-plots and 1 eps scatter plot. The pdf files are: 1. A scatter plot of each nodes out- vs in-degree where points outside the staircase bundaries are those rejected in the binomial test. 2. A scatter plot of each nodes out- vs in-degree scaled by sqrt where points outside the conic region are those rejected in the binomial test. 3. A histogram for the distribution of p-values for each node

The eps file contains the same information as the second pdf file.

Author(s)

T Chiang

Examples

```
graphs = lapply(bpExperimentNames, function(x) get(x))
names(graphs) = bpExperimentNames
#inOutScatterCharts(graphs)
```

nullDistDoublyTestedEdges *Null distribution of number of reciprocated, unreciprocated and missing edges in stochastic model.*

Description

Calculate the null distribution of the number of reciprocated, unreciprocated and missing edges in a stochastic model where each edge is tested twice.

Usage

```
nullDistDoublyTestedEdges(deltaMax, n, pFP, pFN)
```

Arguments

deltaMax	Integer. Distributions will be calculated for model parameter *delta*=0, 1, 2, ..., deltaMax.
n	Integer. The parameter *n* of the model.
pFP	Numeric. The parameter *pFP* of the model.
pFN	Numeric. The parameter *pFN* of the model.

Details

The model is described in the vignette *Stochastic and systematic errors in PPI data, by looking at unreciprocated in- or out-edges* by W. Huber, T. Chiang and R. Gentleman.

This function can be quite slow, its runtime grows quickly with deltaMax (and is roughly independent of n, pFP, pFN). The example below should take only a few seconds on a reasonable computer, though.

Value

3d array with dimensions nMax+1 x nMax+1 x deltaMax+1 whose element p[nr+1, nu+1, delta+1] is the corresponding joint probability. nMax+1 is calculated (probably too conservatively) by the function to make sure that no probability leaks out of the array.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

Examples

```
p = nullDistDoublyTestedEdges(32, 1000, pFP=0.001, pFN=0.15)

if(interactive() && require("RColorBrewer"))
  for(k in 1:dim(p)[3]) {
    image(sqrt(p[,,k]), xlab=expression(N[rec]), ylab=expression(N[unrec]),
        main = expression(P(N[rec], N[unrec]~";"~ delta^"*", n, p[FP], p[FN])),
        x = 1:dim(p)[1], y = 1:dim(p)[2],
        col = colorRampPalette(brewer.pal(9, "GnBu"))(256))
    text(35, 35, paste("delta", k, sep="="))
  }
```

`separateExptBySize` *A function to partition experiments in user defined small, medium, large scale experiments*

Description

This function partitions protein interaction data into one of three classes defined by boundaries supplied by the users.

Usage

```
separateExptBySize(listOfGraphs, bound1, bound2)
```

Arguments

`listOfGraphs`	A list containing graphNEL objects to be sorted by the number of edges
`bound1`	An integer. The strict upper bound for the small scale experiments and the lower bound for the medium scale experiments.
`bound2`	An integer. The upper bound for the medium scale experiments, and the strict lower bound for the large scale experiments.

Details

A function to partition graphs based on the number of edges.

Value

A list:

`small`	A list of graph objects with the number of edges in each graph less than bound1.
`medium`	A list of graph objects with the number of edges in each graph of a value between that of bound1 and bound2.
`large`	A list of graph objects with the number of edges in each graph greater than bound2.

Author(s)

T. Chiang

`twPERM` *A function to compute a permutation test on a two way table.h*

Description

This function takes the entries of a two-way table and creates two binary logical vectors upon which it computes a permutation test to check for independence.

Usage

```
twowayPERM( v1, v2, NPERM, stat, seed = 123)
makeBinVect(n11, n12, n21, n22)
```

Arguments

`v1`	A logical vector.
`v2`	A logical vector the same length as v1.
`NPERM`	A positive integer. The number of tests to be conducted.
`stat`	A statistical function passed into to test the permutation on two vectors.
`seed`	A positive integer. To set the seed for the random number generator.
`n11`	A postive integer. The value of (1,1) in the two way table.
`n12`	A postive integer. The value of (1,2) in the two way table.
`n21`	A postive integer. The value of (2,1) in the two way table.
`n22`	A postive integer. The value of (2,2) in the two way table.

Value

A numeric vector containing the p-values for the test that the permutation of the vector v1 is independent from v2.

Author(s)

T Chiang

Examples

```
x <- makeBinVect(13, 17, 23, 71)
```

`viabilityCharts` *A function to create summary bar charts for directed graphs*

Description

This function takes a list of directed graph objects and creates a summary of the relative viable baits, viable prey, and viable bait/prey of each data graph.

Usage

```
viabilityCharts(dataGraphs, total=6466)
```

Arguments

`dataGraphs`	A named list of directed graphNELs
`total`	The total number of potential nodes (i.e. proteins) that could have been queried.

Value

A barchart object. Each item in the bar-chart represents one experimental data graph (set).

Author(s)

T Chiang

Examples

```
graphs = lapply(bpExperimentNames, function(x) get(x))
names(graphs) = bpExperimentNames
viabilityCharts(graphs)
```

Index